한글 FreeBSD 핸드북

Korea FreeBSD Users Group – http://www.kr.FreeBSD.org/

Storage: 스토리지

이 영 옥 <rick@rickinc.com> 역.

2004 년 8 월 7 일.
문서갱신번호 1.1

본 문서는 FreeBSD Documentation Project 에서 발행하는 FreeBSD Handbook 을 번역한 것으로 2004 년 8 월 7 일자 수정본을 기준으로 합니다. FreeBSD Handbook 은 지속적으로 갱신되기 때문에, 본 문서의 발간 이후에 갱신된 내용이 무엇인지 알고자 한다면, CVS Repository 를 조회하여 해당일 이후에 갱신된 내용만을 쉽게 찾아볼 수 있습니다. 또한 표지에 표시되는 날짜와 문서갱신번호는 이 문서의 갱신상태를 나타냅니다. 날짜는 번역의 대상이 되는 오리지날 핸드북의 최종 수정일자를 표시하며, 문서갱신번호는 문서에 수정이 있을 때 증가하므로 사용자가 이 문서의 갱신상태를 쉽게 파악할 수 있도록 합니다. 본 문서는 이영옥이 관리, 배포하므로 내용에 오탈자 및 오역이 발견될 경우엔 역자에게 알려주어 수정될 수 있도록 해 주십시오.

16 장 스토리지

16.1 계요

이번 장에서는 FreeBSD 에서 디스크 사용에 대해 다룬다. 여기에는 메모리와 네트워크 기반 디스크 그리고 표준 SCSI/IDE 스토리지 장치가 포함된다.

이번 장을 잃고 다음과 같은 사항을 알 수 있다:

- 물리적인 디스크에서(파티션과 슬라이스) 데이터 구성을 설명하기 위해 FreeBSD 에서 사용하는 용어.
- 시스템에 디스크를 어떻게 추가하는가
- USB 저장 장치를 사용하기 위해 FreeBSD 를 어떻게 설정하는가
- 메모리 디스크와 같은 가상 파일 시스템은 어떻게 설정하는가
- 디스크 사용량을 제한하기 위해 쿼타는 어떻게 사용하는가
- 침입에 대비하여 디스크는 어떻게 암호화하는가
- FreeBSD 에서 CD 와 DVD 는 어떻게 생성하는가
- 백업을 위한 다양한 스토리지 미디어 옵션
- FreeBSD 에서 백업 프로그램은 어떻게 사용하는가
- 플로피 디스크에 어떻게 백업하는가.
- 스냅샷이 무엇이고 효과적으로 사용방법은 무엇인가

이번 장을 읽기 전에 다음 사항을 알고 있어야 된다:

- 새로운 FreeBSD 커널을 어떻게 설정하고 설치하는지 알고 있어야 된다 (8 장).

16.2 장치 이름

다음은 FreeBSD 가 지원하는 물리적인 스토리지 장치 리스트고 장치 이름은 이들 장치와 관련이 있다.

테이블 16-1. 물리적인 디스크 이름 관례

드라이브 종류	드라이브 장치 이름
IDE 하드 드라이브	*ad*
IDE CDROM 드라이브	*acd*
SCSI 하드 드라이브와 USB 스토리지 장치	*da*
SCSI CDROM 드라이브	*cd*
여러 가지 비 표준 CDROM 드라이브	Mitsumi CD-ROM 은 *mcd*, Sony CD-ROM 은 *scd*, Matsushita/Panasonic CD-ROM 은 matcd ❖
Floppy 드라이브	fd
SCSI 테잎 드라이브	sa
IDE 테잎 드라이브	ast
Flash 드라이브	DiskOnChip® Flash 장치의 fla.
RAID 드라이브	Adaptec® AdvancedRAID 는 aacd, Mylex®는 mlxd 와 mlyd, AMI MegaRAID®는 amrd, Compaq Smart RAID 는 idad, 3ware® RAID 는 twed.
Notes: ❖matcd(4) 드라이버는 2002 년 10 월 5 일에 FreeBSD 4.X 분기에서 삭제되었기 때문에 FreeBSD 5.0 과 5.1 에는 없다. 그러나 이 드라이버는 2003 년 6 월 16 일부터 5.X 분기에 추가되었다.	

16.3 디스크 추가

드라이브가 하나만 있는 머신에 새로운 SCSI 디스크를 추가하려고 한다. 컴퓨터를 끄고 컴퓨터와 컨트롤러 그리고 드라이브 제조사의 매뉴얼대로 드라이브를 설치한다. 드라이브 설치는 아주 다양한 절차가 필요하기 때문에 자세한 설명은 이 문서의 범위를 벗어난다.

root 로 로그인 한다. 드라이브를 설치하고 /var/run/dmesg.boot 에서 새로운 디스크를 감지

했는지 확인한다. 예제를 계속하면 새로 추가된 드라이브는 da1 이 되고 우리는 /1 에(IDE 드라이브를 추가했다면 장치 이름은 4.0 이전 시스템에서는 wd1 이 되거나 대부분의 4.X 시스템에서는 ad1 이 된다.) 마운트하려고 한다.

FreeBSD 는 IBM-PC 호환 컴퓨터에서 운용되기 때문에 PC BIOS 파티션에 등록을 해야 된다. 이 의미는 전통적인 BSD 파티션과 다르다. PC 디스크는 4 개까지 BIOS 파티션 엔트리를 가지고 있다. 디스크를 FreeBSD 전용으로 사용한다면 *dedicated* 모드를 사용할 수 있다. 그렇지 않다면 FreeBSD 는 PC BIOS 파티션 중 하나가 된다. FreeBSD 는 PC BIOS 파티션을 슬라이스라고 부르기 때문에 전통적인 BSD 파티션과 혼동하지 않도록 한다. FreeBSD 전용 디스크에서도 슬라이스를 사용하겠지만 다른 운영체제가 설치된 컴퓨터에서도 사용된다. 따라서 다른 운영체제의 fdisk 유틸리티와 혼동하지 않는다.

슬라이스의 경우 드라이브는 /dev/da1s1e 처럼 추가된다. 이것을 SCSI 디스크, 유닛 번호 1(두 번째 SCSI 디스크), 슬라이스 1(PC BIOS 파티션 1) 그리고 BSD 파티션 e 로 읽는다. FreeBSD 전용의 경우 이 드라이브는 간단히 /dev/da1e 에 추가된다.

16.3.1 sysinstall(8)을 사용하여 디스크 추가하기

① **Sysinstall 탐색**

/stand/sysinstall 의 사용하기 쉬운 메뉴로 새로운 디스크를 파티션해서 라벨을 할당할 수 있다. root 유저 또는 su 명령을 사용하여 /stand/sysinstall 을 실행하고 *Configure* 메뉴에 들어간다. FreeBSD *Configuration* 메뉴에서 아래로 이동하여 *Fdisk* 옵션을 선택한다.

```
Disk name:      da0                                  FDISK Partition Editor
DISK Geometry:  4467 cyls/255 heads/63 sectors = 71762355 sectors (35040MB)

Offset       Size(ST)       End      Name  PType       Desc  Subtype    Flags

          0   71775284   71775283         -      6     unused        0

The following commands are supported (in upper or lower case):

A = Use Entire Disk   G = set Drive Geometry   C = Create Slice   F = `DD' mode
D = Delete Slice      Z = Toggle Size Units    S = Set Bootable   I = Wizard m.
T = Change Type       U = Undo All Changes     W = Write Changes
```

② fdisk 파티션 편집기

fdisk 안에서 A를 눌러 전체 디스크를 FreeBSD에 할당할 수 있다. "remain cooperative with any future possible operating systems" 라는 질문에 *YES*로 대답한다. W로 변경된 내용을 디스크에 저장하고 q를 눌러서 FDISK에서 빠져 나온다. 다음은 마스터 부트 레코드에 대한 질문이다. 실행중인 시스템에 디스크를 추가했기 때문에 *None*을 선택한다.

```
Disk name:      da0                                    FDISK Partition Editor
DISK Geometry:  4467 cyls/255 heads/63 sectors = 71762355 sectors (35040MB)

Offset       Size(ST)        End      Name  PType       Desc  Subtype    Flags

         0          63         62         -      6     unused        0
        63    71762292   71762354     da0s1      3     freebsd     165    C
  71762355       12929   71775283         -      6     unused        0

The following commands are supported (in upper or lower case):

A = Use Entire Disk   G = set Drive Geometry   C = Create Slice   F = `DD' mode
D = Delete Slice      Z = Toggle Size Units     S = Set Bootable   | = Wizard m.
T = Change Type       U = Undo All Changes      W = Write Changes
```

③ 디스크 라벨 편집기

sysinstall를 빠져 나온 후 다시 시작한다. 여기서 Label 옵션을 선택하더라도 위에서 설명한 곳으로 이동한다. *Disk Label Editor*에 들어간다. 여기서 전통적인 BSD 파티션을 생성한다. 하나의 디스크는 라벨 *a-h*로 8개의 파티션으로 나눌 수 있다. 파티션 라벨 중 몇 개는 특별하게 사용된다. *a* 파티션은 root 파티션(/)으로 사용된다. 그래서 시스템 디스크는(다시 말해 부팅할 수 있는 디스크) *a* 파티션을 가지고 있어야 된다. *b* 파티션은 스왑 파티션으로 사용되기 때문에 스왑 파티션으로 많은 디스크를 가지고 있을 것이다. *c* 파티션은 전체 디스크를 FreeBSD 전용 모드 또는 슬라이스 모드에서 전체를 FreeBSD 슬라이스로 사용함을 의미한다. 다른 파티션은 일반적으로 사용된다.

sysinstall의 라벨 편집기는 파티션 *e*를 root나 스왑 파티션으로 사용하지 않는다. 라벨 편집기에서 C를 입력해서 파일시스템 하나를 생성한다. 이것이 FS(파일시스템) 또는 swap 용인지 묻는 프롬프트가 나타나면 FS를 선택하고 마운트 포인트를(예: /mnt) 입력한다. 설치 후 모드에서 디스크를 추가 했을 때 sysinstall이

/etc/fstab 에 엔트리를 생성하지 않았기 때문에 마운트 포인트 지정은 중요하지 않다.

이제 W 를 입력해서 새로운 라벨을 디스크에 작성하고 파일시스템을 만든다. sysinstall 의 에러를 무시하면 새로운 파티션을 마운트할 수 없다. 라벨 편집기에서 나와서 sysinstall 를 완전히 빠져 나온다.

④ 끝내기

마지막 단계는 새로운 디스크 엔트리를 /etc/fstab 에 추가한다.

```
# See the fstab(5) manual page for important information on automatic mounts
# of network filesystems before modifying this file.
#
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/da0s1b             none            swap    sw              0       0
/dev/da0s1a             /               ufs     rw              1       1
/dev/da0s1f             /tmp            ufs     rw              2       2
/dev/da0s1g             /usr            ufs     rw              2       2
/dev/da0s1e             /var            ufs     rw              2       2
/dev/acd0c              /cdrom          cd9660  ro,noauto       0       0
proc                    /proc           procfs  rw              0       0
```

16.3.2 명령어 라인 유틸리티를 사용하여 디스크추가

16.3.2.1 슬라이스 사용

이 설정은 컴퓨터에 설치된 다른 운영체제와 디스크가 정확히 작동하도록 하기 때문에 다른 운영체제의 fdisk 유틸리티와 혼돈하지 안는다. 새로운 디스크를 설치할 때 이 방법을 사용하도록 권장한다. 이 방법을 사용하겠다면 *dedicated* 모드만 사용한다.

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# fdisk -BI da1 ❶
# disklabel -B -w -r da1s1 auto ❷
# disklabel -e da1s1 ❸
# mkdir -p /1
# newfs /dev/da1s1e ❹
# mount /dev/da1s1e /1 ❺
# vi /etc/fstab ❻

❶ 새로운 디스크를 초기화한다.
❷ 라벨을 할당한다.
❸ 방금 생성한 파티션의 라벨을 수정한다.
❹ 원하는 파티션을 생성할 때까지 이 명령을 반복한다.
❺ 파티션을 마운트한다.
❻ 적절한 엔트리를 /etc/fstab 에 추가한다.
```

IDE 디스크를 가지고 있으면 da 대신 ad 를 사용한다. 이전 FreeBSD 4.X 시스템에는 wd 를 사용한다.

16.3.2.2 FreeBSD 전용모드

다른 운영체제와 새로운 드라이브를 같이 사용하지 않는다면 *dedicated* 모드를 사용한다. 이 모드는 Microsoft 운영체제가 혼동할 수 있겠지만 손상을 입히지는 않을 것이다. 그러나 IBM 의 OS/2 는 이 파티션을 이해하지 못한다.

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# disklabel -Brw da1 auto
# disklabel -e da1                    ❶
# newfs -d0 /dev/da1e
# mkdir -p /1
# vi /etc/fstab                       ❷
```

```
# mount /1

❶ e 파티션을 생성한다.
❷ /etc/fstab 에 /dev/da1e 엔트리를 추가한다.


다른 방법은:

# dd if=/dev/zero of=/dev/da1 count=2
# disklabel /dev/da1 | disklabel -BrR da1 /dev/stdin
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab                    ❶
# mount /1

❶ /etc/fstab 에 /dev/da1e 엔트리를 추가한다.
```

Note: FreeBSD 5.1-릴리즈 이후부터 bsdlabel(8) 유틸리티가 예전 disklabel(8) 프로그램을 대체한다. bsdlabel(8)로 쓰지 않는 다양한 옵션과 매개변수가 없어졌다; 위의 예제에서 옵션 *-r*은 bsdlabel(8)로 제거됐다. 더 많은 정보는 bsdlabel(8) 매뉴얼 페이지를 참고한다.

16.4 RAID

16.4.1 소프트웨어 레이드

16.4.1.1 연속된 디스크 드라이버 (CCD) 설정

대형 스토리지 솔루션을 선택할 때 가장 중요한 요소는 속도, 안정성 그리고 가격을 검토해야 된다. 3 가지를 고루 갖춘 제품은 찾기가 힘들다; 보통 빠르고 안정적인 대형 스토리지 장치는 비싸기 때문에 비용을 줄이기 위해 속도나 안정성을 감소시켜야 된다.

아래에서 설명하는 시스템 디자인은 가격을 최우선으로 꼽고 속도와 안정성을 따진다. 이

시스템에서 데이터 전송 속도는 결국 네트워크에 따라 다르다. 반면 안정성이 매우 중요하기 때문에 아래에서 설명하는 CCR 드라이브는 이미 CD-R 에 풀 백업되어 쉽게 대체할 수 있는 온라인 데이터를 제공하다.

필요한 사항을 규정하는 것이 대형 스토리지 솔루션 선택의 첫 번째 단계다. 속도나 안정성을 비용보다 중요시 한다면 이번 섹션에서 설명하는 시스템과 다를 것이다.

16.4.1.2 하드웨어 설치

대략 90GB 의 온라인 스토리지를 제공하기 위해 아래에서 설명하는 CCD 디스크의 코어 방식으로 Western Digital 30GB 5400 RPM IDE 디스크를 시스템에 추가한다. 최고의 성능을 위해 각 IDE 디스크는 각각의 컨트롤러와 케이블에 연결되어 있어야 되지만 최소 가격을 위해 추가적인 IDE 컨트롤러를 사용하지 않는다. 대신 디스크를 점퍼로 설정하여 각 IDE 컨트롤러에 하나는 마스터를 다른 하나는 슬레이브를 연결한다.

재 부팅하면 시스템 BIOS 는 자동으로 추가된 디스크를 감지한다. 더욱 중요한 것은 FreeBSD 가 재 부팅할 때 디스크를 검색한다는 것이다:

```
ad0: 19574MB <WDC WD205BA> [39770/16/63] at ata0-master UDMA33
ad1: 29333MB <WDC WD307AA> [59598/16/63] at ata0-slave UDMA33
ad2: 29333MB <WDC WD307AA> [59598/16/63] at ata1-master UDMA33
ad3: 29333MB <WDC WD307AA> [59598/16/63] at ata1-slave UDMA33
```

> Note: FreeBSD 가 모든 디스크를 감지하지 못했다면 점퍼를 정확하게 설정했는지 확인한다. 대부분의 IDE 드라이브는 "케이블 선택" 방식의 점퍼를 가지고 있다. 이것은 마스터/슬레이브와 관련된 점퍼가 아니다. 정확한 점퍼를 확인하기 위해 드라이브 문서를 참고한다.

파일시스템에 이들을 어떻게 추가할지 생각한다. vinum(8)과(17 장) ccd(4)를 사용한다. 여기서는 ccd(4)를 사용한다.

16.4.1.3 CCD 설정

ccd(4) 드라이버는 몇 개의 동일한 디스크를 연결하여 하나의 논리적인 파일시스템으로 만들 수 있다. ccd(4)를 사용하려면 ccd(4)를 지원하도록 커널을 빌드해야 된다. 다음 라인을 커널 설정파일에 추가하고 커널을 다시 빌드해서 설치한다:

```
pseudo-device    ccd      4
```

5.X 시스템에서는 다음 라인을 대신 사용한다:

```
device    ccd
```

> Note: FreeBSD 5.X 에서는 ccd(4) 장치 드라이버가 자가 복제를 하기 때문에 ccd(4) 장치 개수를 지정할 필요가 없다. 새로운 장치 인스턴스는 요청이 있으면 자동으로 생성된다.

FreeBSD 3.0 또는 이후 버전에서 ccd(4) 지원은 커널이 로드할 수 있는 모듈처럼 로드할 수 있다.

ccd(4)를 설정하려면 첫째로 디스크에 라벨을 할당하기 위해 disklabel(8)을 사용해야 된다:

```
# disklabel -r -w ad1 auto
# disklabel -r -w ad2 auto
# disklabel -r -w ad3 auto
```

이 명령은 ad1c, ad2c 그리고 ad3c 를 하나의 디스크로 연결하도록 디스크 라벨을 생성한다.

다음 단계는 디스크 라벨 종류를 변경한다. 디스크를 편집하기 위해 disklabel(8)을 사용할 수 있다:

```
# disklabel -e ad1
# disklabel -e ad2
# disklabel -e ad3
```

이 명령은 EDITOR 환경변수에 지정된 에디터로(보통 vi(1)) 각 디스크의 현재 디스크 라벨을 연다.

수정하지 않은 디스크 라벨은 다음과 비슷하게 보인다:

```
8 partitions:
#          size     offset     fstype    [fsize bsize bps/cpg]
  c: 60074784          0     unused          0     0     0   # (Cyl.     0 - 59597)
```

ccd(4)가 사용할 수 있도록 새로운 파티션 *e*를 추가한다. 이것은 보통 *c* 파티션에서 복사할 수 있지만 *fstype*은 4.2BSD로 만들어야 한다. 이제 디스크 라벨은 다음과 비슷하게 보인다:

```
8 partitions:
#          size     offset     fstype    [fsize bsize bps/cpg]
  c: 60074784          0     unused          0     0     0   # (Cyl.     0 - 59597)
  e: 60074784          0     4.2BSD          0     0     0   # (Cyl.     0 - 59597)
```

16.4.1.4 파일시스템 생성

ccd0c의 장치 노드가 없기 때문에 다음 명령을 실행해서 생성해야 된다:

```
# cd /dev
# sh MAKEDEV ccd0
```

Note: FreeBSD 5.0에서 devfs(5)가 /dev의 장치 노드를 자동으로 관리하기 때문에 MAKEDEV는 필요없다.

이제 모든 디스크 라벨을 할당했으므로 ccd(4)를 빌드해야 된다. 빌드하려면 다음과 비슷한 옵션으로 ccdconfig(8)를 사용한다:

```
# ccdconfig ccd0❶ 32❷ 0❸ /dev/ad1e❹ /dev/ad2e /dev/ad3e
```

각 옵션의 사용과 의미는 다음과 같다:

❶ 첫 번째 인자는 이 예제의 경우 /dev/ccd0c 인 설정하려는 장치를 의미한다. /dev/ 부분은 옵션이다.

❷ 파일시스템의 인터리브(interleave: 성능을 높이기 위해 데이터가 서로 인접하지 않도록 배열하는 방법). 인터리브는 디스크 블록에서 스트라이프(stripe) 크기를 정의하고 보통 512 바이트다. 그래서 32 의 인터리브는 16,384 바이트가 된다.

❸ ccdconfig(8)의 플래그. 드라이브 미러링을 원한다면 여기에 특정 플래그를 지정할 수 있다. 이 설정은 ccd(4)에 미러링을 제공하지 않기 때문에 0 으로 설정한다.

❹ ccdconfig(8)의 마지막 인자는 장치를 어레이로 만든다. 각 장치는 절대 경로를 사용한다.

ccdconfig(8)를 실행하면 ccd(4)가 설정된다. 그래서 파일시스템을 설치할 수 있다. 옵션은 newfs(8)를 참고하거나 단순히 다음 명령을 실행한다:

```
# newfs /dev/ccd0c
```

16.4.1.5 자동화하기

보통 재 부팅할 때마다 ccd(4)를 자동으로 마운트하려면 다음 명령으로 현재 설정을 /etc/ccd.conf 에 작성한다:

```
# ccdconfig -g > /etc/ccd.conf
```

/etc/cd.conf 가 있다면 재 부팅하는 동안 /etc/rc 스크립트는 ccdconfig -C 를 실행한다. 따라서 자동으로 ccd(4)를 설정하여 마운트할 수 있다.

Note: ccd(4)를 mount(8)하기 전에 싱글 유저모드로 부팅했다면 어레이를 설정하도록 다음 명령을 실행해야 된다:

```
# ccdconfig -C
```

자동으로 ccd(4)를 마운트하도록 ccd(4) 엔트리를 /etc/fstab 에 생성해서 부팅할 때 마운트 한다:

```
/dev/ccd0c              /media      ufs     rw      2       2
```

16.4.1.6 The Vinum 볼륨 매니저

가상 디스크 드라이브 툴인 Vinum 볼륨 매니저는 블록 장치 드라이버다. 디스크 하드웨어를 블록 장치 인터페이스와 맵 데이터에서 분리하여 전통적인 디스크 스토리지와 비교하여 유연성과 성능 그리고 안정성을 증가시켰다. vinum(8)은 RAID-0, RAID-1 과 RAID-5 모델을 따로 또는 결합하여 구성한다.

vinum(8)에 대한 더 많은 정보는 17 장을 본다.

16.4.2 하드웨어 RAID

그리고 FreeBSD 는 다양한 하드웨어 레이드 컨트롤러를 지원한다. 이런 장치는 FreeBSD 에 어레이 관리를 위한 소프트웨어 없이 RAID 서브시스템을 제어한다.

카드에 있는 BIOS 를 사용하여 카드는 디스크의 대부분을 제어한다. 다음은 Promise IDE RAID 컨트롤러를 사용하기 위한 설정을 간단히 요약한 것이다. 이 카드를 설치하면 시스템이 시작될 때 필요한 정보를 요청하는 프롬프트를 보여준다. 지시에 따라 카드의 설정 화면으로 들어간다. 여기서는 모든 드라이브를 결합하는 기능을 사용해서 디스크가 FreeBSD 에 하나의 드라이브처럼 보인다. 다른 RAID 레벨도 적절히 설정할 수 있다.

16.4.3 ATA RAID1 어레이 다시 빌드하기

FreeBSD 는 어레이에서 문제가 발생한 디스크를 바로 교체할 수 있게 한다. 이것은 재 부팅하기 전에 감지해야 된다.

다음과 비슷한 메시지를 /var/log/messages 또는 dmesg(8) 결과에서 볼 수 있다:

```
ad6 on monster1 suffered a hard error.
ad6: READ command timeout tag=0 serv=0 - resetting
ad6: trying fallback to PIO mode
```

```
ata3: resetting devices .. done
ad6: hard error reading fsbn 1116119 of 0-7 (ad6 bn 1116119; cn 1107 tn 4 sn 11)
status=59 error=40
ar0: WARNING - mirror lost
```

atacontrol(8)로 더 많은 정보를 체크한다:

```
# atacontrol list
ATA channel 0:
    Master:     no device present
    Slave:    acd0 <HL-DT-ST CD-ROM GCR-8520B/1.00> ATA/ATAPI rev 0

ATA channel 1:
    Master:     no device present
    Slave:      no device present

ATA channel 2:
    Master:   ad4 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:      no device present

ATA channel 3:
    Master:   ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:      no device present

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

① 첫째로 어레이에서 디스크를 분리하여 안전하게 제거할 수 있다:

```
# atacontrol detach 3
```

② 디스크를 분리한다.

③ 예비 디스크로 교체한다:

```
# atacontrol attach 3
Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
Slave:   no device present
```

④ 어레이를 다시 빌드한다:

```
# atacontrol rebuild ar0
```

⑤ 다시 빌드하는 명령은 끝날 때까지 멈춰있다. 그렇지만 다른 터미널을(Alt+Fn 사용) 열고 다음 명령으로 진행 사항을 체크할 수 있다:

```
# dmesg | tail -10
[output removed]
ad6: removed from configuration
ad6: deleted from ar0 disk1
ad6: inserted into ar0 disk1 as spare

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: REBUILDING 0% completed
```

⑥ 이 작업이 끝날 때까지 기다린다.

16.5 USB 스토리지 장치

요즘 수많은 외장형 스토리지 솔루션들이 Universal Serial Bus(USB)를 사용한다: 하드 드라이브, USB 스토리지, CD-R 레코더 등. FreeBSD 는 이들 장치를 지원한다.

16.5.1 설정

USB 스토리지 장치 드라이버 umass(4)가 USB 스토리지 장치를 지원한다. GENERIC 커널을 사용한다면 설정을 변경할 필요 없다. 사용자 커널을 사용한다면 커널 설정파일에 다음 라인들이 필요하다:

```
device scbus
device da
device pass
device uhci
device ohci
device usb
device umass
```

umass(4) 드라이버는 USB 스토리지 장치를 사용하기 위해 SCSI 서브시스템을 사용하므로 USB 장치는 시스템에서 SCSI 장치처럼 보인다. 마더보드의 USB 칩셋에 따라 *device uhci* 와 *device ohci* 중 하나만 필요하지만 커널 설정파일에 둘 다 가지고 있어도 문제는 없다. 라인을 추가하고 새로운 커널을 컴파일해서 설치해야 된다.

> **Note:** USB 장치가 CD-R 레코더면 SCSI CD-ROM 드라이버 cd(4)를 다음 라인으로 커널에 추가해야 된다:
>
> ```
> device cd
> ```
>
> 레코더가 SCSI 드라이브처럼 보이기 때문에 atapicam(4) 드라이버를 커널 설정에 사용하지 않는다.

USB 2.0 컨트롤러는 FreeBSD 5.X 와 FreeBSD 4.X 는 FreeBSD 4.10-RELEASE 부터 지원한다. 다음 라인을 커널 설정파일에 추가한다:

```
device ehci
```

USB 1.X 를 지원하려면 uhci(4)와 ohci(4) 드라이버가 필요하다.

> **Note:** FreeBSD 4.X 에서 USB 데몬(usbd(8))이 USB 장치를 감지할 수 있도록 실행

해야 된다. 이렇게 하려면 *usbd_enable="YES"*를 /etc/rc.conf 파일에 넣고 머신을 재 부팅한다.

16.5.2 설정 테스트

설정을 테스트할 준비가 됐다: USB 장치를 꽂으면 시스템의 메시지 버퍼에(dmesg(8)) 다음과 같이 드라이브가 나타난다:

```
umass0: USB Solid state disk, rev 1.10/1.00, addr 2
GEOM: create disk da0 dp=0xc2d74850
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <Generic Traveling Disk 1.11> Removable Direct Access SCSI-2 device
da0: 1.000MB/s transfers
da0: 126MB (258048 512 byte sectors: 64H 32S/T 126C)
```

물론 장치 노드(da0)와 다른 사항은 여러분의 설정과 다르다.

USB 장치가 SCSI 장치로 보이기 때문에 시스템에 꽂혀 있는 USB 스토리지 장치를 보기 위해 camcontrol 명령을 사용할 수 있다:

```
# camcontrol devlist
<Generic Traveling Disk 1.11>          at scbus0 target 0 lun 0 (da0,pass0)
```

장치가 파일시스템이면 마운트할 수 있다. 필요하다면 16.3 장을 따라 USB 드라이브에 파티션을 생성하고 포맷할 수 있다.

장치를 빼면(디스크를 먼저 언마운트 한다) 시스템 메시지 버퍼에서 다음과 비슷한 내용을 보게 된다:

```
umass0: at uhub0 port 1 (addr 2) disconnected
(da0:umass-sim0:0:0:0): lost device
(da0:umass-sim0:0:0:0): removing device entry
GEOM: destroy disk da0 dp=0xc2d74850
umass0: detached
```

16.5.3 더 많은 정보

디스크 추가(16.3 장), 파일시스템 마운트 및 언마운트 섹션(3.6 장)과 다양한 매뉴얼 페이지가 유용하다: umass(4), camcontrol(8), 그리고 usbdevs(8).

16.6 광학 미디어 생성과 사용 (CD & DVD)

16.6.1 소개

CD 는 전통적인 디스크와 다른 여러 가지 특징을 가지고 있다. 처음에는 CD 에 유저가 데이터를 작성할 수 없었다. CD 는 헤드가 트랙 사이를 움직이는 동안 지체되지 않고 계속해서 읽을 수 있도록 디자인되었다. CD 는 시스템 사이에서 그 당시 비슷한 크기의 다른 미디어보다 쉽게 데이터를 전송했다.

CD 는 트랙을 가지고 있지만 계속해서 읽기 위한 데이터의 섹션을 의미할 뿐 디스크의 물리적인 특성을 의미하지 않는다. FreeBSD 에서 CD 를 만들려면 CD 트랙에 생성할 데이터 파일을 준비하여 CD 트랙에 작성한다.

ISO 9660 파일시스템은 이런 이유로 디자인되었다. 불행히 일반적인 것보다 파일시스템 분류에 한계가 있다. 다행히 CD 를 알맞게 작성할 수 있도록 이런 한계를 보충하는 확장 메커니즘을 제공하지만 이런 확장을 지원하지 않는 시스템이 아직도 존재한다.

sysutils/mkisofs 프로그램은 ISO 9660 파일시스템을 가지고 있는 데이터 파일을 생성하는데 사용된다. 이 프로그램은 다양한 확장을 지원하는 옵션을 가지고 있고 아래에서 설명할 것이다. sysinstalls/mkisofs 포트에서 설치할 수 있다.

CD 레코더의 방식에 따라 CD를 레코드하는 툴이 결정된다. ATAPICD 레코더는 기본 시스템에 포함된 burncd 프로그램을 사용한다. SCSI와 USB CD레코더는 sysutils/cdrtools 포트에서 cdrecord를 사용한다.

burncd 는 지원하는 드라이브 수에 제한이 있다. 드리이브가 지원되는지 보려면 지원하는

CD-R/RW 드라이브 리스트를 본다(http://www.freebsd.dk/ata/).

> Note: FreeBSD 4.8-릴리즈 버전이나 더 높은 FreeBSD 5.X 를 사용한다면 cdrecord 와 ATAPI/CAM 모듈의 ATAPI 하드웨어에서 SCSI 드라이버용도 사용할 수 있다.

그래픽 유저 인터페이스(GUI) CD 레코딩 소프트웨어를 찾는다면 X-CD-Roast나 K3b을 확인해 본다. 이들 툴은 패키지나 sysutils/xcdroast와 sysutils/k3b 포트에서 사용할 수 있다. X-CD-Roast와 K3b는 ATAPI 하드웨어와 ATAPI/CAM 모듈이 필요하다.

16.6.2 mkisofs

sysutils/mkisofs는 유닉스 파일시스템에 디렉터리 트리 이미지인 ISO 9660 파일시스템을 생성한다. 간단한 사용 법은 다음과 같다:

```
# mkisofs -o imagefile.iso /path/to/tree
```

이 명령은 /path/to/tree 를 복사한 ISO 9660 파일시스템을 가지고 있는 imagefile.iso 를 생성한다. 이 과정에서 표준 ISO 9660 파일시스템에 맞는 이름으로 파일 이름을 변경하고 ISO 파일시스템 특성이 아닌 이름을 가진 파일은 제외시킨다.

다양한 옵션으로 이런 한계를 극복할 수 있다. 특히 *-R*은 유닉스 시스템에 일반적인 Rock Ridge 를 확장, *-J*는 Microsoft 시스템이 사용하는 Joliet 확장을 활성하고 *-hfs* 는 MacOS 가 사용하는 HFS 파일시스템을 생성하는데 사용할 수 있다.

FreeBSD 시스템에서만 사용하려는 CD 는 *-U*로 모든 파일이름 제한을 해결할 수 있다. *-R*을 사용하면 ISO 9660 표준을 깨는 방법이라도 여러분이 시작한 FreeBSD 트리와 동일한 파일시스템 이미지를 만든다.

보통 사용하는 마지막 옵션은 *-b* 다. 이 옵션은 "El Torito" 부팅 가능한 CD 를 만들 때 사용하도록 부트 이미지 위치를 지정할 때 사용된다. 이 옵션은 CD 로 작성할 트리의 최상단에서 부트 이미지 경로를 인자로 갖는다. 지정한 /tmp/myboot 는 /tmp/myboot/boot/cdboot 의 부트 이미지로 부팅 가능한 FreeBSD 시스템을 가지고 있다. 따라서 다음과 같은 명령으로 /tmp/bootable.iso 에 ISO 9660 파일시스템 이미지를 만들 수 있다:

```
# mkisofs -U -R -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

위의 명령이 끝나고 커널에 vn(FreeBSD 4.X)이나 md(FreeBSD 5.X)를 설정하였다면 다음 명령으로 파일시스템을 마운트할 수 있다:

FreeBSD 4.X 는 아래 명령을 사용한다:

```
# vnconfig -e vn0c /tmp/bootable.iso
# mount -t cd9660 /dev/vn0c /mnt
```

FreeBSD 5.X 에서는 다음 명령을 사용한다:

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

여기서 /mnt 와 /tmp/myboot 가 동일함을 확인할 수 있다.

미세한 조정을 위해 sysinstalls/mkisofs에 사용할 수 있는 다양한 옵션이 있다. 특히 ISO 9660 레이아웃을 수정하고 Joliet 와 HFS 디스크를 작성할 수 있다. 더 자세한 사항은 mkisofs(8) 매뉴얼 페이지를 본다.

16.6.3 CD 굽기

ATAPI CD 레코더를 가지고 있다면 ISO 이미지를 CD 로 레코드하기 위해 burncd 명령을 이용할 수 있다. burncd 는 기본 시스템의 /usr/sbin/burncd 에 설치되어 있다. 옵션이 몇 개 없으므로 사용법은 매우 단순하다:

```
# burncd -f cddevice data imagefile.iso fixate
```

위 명령은 *imageifle.sio* 를 *cddevice* 에 레코드한다. 기본 장치는 /dev/acd0c 다. 레코드 속도와 레코드 후 CD 꺼내기 그리고 오디오 데이터 작성에 대한 설정 옵션은 burncd(8)을 확인한다.

16.6.4. cdrecord

ATAPI CD 레코더를 가지고 있지 않다면 CD를 레코드하기 위해 cdrecord를 사용해야 된다. cdrecord는 기본 시스템에 없어서 sysutils/cdrtools 포트나 적당한 패키지로 설치해야 된다. 기본 시스템을 변경하면 이 프로그램의 바이너리에 문제가 발생할 수 있다. 그래서 시스템을 업그레이드할 때 포트를 업그레이드 하거나, STABLE 버전을 사용한다면 새로운 버전을 사용할 수 있을 때 포트를 업그레이드 한다.

cdrecord 는 많은 옵션을 가지고 있지만 기본 사용법은 burncd 보다 간단하다. ISO 9660 레코드는 다음 명령을 이용한다:

```
# cdrecord dev=device imagefile.iso
```

cdrecord 를 제대로 이용하는 방법은 사용할 *dev* 를 찾을 때 사용한다. 적당한 설정을 찾으려면 다음과 비슷한 결과를 출력하는 cdrecord 에 *-scanbus* 플래그를 사용한다.

```
# cdrecord -scanbus
Cdrecord 1.9 (i386-unknown-freebsd4.2) Copyright (C) 1995-2000 Jorg Schilling
Using libscg version 'schily-0.1'
scsibus0:
        0,0,0     0) 'SEAGATE ' 'ST39236LW        ' '0004' Disk
        0,1,0     1) 'SEAGATE ' 'ST39173W         ' '5958' Disk
        0,2,0     2) *
        0,3,0     3) 'iomega  ' 'jaz 1GB          ' 'J.86' Removable Disk
        0,4,0     4) 'NEC     ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
0,5,0     5) *
        0,6,0     6) *
        0,7,0     7) *
scsibus1:
        1,0,0   100) *
        1,1,0   101) *
        1,2,0   102) *
        1,3,0   103) *
        1,4,0   104) *
        1,5,0   105) 'YAMAHA  ' 'CRW4260        ' '1.0q' Removable CD-ROM
        1,6,0   106) 'ARTEC   ' 'AM12S          ' '1.06' Scanner
```

```
        1,7,0    107) *
```

리스트에 있는 장치의 *dev* 값으로 이 리스트가 적당하다. CD 레코더를 설치하고 *dev* 값에 컴마(,)로 나뉜 3 개의 번호를 사용한다. 여기서 CRW 장치는 1,5,0 이기 때문에 적당한 입력은 *dev=1,5,0* 이다. 이 값을 지정하는 쉬운 방법은 cdrecord(1) 매뉴얼 페이지에 있다. 또한 오디오 트랙 작성, 속도 제어와 다른 것에 대한 정보도 찾을 수 있다.

16.6.5 오디오 CD 복제

CD 에서 오디오 데이터를 연속된 파일로 추출해서 공 CD 에 레코드하여 오디오 CD 를 복제할 수 있다. 이 과정은 ATAPI 와 SCSI 드라이브에서 약간 다르다.

SCSI 드라이브

① 오디오를 추출하기 위해 cdda2wav 를 사용한다

```
% cdda2wav -v255 -D2,0 -B -Owav
```

② .wav 파일을 레코드하기 위해 cdrecord 를 사용한다.

```
% cdrecord -v dev=2.0 -dao -useinfo  *.wav
```

16.6.4 장에서 설명했듯이 *2.0* 이 적당한 설정이다.

ATAPI 드라이브

① ATAPI CD 드라이버는 각 트랙을 /dev/acd*d*t*nn* 에 작성한다. *d* 는 드라이브 번호고 *nn* 은 필요하다면 0 을 앞에 붙인(예: 02) 두 개의 10 진수로 작성하는 트랙번호다. 따라서 첫 번째 디스크의 첫 번째 트랙은 /dev/acd0t01, 두 번째는 /dev/acd0t02, 세 번째는 /dev/acd0t03 등으로 변환된다.

/dev 에 적당한 파일이 있는지 확인한다.

```
# cd /dev
# sh MAKEDEV acd0t99
```

Note: FreeBSD 5.0 에서 devfs(5)가 /dev 의 엔트리를 자동으로 생성하고 관리하기 때문에 MAKEDEV 는 필요 없다.

② dd(1)로 각 트랙을 추출한다. 파일을 추출할 때 블록 크기도 지정해야 된다.

```
# dd if=/dev/acd0t01 of=track1.cdr bs=2352
# dd if=/dev/acd0t02 of=track2.cdr bs=2352
...
```

③ burncd 로 추출한 파일을 레코드한다. 오디오 파일임을 지정해야 되고 burncd 는 레코딩이 끝나고 디스크를 닫는다.

```
# burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
```

16.6.6 데이터 CD 복제

데이터CD를 sysutils/mkisofs로 생성한 이미지 파일과 기능적으로 동등한 이미지 파일로 복사할 수 있고 이 방법으로 어떤 데이터 CD라도 복제할 수 있다. 여기 주어진 예제의 CDROM 장치는 acd0 다. 실제 CDROM 장치로 변경한다. FreeBSD 4.X에서 *c*는 전체 파티션 또는 이 경우 CDROM이고 전체 디스크를 표시하기 위해 장치 이름 끝에 붙인다.

```
# dd if=/dev/acd0c of=file.iso bs=2048
```

이제 이미지를 갖게 되었고 위에서 설명한대로 CD 에 레코드한다.

16.6.7 데이터 CD 사용하기

이제 표준 데이터 CDROM 을 만들었기 때문에 마운트해서 데이터를 읽으려고 할 것이다. 기본적으로 mount(8)은 파일시스템을 *ufs* 로 생각한다. 다음 명령을 사용하면 "Incorrect super block"이라는 메시지가 나타나고 마운트되지 않는다.

```
# mount /dev/cd0c /mnt
```

CDROM 은 *UFS* 파일시스템이 아니기 때문에 마운트하려는 시도는 위와 같이 실패한다. mount(8)에 파일시스템이 ISO966 이라고 지정하면 모든 것이 정상으로 동작한다. mount(8)에 옵션 *-t cd9660* 을 지정하면 된다. 예를 들어 CDROM 장치 /dev/cd0c 를 /mnt 에 마운트하려면 다음 명령을 사용한다:

```
# mount -t cd9660 /dev/cd0c /mnt
```

CDROM 이 사용하는 인터페이스에 따라 장치 이름이(이 예제에서는/dev/cd0c) 다를 수 있다. 또한 *-t cd9660* 옵션은 mount_cd9660(8)를 실행한다. 위의 예제를 짧게 만들 수 있다:

```
# mount_cd9660 /dev/cd0c /mnt
```

이 방법으로 어떤 벤더의 데이터 CDROM이라도 보통 사용할 수 있다. 그러나 특정 ISO 9660 으로 확장된 디스크는 이상하게 수행된다. 예를 들어 Joliet 디스크는 모든 파일 이름을 2-byte 유니코드 문자로 저장한다. FreeBSD커널은 아직 유니코드를 지원하지 못하기 때문에 영문자가 아닌 문자는 물음표(?)로 나타난다(FreeBSD 4.3 이나 이후 버전을 사용한다면 CD9660 드라이버는 적당한 유니코드 변환 테이블을 가지고 있다. 몇몇 일반적인 암호화 모듈은 sysutils/cd9660_unicode 포트에서 사용할 수 있다).

때때로 CDROM 을 마운트할 때 “Device not configured”를 보게 될 것이다. 이 의미는 보통 트레이에 디스크가 없거나 버스에 드라이브가 없는 것으로 CDROM 드라이브가 인지한다. CDROM 드라이브가 실제로 확인하려면 몇 초 정도가 소요되므로 끈기 있게 기다린다.

종종 SCSI CDROM 은 버스 리셋에 응답할 충분한 시간이 없기 때문에 실패한다. SCSI CDROM 을 가지고 있다면 커널 설정에 다음 옵션을 추가하고 커널을 다시 빌드한다.

```
options SCSI_DELAY=15000
```

이것은 SCSI 버스에게 부팅 시 15 초 동안 기다리게 하여 CDROM 드라이브에게 버스 리셋에 응답할 수 있는 모든 가능한 기회를 주기 위해서다.

16.6.8 raw 데이터 CD 레코드

ISO 9660 파일시스템을 생성하지 않고 파일을 직접 CD 로 레코드하도록 선택할 수 있다. 어떤 사람들은 백업 목적으로 이렇게 하고 있다. 이것은 표준 CD 레코드보다 더 빨리 실행된다:

```
# burncd -f /dev/acd1c -s 12 data archive.tar.gz fixate
```

데이터를 CD 같은 곳에 저장하려면 데이터를 raw 장치 노드에서 읽어야 한다:

```
# tar xzvf /dev/acd1c
```

일반 CDROM처럼 이 디스크를 마운트할 수 없다. 이런 CDROM은 FreeBSD를 제외한 어떤 운영체제에서도 읽을 수 없다. CD를 마운트하거나 다른 운영체제와 데이터를 공유하려면 위에서 설명한대로 sysutils/mkisofs을 사용해야 된다.

16.6.9 ATAPI/CAM 드라이버 사용

이 드라이버는 SCSI 서브시스템을 통해 ATAPI 장치에(CD-ROM, CD-RW, DVD 장치 등) 접근하도록 해서 sysutils/cdrdao 또는 cdrecord(1)같은 어플리케이션을 사용할 수 있도록 한다.

이 드라이버를 사용하려면 커널 설정파일에 다음 라인을 추가해야 된다:

```
device atapicam
device scbus
device cd
device pass
```

또한 다음 라인도 커널 설정파일에 필요하다:

```
device ata
```

그리고 다시 빌드하고 새로운 커널을 설치한 후 머신을 재 부팅한다. 부팅할 때 레코더는 다음과 같이 나타난다:

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at ata1-master PIO4
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

이 드라이브는 이제 /dev/cd0 장치 이름으로 접근할 수 있고, 예를 들어 CD-ROM 을 /mnt 에 마운트하려면 다음 명령을 입력한다:

```
# mount -t cd9660 /dev/cd0 /mnt
```

root 에서 레코더의 SCSI 주소를 알기 위해 다음 명령을 실행할 수 있다:

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00>     at scbus1 target 0 lun 0 (pass0,cd0)
```

그래서 *1,0,0* 이 cdrecord(1)과 다른 SCSI 어플리케이션이 사용하는 주소가 된다.

ATAPI/CAM 과 SCSI 시스템에 대한 더 많은 정보는 atapicam(4)와 cam(4) 매뉴얼 페이지를 참고한다.

16.7 광학 미디어(DVD) 생성과 사용

16.7.1 소개

CD 와 비교하여 DVD 는 차세대 광학 미디어 스토리지 기술이다. DVD 는 CD 보다 많은 데이터를 저장하고 요즘 비디오 배포의 표준이다.

물리적으로 레코드 할 수 있는 5 가지 포맷을 레코드하는 DVD 에 정의할 수 있다:

- DVD-R: 이것은 처음으로 DVD를 레코드 할 수 있던 포맷이다. DVD-R 표준은 DVD 포럼(http://www.dvdforum.com/forum.shtml)에 정의되어 있다. 이 포맷은 한 번만 레코드 한다.

- DVD-RW: 이것은 DVD-R 표준의 다시 레코드 할 수 있는(rewriteable) 버전이다. DVD-RW 는 1000 회 정도 다시 레코드 할 수 있다.

- DVD-RAM: 이것도 DVD 포럼이 지원하는 다시 레코드 할 수 있는 포맷이다. DVD-RAM 은 이동할 수 있는 하드 드라이브처럼 보인다. 그러나 이 미디어는 대부분의 DVD-ROM 드라이브와 DVD 비디오 플레이어와 호환되지 않는다; 오직 몇 개의 DVD 레코더만 DVD-RAM 을 지원한다.

- DVD+RW: 이것은 DVD+RW Alliance(http://www.dvdrw.com/)가 정의한 다시 레코드 할 수 있는 포맷이다. DVD+RW는 대략 1000 회 정도 다시 레코드 할 수 있다.

- DVD+R: 이 포맷은 다양한 DVD+RW 포맷을 한번만 레코드 한다.

한 장의 DVD 는 실제 4.38GB 또는 4485MB 에(1 kilobyte 는 1024bytes) 달하는 4,700,000,000 바이트를 저장할 수 있다.

> Note: 물리적인 미디어와 어플리케이션에 따라 차이가 있다. 예를 들어 DVD 비디오는 레코딩하는 물리적인 미디어에 작성할 수 있는 특정 파일 레이아웃이다: DVD-R, DVD+R, DVD-RW 등. 미디어 종류를 선택하기 전에 레코딩 소프트웨어와 DVD 비디오 플레이어가(표준 플레이어나 컴퓨터의 DVD-ROM) 미디어와 호환되는지 확인해야 된다.

16.7.2 설정

프로그램 growisofs(1)은 DVD 레코딩에 사용된다. 이 명령은 dvd+rw-tools 유틸리티의 (sysutils/dvd+rw-tools) 일부분이다. dvd+rw-tools는 모든 DVD 미디어 종류를 지원한다

이들 툴은 장치에 접근하기 위해 SCSI 서브시스템을 사용하므로 커널에 ATAPI/CAM 지원을 추가해야 된다.

또한 ATAPI 장치를 사용하기 위해 DMA 도 활성화해야 된다. 다음 라인을 /boot/loader.conf 파일에 추가하면 된다:

```
hw.ata.atapi_dma="1"
```

dvd+rw-tools 를 사용하기 전에 여러분의 DVD 레코더와 관련된 정보를 얻기 위해 dvd+rw-tools 의 하드웨어 호환 노트를 참고한다.

16.7.3 데이터 DVD 굽기

growisofs(1) 명령은 파일시스템 레이아웃을 만들어서 DVD 에 작성하는 mkisofs(8)을 실행한다. 이 의미는 DVD 를 굽기 전에 데이터 이미지를 만들 필요가 없다는 것이다.

/path/to/data 디렉터리의 데이터를 DVD+R 이나 DVD-R 로 굽기 위해 다음 명령을 사용한다:

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

옵션 *-J -R*은 파일시스템을 생성하기 위해(이 예제에서는 Joliet 와 Rock Ridge 로 확장하여 ISO 9660 파일시스템을 생성) mkisofs(8)에 적용한다. 더 자세한 내용은 mkisofs(8) 매뉴얼 페이지를 참고한다.

옵션 *-Z*는 멀티 세션이던지 아니던지 상관없이 초기 세션을 레코딩 할 때 사용한다. DVD 장치 */dev/cd0*는 여러분의 설정에 따라 변경해야 된다. *-dvd-compat* 매개변수는 디스크를 닫아서 추가적으로 레코딩 할 수 없다. 대신 이것은 더 많은 DVD-ROM 드라이브와 미디어 호환성을 제공한다.

이미 생성한 이미지를 레코드 할 수도 있다. 예를 들면 *imagefile.iso* 이미지를 굽기 위해 다음 명령을 실행한다:

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

쓰기 속도는 미디어에 따라 자동으로 감지되어 설정된다. 쓰기 속도를 강제로 조정하려면 *-speed=* 매개변수를 사용한다. 더 많은 정보는 growisofs(1) 매뉴얼 페이지를 읽는다.

16.7.4 DVD 비디오 굽기

DVD 비디오는 ISO 9660 과 micro-UDF(M-UDF) 기술서를 기반으로 하는 특정 파일 레이아웃이다. 또한 DVD 비디오에는 특정 데이터 구조 계층이 있다. 이것이 DVD를 작성하기 위해 sysutils/dvdauthor같은 특정 프로그램을 사용하는 이유다.

DVD 비디오 파일시스템의 이미지를 가지고 있다면 이전 섹션의 예제에서 다른 이미지처럼 굽는다. 예를 들어 DVD 이미지를 만들었고 그 결과가 /path/to/video 디렉터리에 있다면 DVD 비디오로 굽기 위해 다음 명령을 사용한다:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

-dvd-video 옵션은 mkisofs(8)에 적용되어 DVD 비디오 파일시스템 레이아웃을 생성한다. –dvd-video 옵션은 growisofs(1) –dvd-compat 옵션과 동일하다.

16.7.5 DVD+RW 사용하기

CD-RW 와 달리 DVD+RW 는 처음 사용하기 전에 포맷을 해야 된다. growisofs(1) 프로그램이 자동으로 적절하게 포맷하기 때문에 권장하는 방법이다. 그러나 DVD+RW 를 포맷하기 위해 dvd+rw-format 명령을 사용할 수 있다:

```
# dvd+rw-format /dev/cd0
```

이 옵션을 딱 한번만 실행하면 되기 때문에 최초의 DVD+RW 미디어에만 포맷이 필요하다. 그리고 이전 섹션에서 보여준 것처럼 DVD+RW 를 굽는다.

새로운 데이터를 DVD+RW 에 굽기 위해 미디어의 내용을 삭제할 필요 없이 다음과 같이 이전 레코딩을 덮어쓰면 된다:

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

DVD+RW 포맷은 이전 레코딩에 데이터를 쉽게 추가할 수 있다. 멀티 세션으로 레코딩하지 않고 기존 데이터에 새로운 세션을 합한다. growisofs(1)은 미디어에 있던 ISO 9660 파일시스템을 확장한다.

예를 들어 이전 DVD+RW 에 데이터를 추가하려면 다음 명령을 사용해야 된다:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

초기 세션을 레코딩 할 때 사용했던 mkisofs(8) 옵션을 다음에 레코딩 할 때도 사용해야 된다.

> Note: DVD-ROM 미디어가 더 많은 드라이브와 호환되기를 원한다면 *-dvd-compat* 옵션을 사용할 수 있다. DVD+RW 에서는 이 옵션이 있어도 데이터를 추가할 수 있다.

특별한 이유로 미디어의 데이터를 삭제하려면 다음 명령을 사용한다:

```
# growisofs -Z /dev/cd0=/dev/zero
```

16.7.6 DVD-RW 사용

DVD-RW 는 두 가지 디스크 포맷을 허용한다: 연속적으로 증가하는 것과 덮어쓰기를 제한한 것. 기본적으로 DVD-RW 디스크는 연속적인(추가할 수 있는) 포맷이다.

최초의 DVD-RW 는 포맷 없이 바로 사용할 수 있지만 연속 포맷으로 사용한 DVD-RW 는 새로운 세션을 작성하기 전에 데이터를 삭제해야 된다.

연속 모드에서 DVD-RW 데이터를 삭제하기 위해 다음 명령을 실행한다:

```
# dvd+rw-format -blank=full /dev/cd0
```

> Note: 1x 미디어에서 전체를 삭제하는데(*-blank=full*) 대략 한 시간 정도 걸린다. DVD-RW 가 Disk-At-Once(DAO) 모드에서 작성됐다면 *-blank* 옵션으로 빠르게 삭제할 수 있다. DAO 모드에서 DVD-RW 를 구우려면 다음 명령을 사용한다:
>
> ```
> # growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
> ```
>
> growisofs(1)이 최소(빠른 삭제) 미디어로 감지하여 DAO 에서 작성하기 때문에 *-use-the-force-luke=dao* 옵션은 필요 없다.
>
> 이 포맷이 기본값인 연속적으로 증가하는 것보다 더 유연하기 때문에 어떤 DVD-

RW 든 덮어 쓰기 제한 모드로 사용하는 것이 좋다.

연속 DVD-RW 에 데이터를 작성하려면 다른 DVD 포맷과 같은 명령을 사용한다:

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

이전에 레코딩 한 데이터에 다른 데이터를 추가하려면 growisofs(1)에 *-M* 옵션을 사용한다. 그러나 연속 증가모드에서 DVD-RW 에 데이터를 추가한다면 새로운 세션이 디스크에 생성되어 멀티 세션 디스크가 된다.

덮어쓰기 제한 포맷에서 DVD-RW 는 새로운 세션을 생성하기 전에 데이터를 삭제할 필요 없이 DVD+RW 의 경우와 비슷하게 *-Z* 옵션으로 디스크를 덮어쓴다. *-M* 옵션을 DVD+RW 에 사용한 것처럼 디스크에 있는 이전 ISO 9660 파일시스템에 추가할 수 있다. 결과는 단일 세션 DVD 가 된다.

DVD-RW 를 덮어쓰기 제한으로 포맷하려면 다음 명령을 사용한다:

```
# dvd+rw-format /dev/cd0
```

이전의 연속 포맷으로 변환하려면 다음 명령을 사용한다:

```
# dvd+rw-format -blank=full /dev/cd0
```

16.7.7 멀티 세션

매우 적은 DVD-ROM 과 DVD 비디오 플레이어만 멀티세션 DVD 를 지원한다. 대부분 이들은 첫 번째 세션만 읽는다. 연속 포맷에서 DVD+R, DVD-R 과 DVD-RW 는 멀티 세션을 허용한다. 덮어쓰기 제한 포맷에서는 DVD+RW 와 DVD-RW 에 멀티 세션이라는 개념이 없다.

연속 포맷의 DVD+R, DVD-R 또는 DVD-RW 에서 초기화 세션 이후에 다음 명령을 사용하면 디스크에 새로운 세션을 추가한다:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

덮어쓰기 제한 모드에서 DVD+RW 또는 DVD-RW 에 이 명령 라인을 사용하면 이전 데이터

에 새로운 세션을 합병하여 데이터를 추가한다. 이 결과는 단일 세션 디스크가 된다. 이것이 이들 미디어를 작성한 후 데이터를 추가하는 방법이다.

> Note: 미디어의 어떤 공간이 각 세션의 처음과 끝에 사용된다. 따라서 미디어 공간을 최적화하기 위해 대형 데이터를 단일 세션으로 추가한다. DVD-R 에서 세션 개수는 대략 200 개 그리고 DVD+R 에서는 154 개 제한이 있다.

16.7.8 더 많은 정보

DVD 에 대한 더 많은 정보는 dvd+rw-mediainfo 를 확인한다.

dvd+rw-tools 에 대한 더 많은 정보는 growisofs(1) 매뉴얼 페이지와 dvd+rw-tools 웹 사이트 그리고 cdwrite 메일링 리스트 모음에서 찾을 수 있다.

> Note: 레코딩 결과나 미디어 문제에 대한 dvd+rw-mediainfo 출력은 어떠한 문제 보고서에도 필수다. 이 결과 없이 여러분을 돕는 것은 거의 불가능하다.

16.8 플로피 디스크 생성과 사용

예를 들어 누군가 이동할 수 있는 스토리지 미디어가 없을 때 다른 컴퓨터로 작은 양의 데이터를 전송해야 된다면 플로피 디스크에 데이터를 저장하는 것도 유용한다.

이 섹션은 FreeBSD 에서 플로피를 어떻게 사용하는지 설명한다. 여기서는 3.5 인치 DOS 플로피 포맷을 사용하는 방법에 대해 주로 다루지만 개념은 다른 플로피 디스크 포맷과 비슷하다.

16.8.1 플로피 포맷

16.8.1.1 장치

플로피도 다른 장치처럼 /dev 엔트리로 접근한다. FreeBSD 4.X 와 이전 릴리즈에서 raw 플

로피 디스크를 사용하려면 /dev/fd*N* 이나 /dev/fd*NX* 를 사용한다. *N* 은 보통 0 인 드라이브 번호고 *X* 는 문자를 표시한다.

5.0 과 새로운 릴리즈에는 간단히 /dev/fd*N* 을 사용한다.

5.0 이전 릴리즈와 5.0 과 이후 릴리즈의 디스크 크기 설정

① 4.X 와 이전 릴리즈에서 디스크 크기

/dev/fd*N.size* 장치가 있다. *size* 는 킬로 바이트의 플로피 디스크 크기다. 이런 엔트리는 디스크 크기를 결정하기 위해 low 레벨로 포맷할 때 사용된다. 1440KB 는 다음 예제에서 사용할 크기다.

가끔 /dev 아래의 엔트리를 생성해야 된다. 생성하려면 다음 명령을 실행한다:

```
# cd /dev && ./MAKEDEV "fd*"
```

② 5.0 또는 새로운 릴리즈에서 디스크 크기

5.0 에서는 devfs(5)가 /dev 에서 자동으로 장치 노드를 관리하기 때문에 MAKEDEV 는 필요 없다.

원하는 디스크 크기는 fdformat(1)의 *-f* 플래그로 설정한다. 지원되는 크기는 fdcontrol(8)에 나열되어 있지만 가장 정확히 작동하는 1440kB 를 권장한다.

16.8.1.2 포맷

플로피 디스크는 사용하기 전에 low 레벨 포맷을 해야 된다. 보통 벤더에서 미리 포맷을 하지만 포맷은 미디어의 무결성을 체크하는 좋은 방법이다. 큰 디스크 크기를 지정하여 강제로 포맷할 수 있지만 대부분의 디스크가 1440kB 용으로 디자인되어있다.

플로피를 low-레벨 포맷하려면 fdformat(1)를 사용해야 된다. 이 유틸리티는 장치 이름을 인자로 인식한다.

에러 메시지가 발생했다면 적어두고 이로서 디스크가 좋은지 나쁜지 결정하는데 도움이 된다.

5.0 이전 릴리즈와 5.0 과 이후 릴리즈의 디스크 포맷하기

① 4.X 와 이전 릴리즈에서 포맷

플로피를 포맷 하려면 /dev/fd*N.size* 를 사용한다. 드라이브에 3.5 인치 디스크를 넣고 다음 명령을 실행한다:

```
# /usr/sbin/fdformat /dev/fd0.1440
```

② 5.0 과 새로운 릴리즈에서 포맷

플로피를 포맷하기 위해 /dev/fd*N* 장치를 사용한다. 3.5 인치 플로피 디스크를 드라이브에 넣고 다음 명령을 실행한다:

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

16.8.2 디스크 라벨

디스크를 low-레벨 포맷하고 디스크 라벨을 할당해야 된다. 이 디스크 라벨은 이후에 삭제되지만 시스템이 디스크 크기와 구조를 결정하기 위해 필요하다.

새로운 디스크 라벨이 전체 디스크에 할당되어 플로피 디스크의 구조에 대해 필요한 정보를 얻게 된다. 디스크 라벨의 구조 값은 /etc/disktab 에 나열되어 있다.

이제 다음과 같이 disklable(8)을 실행한다:

```
# /sbin/disklabel -B -r -w /dev/fd0 fd1440
```

16.8.3 파일시스템

이제 플로피에 high-레벨 포맷을 하기 위한 준비가 되었다. FreeBSD 가 디스크를 읽고 쓸 수 있도록 새로운 파일시스템을 생성한다. 새로운 파일시스템을 생성한 후 디스크 라벨이 삭제되기 때문에 디스크를 다시 포맷하려면 디스크 라벨을 다시 생성해야 된다.

플로피의 파일시스템은 UFS 나 FAT 로 만들 수 있다. 보통 FAT 가 플로피에 적절하다.

플로피에 새로운 파일시스템을 작성하려면 다음 명령을 실행한다:

```
# /sbin/newfs_msdos /dev/fd0
```

이제 디스크를 사용할 준비가 되었다.

16.8.4 플로피 사용

플로피를 사용하려면 mount_msdos(8)(4.X와 이전 릴리즈에서) 또는 mount_msdosfs(8)로(5.0 과 새로운 릴리즈에서) 마운트한다. 포트 컬렉션에서 emulators/mtools도 사용할 수 있다.

16.9 데이터 테잎 생성과 사용

주로 사용하는 테잎 미디어는 4mm, 8mm, QIC, 미니 카트리지와 DLT 다.

16.9.1 4mm (DDS: 디지털 데이터 스토리지)

4mm 테잎은 워크스테이션의 백업 미디어에서 QIC 를 대체하고 있다. 이런 경향은 QIC 드라이브의 생산을 주도하는 Conner purchased Archive 사가 QIC 드라이브 생산을 중단했을 때 더욱 가속화 되었다. 4mm 드라이브는 작고 조용하지만 안정성에 대한 평판이 좋지 않아 8mm 드라이브를 즐겨 사용한다. 이 카트리지는 8mm 카트리지 보다 더 싸고 작다(3 x 2 x 0.5 인치, 76 x 51 x 12 mm). 8mm 와 비슷한 4mm 는 둘 다 나선형 스캔 방식을 사용하기 때문에 헤드 수명이 비교적 짧다.

IMATION DDS 3 4mm DAT TAPE

이런 드라이브에서 데이터 처리량은 ~150 kB/s 에서 최고 500kB/s 에 이른다. 데이터 저장 용량은 1.3GB 에서 2.0GB 다. 이들 드라이브 대부분에서 사용할 수 있는 하드웨어 압축은 대략 두 배 용량이다. 멀티 드라이브 테잎 라이브러리 유닛은 케비넷 하나에 드라이브 6 개를 가지고 자동으로 테잎을 교체한다. 라이브러리 저장 용량은 40GB 에 달한다.

DDS-3 표준은 이제 12GB 이상의(또는 압축해서 24GB) 용량을 지원한다.

테잎은 2,000 회가 지나거나 100 번의 전체 백업에 사용하면 사용할 수 없다.

16.9.2 8mm (Exabyte)

8mm 테잎은 보편적인 SCSI 테잎 드라이브로 테잎을 교체하는 방식의 최고의 선택이 되고 있다. 거의 모든 사이트는 2GB 8mm 테잎 드라이브를 가지고 있다. 8mm 드라이브는 안정적이고 상당히 편리하지만 카트리지가 비싸고 작다(4.8 x 3.3 x 0.6 인치; 122 x 84x 15mm). 8mm 테잎의 단점은 헤드를 스치는 테잎의 많은 움직임으로 헤드와 테잎의 수명이 비교적 짧다.

Verbatim 8mm DAT TAPE

데이터를 처리하는 범위는 ~250kB/s 에서 ~500 kB/s 다. 데이터 크기는 300MB 에서 7GB 에 달한다. 대부분의 이들 드라이브에서 사용할 수 있는 하드웨어 압축은 대략 두 배 용량이다. 이들 드라이브는 싱글 유닛 또는 드라이브 6 개로 된 멀티 드라이브 테잎 라이브러리와 120 개의 테잎을 싱글 케비넷에 사용할 수 있다. 테잎은 유닛에 의해 자동으로 교체되고 라이브러리 용량은 840+ GB 에 달한다.

대형 "맘모스" 모델은 하나의 테잎에(압축해서 24GB) 12GB 를 지원하고 가격은 기존 테잎 드라이브의 대략 두 배 정도다.

데이터는 나선 스캔으로 테잎에 저장되고 헤드는 미디어의(대략 6 도 정도) 각진 부분에 위치한다. 테잎은 대략 헤드를 잡고 있는 스풀의 270 도 각도로 감겨 있다. 스풀은 테잎이 스풀 위를 미끄러지는 동안 감긴다. 그 결과 데이터의 밀도는 높고 한쪽 끝에서 다른 쪽으로 테잎을 교차하여 빡빡하게 트랙에 저장된다.

16.9.3 QIC

QIC-150 테잎과 드라이브는 아주 보편적인 테잎 드라이브와 미디어일 것이다. QIC 테잎 드라이브는 가장 싼 가격의 백업 드라이브 시리즈다. 단점은 미디어의 가격이다. QIC 테잎은 8mm 나 4mm 테잎에 비해 비싸고 GB 데이터 스토리지당 5 배 정도 비싸다. 그러나 6 개정도의 테잎으로 만족한다면 QIC 가 정확한 선택일 것이다. QIC 는 아주 보편적인 테잎 드라이브기 때문에 모든 사이트는 QIC 드라이브를 다른 장비만큼 가지고 있다. QIC 는 물리적으로 테잎과 비슷한(종종 똑 같은) 큰 밀도를 가진다. QIC 드라이브는 조용하지 않고 데이터를 레코드하기 전과 읽기, 쓰기 또는 검색할 때는 시끄럽다. QIC 테잎 치수는 6 x 4 x 0.7 인치; 15.2 x 10.2 x 1.7 mm 다. 테잎 라이브러리와 교환기는 사용할 수 없다.

Verbatim QIC TAPE

데이터 처리량은 ~150kB/s 에서 ~500 kB/s 이고 용량은 40MB 에서 15GB 다. 하드웨어 압축은 새로운 QIC 드라이브에서 사용할 수 있으며 점점 사용하는 사이트가 적어지고 있다; 이들은 DAT 드라이브로 교체되고 있다.

데이터는 테잎의 트랙에 저장되고 트랙은 테잎 미디어의 긴 축을 따라 끝에서 끝으로 동작한다. 트랙은 테잎의 용량에 따라 넓이가 다양하다. 대부분의 새로운 드라이브는 예전 것과 읽기(종종 쓰기도) 호환을 제공한다. QIC 는 데이터 보존에 대한(이 메커니즘은 나선 스캔 드라이브 보다 더 단순하고 강하다) 좋은 평판을 가지고 있다.

테잎은 5,000 번 백업 이후에 폐기된다.

16.9.4 DLT

DLT 는 여기에 나열된 모든 드라이브 중 가장 빠른 데이터 전송율을 가지고 있다. 1/2"(12.5mm) 테잎은 싱글 스풀 카트리지에(4 x 4 x 1 인치; 100 x 100 x 25mm) 포함되어 있다. 카트리지는 카트리지 입구에 흔들리는 뚜껑을 하나 가지고 있다. 이 드라이브 메커니즘은 테잎 리더를 추출하기 위해 이 뚜껑을 연다. 테잎 리더는 테잎을 끌어오기 위해 사용하는 타원형의 구멍를 가지고 있으며 스풀을 팽팽히 당겨서 테잎 드라이브로 들어간다. 여기에 나열된(9 트랙 테잎 만 제외하고) 모든 다른 테잎 카트리지는 테잎 카트리지 내부에서 스풀을 밀고 당긴다.

MAXELL 40GB DLT TAPE

데이터 전송량은 대략 1.5MB/s 이어서 4mm, 8mm 나 QIC 테잎 드라이브의 3 배의 전송량이 된다. 데이터 저장 용량은 싱글 드라이브에서 10GB 에서 20GB 에 달한다. 드라이브는 멀티 테잎 교환기와 멀티 테잎 라이브러리는 5 에서 900 개의 테잎을 1 에서 20 개의 드라이브에 사용할 수 있어서 50GB 에서 9TB 의 저장 용량을 제공한다.

DLT 테잎 IV 포맷으로 압축하면 70GB 이상의 용량을 지원한다.

데이터는 테잎의 트랙에 병렬로 직접(QIC 테잎처럼) 저장되어 한번에 두 개의 트랙에 레코드 되고 읽기/쓰기 헤드의 수명은 비교적 긴 편이다; 테잎의 움직임이 정지되면 헤드와 테잎 사이의 움직임도 없다.

16.9.5 AIT

AIT 는 Sony 의 새로운 포맷이고 테잎당 50GB 까지(압축해서) 저장한다. 테잎은 내용 색인을 가지고 있는 메모리 칩을 포함한다. 이 색인으로 테잎 드라이브가 파일의 위치를 빨리 찾아서 다른 테잎에서 파일 검색에 필요한 몇 분 정도를 단축할 수 있다.

SONY AIT TAPE

SAMS:Alexandria 같은 소프트웨어는 40 개 또는 더 많은 AIT 테잎 라이브러리를 관리할 수 있고 화면에 내용을 출력하기 위해 테잎의 메모리 칩과 직접 통신한다. 그리고 어떤 테잎에 어떤 파일이 백업되었는지 확인하여 정확한 테잎 위치를 찾아서 로드한 후 테잎에서 데이터를 복구할 수 있다.

16.9.6 새로운 테잎을 처음으로 사용하기

완벽한 공 테잎에 처음으로 읽기 또는 쓰기를 하면 제대로 동작하지 않는다. 콘솔 메시지는 다음과 비슷할 것이다:

```
sa0(ncr1:4:0): NOT READY asc:4,1
sa0(ncr1:4:0):   Logical unit is in process of becoming ready
```

테잎은 체크 블록을(블록 번호 0) 가지고 있지 않다. 모든 QIC 테잎 드라이브에 QIC-525 표준을 채용하여 테잎에 체크 블록을 작성한다. 여기 두 가지 방법이 있다:

- mt fsf 1 명령으로 테잎의 체크 블록을 작성한다.

- 테잎을 꺼내기 위해 앞 패널의 버튼을 사용한다.
- 다시 테잎을 넣고 dump 데이터를 테잎에 받는다.
- dump 는 “DUMP: End of tape detected” 리포트를 보여주고 콘솔에 “HARDWARE FAILURE info:280 asc:80,96” 보여준다.
- mt rewind 명령으로 테잎을 되감는다.
- 다른 테잎들도 똑같이 실행한다.

16.10 플로피에 백업하기

16.10.1 데이터를 백업하기 위해 플로피를 사용할 수 있는가?

플로피 디스크는 다음과 같은 이유로 백업용으로 적당하지 않다:

- 미디어의 안정성이 떨어지고 특히 장시간 보관에 취약하다.
- 백업과 복구가 매우 느리다.
- 용량의 한계가 크다.

그러나 데이터를 백업할 만한 다른 방법이 없다면 플로피 디스크를 이용한 백업이 하지 않는 것보다 좋다.

플로피를 사용한다면 고품질의 플로피를 사용해야 된다. 몇 년 동안 사무실에 방치하던 플로피는 좋은 선택이 아니다. 이상적인 방법은 평판이 좋은 회사의 새 제품을 이용하는 것이 좋다.

16.10.2 플로피에 데이터를 어떻게 백업하는가?

플로피에 백업하는 가장 좋은 방법은 여러 플로피에 나눠서 백업하는 *-M* 옵션으로(멀티 볼륨) tar(1)를 사용하는 것이다.

현재 디렉터리와 서브디렉터리 전체를 백업하려면 다음 명령을 사용한다(root 에서):

```
# tar Mcvf /dev/fd0 *
```

첫 번째 플로피가 꽉 차면 tar(1)는 다음 볼륨을(왜냐하면 tar(1)은 미디어와 독립적이기 때문에 이것은 볼륨을 의미한다; 이 문맥에서는 플로피 디스크를 의미한다) 요청하는 프롬프트를 보여준다.

```
Prepare volume #2 for /dev/fd0 and hit return:
```

지정한 모든 파일을 저장할 때까지 이 메시지는 계속된다(볼륨 번호가 증가하면서).

16.10.3 압축해서 백업할 수 있는가?

불행히 tar(1)는 멀티 볼륨을 저장할 때 *-z*옵션을 사용하지 못한다. 물론 모든 파일을 gzip(1)으로 압축하고 이 파일을 tar(1)로 플로피에 저장해서 gunzip(1)으로 압축을 풀 수 있다.

16.10.4 백업한 내용은 어떻게 복구하는가?

전체 데이터를 복구하려면 다음 명령을 사용한다:

```
# tar Mxvf /dev/fd0
```

지정한 파일만 복구하는 두 가지 방법이 있고 첫 번째는 첫 번째 플로피에서 시작할 수 있다:

```
# tar Mxvf /dev/fd0 filename
```

유틸리티 tar(1)는 필요한 파일을 찾을 때까지 다음 플로피를 넣으라는 프롬프트를 보여준다.

다른 방법으로 어떤 플로피의 파일인지 알고 있다면 간단히 플로피를 넣고 위와 같은 명령을 사용한다. 플로피의 첫 번째 파일이 이전 플로피에서 계속된다면 파일을 요청하지 않았어도 tar(1)는 복구하지 못한다는 메시지를 보여준다.

16.11 백업 기본

주요 백업 프로그램 3 가지는 dump(8), tar(1)와 cpio(1)다

16.11.1 덤프와 복구

전통적인 유닉스 백업 프로그램은 dump 와 restore 다. 이 프로그램들은 드라이브에서 디스크 블록과 파일 시스템이 생성한 링크 및 디렉터리를 모은다. dump 는 전체 파일시스템을 장치에 백업한다. 파일시스템의 일부분이나 하나 이상의 새로운 디렉터리 트리(Incremental Backup)만 백업할 수 없다. dump 는 파일과 디렉터리를 테잎에 넣을 수 없고 대신 파일과 디렉터리를 포함한 raw 데이터 블록을 넣을 수 있다.

> Note: root 디렉터리에서 dump 를 사용한다면 /home, /usr 이나 다른 디렉터리는 백업되지 않는다. 왜냐하면 이들 디렉터리는 보통 다른 파일시스템에 마운트되거나 심볼릭으로 링크되기 때문이다.

dump 는 이전 AT&T 유닉스 버전 6 의(1975 년경) 특징을 가지고 있다. 기본 매개변수는 9 트랙 테잎에(6250 bpi) 적당하고 요즘의(62,182 ftpi 이상의) 고밀도 미디어에서는 사용할 수 없다. 이런 기본적인 제한은 현대 테잎 드라이브의 용량을 사용하기 위해 명령어 라인에서 무시해야 된다.

다른 컴퓨터의 테잎 드라이브에 rdump 와 rrestore 로 네트워크를 통해 데이터를 백업하는 것도 가능하다. 두 가지 프로그램은 원격 테잎 드라이브를 사용하기 위해 rcmd 와 ruserok 를 사용한다. 그래서 유저가 원격 머신의 .rhosts 파일에 나열되어 있어야 백업을 수행할 수 있다. rdump 와 rrestore 인자는 원격 컴퓨터를 사용하기에 적절해야 된다. FreeBSD 컴퓨터에서 rdumping 으로 Exabyte 테잎 드라이브가 연결된 Sun 의 komodo 에 사용하려면 다음 명령을 입력한다:

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

주의: .rhosts 인증을 허가하는 것은 보안과 관련이 있다.

더욱 안전한 ssh 로 dump 와 restore 를 사용할 수 있다.

예제 16-1. ssh 를 통한 dump 사용

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh1 -c blowfish ₩
        targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

16.11.2 tar

tar(1)도 AT&T 유닉스 버전 6 부터(1975 년경) 사용되었다. Tar 명령은 파일시스템을 다룬다; 파일과 디렉터리를 테잎에 저장한다. tar 는 cpio(1)에 사용할 수 있는 완벽한 옵션을 지원하지 않지만 cpio 가 사용하는 일반적이지 않은 파이프라인 명령은 필요 없다.

tar 버전 대부분은 네트워크 백업을 지원하지 않는다. FreeBSD 가 사용할 수 있는 GNU 버전의 tar 는 rdump 와 같은 구문으로 원격 장치를 지원한다. tar 로 Exabyte 테잎 드라이브가 연결된 Sun komodo 사용하려면 다음 명령을 사용한다:

```
# /usr/bin/tar cf komodo:/dev/nsa8 . 2>&1
```

원격 장치를 지원하지 않는 버전에서 원격 테잎 드라이브에 데이터를 보내기 위해 파이프라인과 rsh 를 사용할 수 있다.

```
# tar cf - . | rsh hostname dd of=tape-device obs=20b
```

네트워크 백업의 보안이 걱정된다면 rsh 대신 ssh 를 사용한다.

16.11.3 cpio

cpio(1)는 마그네틱 미디어에 맞는 최초의 유닉스 파일 교환 테잎 프로그램이다. cpio 는 파일 교환 수행, 수많은 기록 포맷을 지원하고 데이터를 다른 프로그램에 보내는 파이프 옵션

을 가지고 있다. 마지막 기능 때문에 cpio 가 설치할 때 사용하는 최고의 미디어가 되었다. cpio 는 디렉터리 트리와 파일 리스트가 어떻게 동작하는지 모르기 때문에 표준 입력을 지원 해야 된다.

cpio 는 네트워크 백업을 지원하지 않아서 원격 테잎 드라이브에 데이터를 보내기 위해 파이프 라인과 rsh 를 사용할 수 있다.

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

directory_list 는 백업하려는 디렉터리 리스트며 *user@host* 는 백업을 수행하는 유저/호스트 이름 그리고 *backup_device* 는 백업할 위치다(예: /dev/nsa0).

16.11.4 pax

pax(1)는 tar 와 cpio 의 IEEE/POSIX 버전이다. 세월이 흐르고 tar 와 cpio 의 다양한 버전들이 약간씩 호환되지 않았다. 그래서 완벽한 표준을 제시하여 POSIX 는 새로운 기록 유틸리티를 만들었다. pax 는 다양한 cpio 와 tar 포맷을 읽고 쓸 수 있으며 자신만의 새로운 포맷을 추가했다. 이 명령어 세트는 tar 보다 cpio 와 비슷하다.

16.11.5 Amanda

Amanda 는(발전된 메릴렌드 네트워크 디스크 기록자) 한가지 프로그램 이라기보다 클라이언트/서버 백업 시스템이다. Amanda 서버는 Amanda 클라이언트로 Amanda 서버와 네트워크로 연결된 수많은 컴퓨터의 데이터를 싱글 테잎 드라이브에 저장한다. 사이트에서 대형 디스크와 발생하는 일반적인 문제는 테잎에 데이터를 직접 백업할 시간이 태스크에 할당된 시간의 양을 초과하는 것이다. Amanda 는 이 문제를 해결하였다. Amanda 는 같은 시간에 여러 파일시스템을 백업하기 위해 "holding disk"를 사용할 수 있다. 그리고 Amanda 는 "기록 세트"를 생성한다: 테잎 그룹이 Amanda 의 설정 파일에 나열된 모든 파일시스템의 Full 백업에 일정 시간을 사용한다. "기록 세트"도 밤 마다 모든 파일시스템의 incremental(또는 differential) 백업을 수행한다. 피해가 발생한 파일시스템을 복구하려면 가장 최근의 Full 백업과 incremental 백업이 필요하다.

설정 파일은 백업과 Amanda 가 만들어내는 네트워크 트래픽을 알맞게 조정한다. 또한 데이터를 테잎에 저장하기 위해 위의 백업 프로그램을 사용한다. Amanda 는 기본적으로 설치되지 않지만 포트나 패키지로 사용할 수 있다.

16.11.6 어떤 백업 프로그램이 최고인가?

dump(8)가 사용되던 시기에 Elizabeth D.Zwicky 는 여기서 설명하는 모든 백업 프로그램을 테스트하였다. 모든 데이터와 유닉스 파일시스템의 모든 속성을 보존하는 완벽한 프로그램은 dump 다. Elizabeth 는 다양하고 거대한 파일시스템으로 특별한 상황을 만들고(일반 적인 것도 포함한) 이들 파일시스템의 백업과 복구로 각 프로그램을 테스트했다. 포함된 특성은: 문제가 있는 파일, 파일 이름을 재미있는 문자로 만든 파일, 읽을 수 없고 쓰기를 못하는 파일, 장치, 백업하는 동안 변경된 파일크기, 백업하는 동안 생성/삭제된 파일 등이다. 그녀는 1991 년 9 월에 LISA V 에 결과를 올렸다. 테스트 백업과 백업프로그램을 확인해 본다 (http://berdmann.dyndns.org/zwicky/testdump.doc.html).

16.11.8. 비상시 복구 절차

16.11.8.1 재난이 발생하기 전에

발생할 만한 재난에 대비하기 위한 4 가지 단계가 있다.
첫째, 각 디스크(예: disklabel da0 | lpr)의 라벨, 파일시스템 테이블과(/etc/fstab) 모든 부트 메시지를 각각 두 개씩 출력해둔다.

둘째, 모든 장치를 가지고 있는 부팅 플로피를(boot.flp 와 fixit.flp) 준비한다. 체크하는 가장 쉬운 방법은 플로피 드라이브에 플로피를 넣고 머신을 재 부팅해서 부트 메시지를 체크한다. 모든 장치가 나열되고 동작한다면 단계 3 를 건너뛰어도 된다.

그렇지 않으면 모든 디스크를 마운트하고 테잎 장치를 사용할 수 있는 커널을 가진 두 개의 부팅 플로피를 만들어야 한다. 이들 플로피는 다음과 같은 명령도 가지고 있어야 한다: fdisk, disklabel, newfs, mount 와 사용하려는 백업 프로그램. 이들 프로그램은 정적으로 링크되어 있어야 한다. dump 를 사용한다면 플로피는 restore 도 가지고 있어야 된다.

셋째, 일반적인 백업 테잎을 만든다. 마지막 백업을 수행한 후 변경된 것은 복구할 수 없을

것이다. 백업 테잎에 쓰기 방지 한다.

넷째, 플로피와(boot.flp 와 fixit.flp 또는 단계 2 에서 만든 두 개의 사용자 부팅 플로피 중 하나) 백업 테잎을 테스트한다. 그리고 복구 절차를 적어 둔다. 부팅 가능한 플로피, 출력한 정보와 백업 테잎을 이들 노트와 같이 보관한다. 복구할 때 너무 긴장하여 백업 테잎을 망가뜨리는 것을 적어둔 노트로 방지할 수 있다(어떻게? tar xvf /dev/sa0 대신 순간적으로 tar cvf /dev/sa0 로 백업 테잎을 덮어쓰기 할 것이다).

보안 평가를 추가하려면 부트 플로피와 백업 테입을 만들어서 원격지에 따로 저장한다. 원격지는 같은 사무실 빌딩이 아니다. 세계 무역 센터의 수많은 상업적 회사들은 비싼 값에 이런 사항을 확인하였다. 원격지는 물리적으로 컴퓨터와 디스크 드라이브가 있는 곳에서 일정 거리 이상을 두어야 한다.

예제 16-3. 부팅 플로피를 만드는 스크립트

```
#!/bin/sh
#
# create a restore floppy
#
# format the floppy
#
PATH=/bin:/sbin:/usr/sbin:/usr/bin

fdformat -q fd0
if [ $? -ne 0 ]
then
        echo "Bad floppy, please use a new one"
        exit 1
fi

# place boot blocks on the floppy
#
disklabel -w -B /dev/fd0c fd1440

#
# newfs the one and only partition
#
```

```
newfs -t 2 -u 18 -l 1 -c 40 -i 5120 -m 5 -o space /dev/fd0a

#
# mount the new floppy
#
mount /dev/fd0a /mnt

#
# create required directories
#
mkdir /mnt/dev
mkdir /mnt/bin
mkdir /mnt/sbin
mkdir /mnt/etc
mkdir /mnt/root
mkdir /mnt/mnt              # for the root partition
mkdir /mnt/tmp
mkdir /mnt/var

#
# populate the directories
#
if [ ! -x /sys/compile/MINI/kernel ]
then
        cat << EOM
The MINI kernel does not exist, please create one.
Here is an example config file:
#
# MINI -- A kernel to get FreeBSD onto a disk.
#
machine             "i386"
cpu                 "I486_CPU"
ident               MINI
maxusers            5

options             INET                    # needed for _tcp _icmpstat _ipstat
```

```
                                        #               _udpstat _tcpstat _udb
options         FFS                     #Berkeley Fast File System
options         FAT_CURSOR              #block cursor in syscons or pccons
options         SCSI_DELAY=15           #Be pessimistic about Joe SCSI device
options         NCONS=2                 #1 virtual consoles
options         USERCONFIG              #Allow user configuration with -c XXX

config          kernel  root on da0 swap on da0 and da1 dumps on da0

device          isa0
device          pci0

device          fdc0    at isa? port "IO_FD1" bio irq 6 drq 2 vector fdintr
device          fd0 at fdc0 drive 0

device          ncr0

device          scbus0

device          sc0 at isa? port "IO_KBD" tty irq 1 vector scintr
device          npx0    at isa? port "IO_NPX" irq 13 vector npxintr

device          da0
device          da1
device          da2

device          sa0

pseudo-device   loop            # required by INET
pseudo-device   gzip            # Exec gzipped a.out's
EOM
        exit 1
fi

cp -f /sys/compile/MINI/kernel /mnt
```

```
gzip -c -best /sbin/init > /mnt/sbin/init
gzip -c -best /sbin/fsck > /mnt/sbin/fsck
gzip -c -best /sbin/mount > /mnt/sbin/mount
gzip -c -best /sbin/halt > /mnt/sbin/halt
gzip -c -best /sbin/restore > /mnt/sbin/restore

gzip -c -best /bin/sh > /mnt/bin/sh
gzip -c -best /bin/sync > /mnt/bin/sync

cp /root/.profile /mnt/root

cp -f /dev/MAKEDEV /mnt/dev
chmod 755 /mnt/dev/MAKEDEV

chmod 500 /mnt/sbin/init
chmod 555 /mnt/sbin/fsck /mnt/sbin/mount /mnt/sbin/halt
chmod 555 /mnt/bin/sh /mnt/bin/sync
chmod 6555 /mnt/sbin/restore

#
# create the devices nodes
#
cd /mnt/dev
./MAKEDEV std
./MAKEDEV da0
./MAKEDEV da1
./MAKEDEV da2
./MAKEDEV sa0
./MAKEDEV pty0
cd /

#
# create minimum file system table
#
cat > /mnt/etc/fstab <<EOM
/dev/fd0a       /       ufs     rw   1   1
```

```
EOM

#
# create minimum passwd file
#
cat > /mnt/etc/passwd <<EOM
root:*:0:0:Charlie &:/root:/bin/sh
EOM

cat > /mnt/etc/master.passwd <<EOM
root::0:0::0:0:Charlie &:/root:/bin/sh
EOM

chmod 600 /mnt/etc/master.passwd
chmod 644 /mnt/etc/passwd
/usr/sbin/pwd_mkdb -d/mnt/etc /mnt/etc/master.passwd

#
# umount the floppy and inform the user
#
/sbin/umount /mnt
echo "The floppy has been unmounted and is now ready."
```

16.11.8.2 사고 발생 후

핵심 질문은 하드웨어는 이상이 없는가? 이다. 일반적인 백업을 받아 두었기 때문에 소프트웨어에 대해서는 걱정할 필요가 없다.

하드웨어가 파손 되었다면 파손된 부분을 교체한다.

하드웨어에 이상이 없다면 플로피를 체크한다. 사용자 부트 플로피를 사용하려면 싱글 유저 모드(boot: 프롬프트에서 -s 를 입력한다)로 부팅한다. 다음절을 건너뛴다.

boot.flp 와 fixit.flp 플로피를 사용한다면 계속 읽어야 된다. 플로피 드라이브에 boot.flp 플로피를 넣고 컴퓨터를 부팅한다. 화면에 최초의 설치 메뉴가 나타난다. *Fixit--Repair mode*

with CDROM or floppy 옵션을 선택하고 프롬프트가 나타나면 fixit.flp 를 넣는다. 필요한 restore 와 다른 프로그램은 /mnt2/stand 에 있다.

각 파일 시스템을 복구한다.

첫 번째 디스크의 root 파티션을 mount 한다(예: mout /dev/da0a /mnt). 디스크 라벨이 손상 되었다면 출력해서 저장한 라벨과 맞추어 디스크를 다시 파티션하고 라벨을 할당하도록 disklabel 를 사용한다. 파일시스템을 다시 생성하기 위해 newfs 을 사용한다. 플로피의 root 파티션을 읽기와 쓰기로(mount -u -o rw /mnt) 다시 마운트한다. 이 파일시스템을(예: restore vrf /dev/sa0) 복구하기 위해 백업 프로그램과 백업 테잎을 사용한다. 파일시스템을(예: umount /mnt) 언마운트 한다. 손상된 각 파일시스템 별로 반복한다.

시스템이 다시 살아나면 새로운 테잎에 데이터를 백업한다. 다시 시스템에 문제가 발생하거나 데이터 손실이 발생할 수 있다. 몇 시간을 투자하였기 때문에 문제가 발생하면 복구 시간을 단축할 수 있다.

16.12 네트워크, 메모리와 File-Backed 파일 시스템

물리적으로 컴퓨터에 있는 플로피, CD, 하드 드라이브 등을 제외하고 FreeBSD 가 알고 있는 다른 종류의 디스크는 *가상 디스크*다.

여기에는 네트워크 파일시스템과 메모리기반 파일시스템 그리고 file-backed 파일시스템이 포함된다.

사용 중인 FreeBSD 버전에 따라 file-backed 과 메모리 기반 파일시스템을 생성해서 사용하기 위해 다른 툴을 사용해야 된다.

> Note: FreeBSD 4.X 유저는 필요한 장치를 생성하기 위해 MAKEDEV(8)을 사용해야 된다. FreeBSD5.0 과 새로운 버전은 devfs(5)를 사용하여 장치 노드를 생성한다.

16.12.1 FreeBSD 4.X 에서 File-Backed 파일시스템 사용

vnconfig(8) 유틸리티는 vnode pseudo-disk 장치를 설정하고 활성화한다. *vnode*는 파일을 표현하고 파일의 활동에 초점을 맞춘다. 이 의미는 vnconfig(8)가 파일시스템을 생성하고 운용하기 위해 파일을 이용한다는 것이다. 가능한 한가지 사용 방법은 플로피를 마운트하거나 CD 이미지를 파일로 보관하는 것이다.

vnconfig(8)을 사용하려면 커널 설정파일이 vn(4)을 지원해야 된다:

```
pseudo-device vn
```

파일시스템 이미지를 마운트 하려면 다음 예제를 확인한다:

예제 16-4. FreeBSD 4.X 에서 파일시스템 이미지를 마운트하기 위해 vnconfig 사용하기

```
# vnconfig vn0 diskimage

# mount /dev/vn0c /mnt
```

vnconfig(8)로 새로운 파일시스템을 생성하려면 다음 예제를 확인한다:

예제 16-5. vnconfig 로 새로운 File-Backed 디스크 생성

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# vnconfig -s labels -c vn0 newimage
# disklabel -r -w vn0 auto
# newfs vn0c
Warning: 2048 sector(s) in last cylinder unallocated
/dev/vn0c:      10240 sectors in 3 cylinders of 1 tracks, 4096 sectors
        5.0MB in 1 cyl groups (16 c/g, 32.00MB/g, 1280 i/g)
super-block backups (for fsck -b #) at:
 32
# mount /dev/vn0c /mnt
# df /mnt
```

```
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/vn0c        4927        1     4532     0%    /mnt
```

16.12.2 FreeBSD 5.X 에서 File-Backed 파일시스템 사용

유틸리티 mdconfig(8)는 메모리 디스크를 설정해서 활성화하는데 사용하고 FreeBSD 5.X 에서는 md(4)를 대신 사용한다. mdconfig(8)을 사용하려면 md(4) 모듈을 로드하거나 지원하도록 커널 설정파일에 추가해야 된다:

```
device md
```

mdconfig(8) 명령은 3 종류의 메모리 기반 가상 디스크를 지원한다: 메모리 디스크는 malloc(9)로 할당하고 메모리 디스크도 backing 처럼 파일이나 스왑 공간을 사용한다. 가능한 사용법은 플로피를 마운트하거나 CD 이미지를 파일로 보관하는 것이다.

파일시스템을 마운트하려면 다음 예제를 확인한다:

예제 16-5. FreeBSD 5.X 에서 파일시스템 이미지를 마운트하기 위해 mdcnfig 사용하기

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0c /mnt
```

mdconfig(8)로 새로운 파일시스템 이미지를 생성하려면 다음 예제를 확인한다:

예제 16-6. mdconfig 로 새로운 File-Backed 디스크 생성하기

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f newimage -u 0
# disklabel -r -w md0 auto
# newfs md0c
/dev/md0c: 5.0MB (10240 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.27MB, 81 blks, 256 inodes.
```

```
super-block backups (for fsck -b #) at:
 32, 2624, 5216, 7808
# mount /dev/md0c /mnt
# df /mnt
Filesystem   1K-blocks      Used    Avail Capacity  Mounted on
/dev/md0c         4846         2     4458     0%    /mnt
```

-u 옵션으로 유닛 번호를 지정하지 않았다면 mdconfig(8)은 사용하지 않는 장치를 선택하기 위해 md(4) 자동할당을 사용한다. 할당된 유닛 이름은 md4 처럼 표준 출력으로 출력된다. mdocnfig(8)의 더 자세한 사항은 매뉴얼 페이지를 참고한다.

> **Note:** FreeBSD 5.1-RELEASE 부터 bsdlabel(8) 유틸리티가 예전 disklabel(8) 프로그램을 대체한다. bsdlable(8) 때문에 사용하지 않는 옵션과 매개변수가 없어졌다; 위 예제의 옵션 *-r*은 삭제해야 된다. 더 많은 정보는 bsdlabel(8) 매뉴얼 페이지를 참고한다.

mdconfig(8) 유틸리티는 매우 유용하지만 file-backed 파일시스템을 생성하기 위해 수많은 명령어 라인을 문의한다. 또한 FreeBSD 5.0 은 mdmfs(8)라는 툴을 가지고 있다. 이 프로그램은 디스크가 mdconfig(8)와 newfs(8)을 사용하여 UFS 파일시스템을 생성하고 mount(8)로 마운트하도록 md(4)를 설정한다. 예를 들어 위와 같은 파일시스템 이미지를 생성하고 마운트한다면 다음과 같이 입력한다:

예제 16-8. mdmfs 로 File-Backed 디스크 설정해서 마운트하기

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records in
5120+0 records out
# mdmfs -F newimage -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0         4846    2  4458     0%    /mnt
```

유닛 번호 없이 *md* 옵션을 사용했다면 mdmfs(8)는 사용하지 않는 장치를 자동으로 선택하기 위해 md(4) 자동-유닛 기능을 사용한다. mdmfs(8)에 대한 더 자세한 사항은 매뉴얼 페이지를 참고한다.

16.12.3 FreeBSD 4.X 에서 메모리 기반 파일시스템 사용

md(4) 드라이버는 간단해서 FreeBSD 4.X 에 메모리 기반 파일시스템을 생성하는데 효과적이다. malloc(9)는 메모리를 할당하는데 사용된다.

예를 들어 단순히 vnconfig(8)로 준비한 파일시스템을 사용할 수 있다:

예제 16 -9. FreeBSD 4.X 에서 md 메모리 디스크 사용하기

```
# dd if=newimage of=/dev/md0
5120+0 records in
5120+0 records out
# mount /dev/md0c /mnt
# df /mnt
Filesystem   1K-blocks      Used    Avail Capacity  Mounted on
/dev/md0c         4927         1     4532     0%    /mnt
```

더 자세한 사항은 md(4) 매뉴얼 페이지를 참고한다.

16.12.4 FreeBSD 5.X 에서 메모리 기반 파일시스템 사용

메모리 기반과 file-backed 파일시스템은 같은 툴을 사용한다: mdocnfig(8)나 mdmfs(8). 메모리 기반 파일시스템을 위한 메모리는 malloc(9)로 할당된다.

예제 16-10. mdconfig 로 새로운 메모리 기반 디스크 생성하기

```
# mdconfig -a -t malloc -s 5m -u 1
# newfs -U md1
/dev/md1: 5.0MB (10240 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.27MB, 81 blks, 256 inodes.
        with soft updates
super-block backups (for fsck -b #) at:
 32, 2624, 5216, 7808
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
```

```
/dev/md1          4846     2   4458       0%      /mnt
```

예제 16-11. mdmfs 로 새로운 메모리 기반 디스크 생성하기

```
# mdmfs -M -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md2          4846     2   4458       0%      /mnt
```

mdconfig(8)에서 *malloc*을 *swap*으로 바꾸면 malloc(9)이 backed 파일시스템을 사용하지 않고 스왑을 사용하는 것도 가능하다. mdmfs(8) 유틸리는 기본적으로(*-M*없이) 스왑 기반 디스크를 생성한다. 더 자세한 사항은 mdconfig(8)와 mdmfs(8) 매뉴얼 페이지를 참고한다.

16.12.5 시스템에서 메모리 디스크 삭제

메모리 기반 또는 파일 기반 파일시스템을 사용하지 않을 때 모든 리소스를 시스템 되돌려 줘야 한다. 리소스를 되돌려 주려면 첫째로 파일시스템을 언마운트하고 시스템에서 디스크를 떼어내기 위해 mdconfig(8)을 사용하여 리소스를 되돌려 준다.

예를 들어 모든 리소스를 되돌리기 위해 /dev/md4 를 사용한다:

```
# mdconfig -d -u 4
```

명령어 라인 mdconfig –l 로 md(4) 장치 설정에 대한 정보를 나열하는 것도 가능하다.

FreeBSD 4.X 에서 vnconfig(8)은 장치를 삭제하는데 사용한다. 예를 들어 /dev/vn4 가 사용한 리소스를 되돌리는 방법은 다음 명령을 사용한다:

```
# vnconfig -u vn4
```

16.12 파일시스템 스넵샷

FreeBSD 5.0 은 소프트업데이트와 결합된 새로운 기능을 제공한다: 파일시스템 스냅샷

유저는 지정한 파일시스템의 이미지를 스냅샷으로 생성해서 이미지를 파일처럼 다룰 수 있다. 스냅샷 파일은 운용중인 파일시스템에서 생성해야 되고 유저는 파일시스템 당 20 개 이하의 스냅샷만 생성할 수 있을 것이다. 운용중인 스냅샷은 슈퍼 블록에 저장되기 때문에 시스템이 재 부팅할 때 언 마운트 했다가 다시 마운트할 때 사라지지 않는다. 스냅샷이 더 이상 필요 없다면 표준 rm(1) 명령으로 삭제할 수 있다. 스냅샷은 이 명령으로 삭제되지만 사용되던 공간은 다른 스냅샷이 이런 블록 중 몇 개를 요청할 것이기 때문에 반환되지 않는다.

root 도 스냅샷에 쓰기를 못하도록 최초에 생성될 때 *schg* 플래그가(chflage(1) 매뉴얼 페이지를 본다) 설정된다. unlink(1) 명령은 *schg* 플래그 설정으로 스냅샷을 삭제할 수 있도록 스냅샷 파일에 예외를 만들기 때문에 스냅샷 파일을 삭제하기 전에 *schg* 플래그를 제거할 필요가 없다.

스냅샷은 mount(8) 명령으로 생성된다. /var 스냅샷을 /var/snapshot/snap 파일에 두려면 다음 명령을 따른다:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

다른 방법은 스냅샷을 생성하기 위해 mksnap_ffs(8)를 사용할 수 있다:

```
# mksnap_ffs /var /var/snapshot/snap
```

스냅샷이 생성되면 몇 가지 유용한 사용법이 있다:

- 스냅샷을 CD 나 테잎에 전송할 수 있기 때문에 어떤 관리자는 백업 목적으로 스냅샷 파일을 사용한다.

- fsck(8)은 스냅샷에서 동작 할 것이다. 파일시스템이 마운트 됐을 때 파일시스템은 깨끗하다고 보기 때문에 항상 깨끗한(변하지 않는) 결과를 유지할 수 있다. 이것이 근본적으로 백그라운드에서 fsck(8) 프로세스가 작동하게 된 이유다.

- 스냅샷에서 dump(8) 유틸리티 실행. dump 는 파일시스템과 스냅샷 타임스탬프의 일관성을 유지한다. dump(8)도 명령어 라인에 *-L* 플래그를 사용하여 dump 이미지

를 생성하고 스냅샷을 삭제할 수 있다.

- mount(8)로 정지된 파일시스템의 이미지처럼 스냅샷을 마운트한다. /var/snapshot/snap 에 스냅샷을 mount(8)하기 위해 다음 명령을 실행한다:

```
# mdconfig -a -t vnode -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

이제 /mnt 에 마운트 되어있는 정지된 /var 파일시스템의 계층을 둘러볼 수 있다. 스냅샷을 생성하던 시기와 모든 것이 같은 상태다. 유일한 예외는 이전 스냅샷이 크기가 0 인 파일로 나타나는 것이다. 한계가 있는 스냅샷을 사용할 때 다음 명령으로 언 마운트할 수 있다:

```
# umount /mnt

# mdconfig -d -u 4
```

기술적인 문서를 포함하여 소프트업데이트와 파일시스템 스냅샷에 대한 더 많은 정보는 Marshall Kirk McKusick 의 웹사이트를 방문한다 http://www.mckusick.com.

16.14 파일시스템 쿼타

쿼타는 파일시스템 기반에서 유저나 그룹 맴버가 사용할 수 있는 디스크 공간과 파일의 개수를 제한할 수 있는 운영체제의 부가적인 기능이다. 이 기능은 유저나 유저 그룹에게 할당되는 리소스의 양을 제한할 필요가 있는 대부분의 시분할 시스템에서 자주 사용된다. 쿼타는 유저나 유저 그룹이 사용할 수 있는 디스크 공간이 모두 소모되는 것을 방지한다.

16.14.1 디스크 쿼타를 활성화하도록 시스템 설정하기

디스크 쿼타를 적용하기 전에 쿼타가 커널에 설정되어 있어야 된다. 이것은 다음 라인을 커널 설정파일에 추가하면 된다:

```
options QUOTA
```

표준 GENERIC 커널에서는 기본적으로 활성화되지 않기 때문에 디스크 쿼타를 사용하기 위해 사용자 커널을 설정하고 빌드 후 설치해야 된다. 커널 설정에 대한 더 많은 정보는 8 장을 참고한다.

그리고 /etc/rc.conf 에서 디스크 쿼타를 활성화해야 된다. 다음 라인을 추가하면 된다:

```
enable_quotas="YES"
```

쿼타 시작과 적당한 조정을 위한 부가적인 설정 매개변수가 있다. 보통 부팅할 때 각 파일시스템의 쿼다 무결성은 quotacheck(8) 프로그램이 체크한다. quotacheck(8)은 쿼타 데이터베이스 내의 데이터가 정확히 파일시스템의 데이터와 일치하도록 한다. 이것은 시간을 많이 필요로 하는 절차여서 시스템 부팅시간에 상당한 영향을 끼친다. 이 단계를 건너뛰려면 목적에 맞게 /etc/rc.conf 의 변수를 변경해야 된다:

```
check_quotas="NO"
```

FreeBSD 3.2 릴리즈 이전 버전을 사용한다면 하나의 변수로만 이루어져 설정은 단순하다. 다음을 /etc/rc.conf 설정한다:

```
check_quotas="YES"
```

마지막으로 파일시스템 별로 디스크 쿼타를 활성화하기 위해 /etc/fstab 를 편집해야 된다. 이 설정으로 모든 파일시스템에 유저나 그룹 쿼타 또는 둘 다 적용할 수 있다.

시스템에서 유저 별로 쿼타를 적용하려면, 쿼타를 적용하려는 파일시스템의 /etc/fstab 엔트리 옵션 필드에 userquota 옵션을 추가한다. 예를 들어 다음과 같이 변경한다:

```
/dev/da1s2g     /home       ufs rw,userquota 1 2
```

그룹 쿼타를 적용하려면 비슷한 방법으로 *userquota* 대신 *groupquota* 를 사용한다. 유저와 그룹 모두에게 쿼타를 적용하려면 다음과 같이 엔트리를 변경한다:

```
/dev/da1s2g     /home       ufs rw,userquota,groupquota 1 2
```

기본적으로 쿼타 설정파일은 유저와 그룹 쿼타에 맞도록 각각 quota.user 와 quota.group 라는 이름으로 파일시스템의 root 디렉터리에 저장되어 있다. 더 많은 정보는 fstab(5)을 본다. fstab(5) 매뉴얼 페이지에서는 쿼타 파일의 위치를 지정할 수 있다고 하지만 다양한 쿼타 유틸리티가 정확한 위치를 파악하지 못하는 것 같아 권장하지는 않는다.

여기서 새로운 커널로 시스템을 재 부팅한다. /etc/rc 는 /etc/fstab 에서 활성화한 모든 쿼타의 최초 쿼타 파일을 생성하기 위해 적절한 명령을 자동으로 실행하기 때문에 크기가 0 인 쿼타 파일을 직접 만들 필요가 없다.

일반적인 운용에서는 quotacheck(8), quotaon(8)또는 quotaoff(8) 명령을 직접 실행할 필요가 없지만 사용법을 이해할 수 있도록 매뉴얼 페이지를 읽는 것도 좋은 방법이다.

16.14.2 쿼타 제한 설정

시스템에 쿼타를 활성화하도록 설정하였으므로 실제로 활성화됐는지 확인한다. 확인하는 방법은 다음 명령을 실행한다:

```
# quota -v
```

디스크 사용량과 쿼타가 활성화된 각 파일시스템의 현재 쿼타 제한을 요약한 라인을 보게 된다.

```
Disk quotas for user root (uid 0):
     Filesystem  usage  quota  limit  grace  files  quota  limit  grace
        /home 2158046      0      0             16      0      0
```

이제 edquota(8) 명령으로 쿼타 제한을 적용할 준비가 되었다.

디스크 공간을 유저나 그룹에 어떻게 할당할 것인가, 얼마나 많은 파일을 유저와 그룹이 생성할 것인가에 따라 여러 옵션이 있다. 디스크 공간이나 파일 개수 또는 두 가지를 결합하여 제한을 적용할 수 있다. 이런 제한은 두 개의 카테고리로 나누어진다: 하드와 소프트 제한.

하드 제한을 초과하지는 못한다. 유저가 하드 제한에 도달하면 이 제한이 있는 파일시스템에는 더 이상 파일을 생성하지 못한다. 예를 들어 유저가 파일시스템에 500 블록의 하드

제한을 가지고 있고 현재 490 블록을 사용 중이면 유저는 추가적으로 10 블록의 파일만 생성할 수 있다. 11 블록을 추가하려면 실패한다.

다른 방법의 소프트 제한은 일정한 시간 동안 초과할 수 있다. 이 기간은 기본적으로 일주일이며 유예기간(grace period)으로 알려져 있다. 유저가 유예기간을 어긴다면 소프트 제한은 하드 제한으로 바뀌고 더 이상 파일을 할당하지 못한다. 유저가 소프트 제한 이하로 용량을 줄이면 유예기간도 복구된다.

다음은 edquota(8) 명령을 실행했을 때 보게 될 예제를 보여준다. eduquota(8) 명령이 실행되면 쿼타 제한을 편집하도록 EDITOR 환경변수에 지정된 에디터로 들어가거나 EDITOR 변수가 설정되지 않았다면 vi 로 들어간다.

```
# edquota -u test

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

보통 쿼타가 활성화된 각 파일시스템 별로 두 라인씩 보게 된다. 하나는 블록 제한 그리고 다른 하나는 inode 제한이다. 쿼타 제한을 수정하려면 단순히 값을 변경하고 업데이트한다. 예를 들어 유저의 블록 제한을 소프트 제한 50 과 하드 제한 75 에서 소프트제한 500 과 하드 제한 600 으로 올리기 위해 다음 내용을 아래와 같이 변경한다:

```
/usr: blocks in use: 65, limits (soft = 50, hard = 75)

위의 내용을 다음과 같이 변경한다:

/usr: blocks in use: 65, limits (soft = 500, hard = 600)
```

새로운 쿼타 제한은 에디터를 끝낼 때 적용된다.

종종 UID 의 범위에 쿼타 제한을 설정하는 것도 바람직하다. *-p* 옵션을 edquota(8) 명령에 사용하면 된다. 첫째로 유저에게 원하는 쿼타 제한을 할당하고 edquota -p protouser(모델

이 되는 유저) startuid(시작되는 ID)-enduid(끝나는 ID)를 실행한다. 예를 들어 유저 test 에게 쿼타 제한을 설정하고 UID 10,000 부터 19,999 까지 똑 같이 제한할 때 다음 명령을 사용할 수 있다:

```
# edquota -p test 10000-19999
```

더 많은 정보는 edquota(8) 매뉴얼 페이지를 본다.

16.14.3 쿼타 제한과 디스크 사용량 체크

쿼타 제한과 디스크 사용량을 체크하기 위해 quota(1)나 repquota(8) 명령을 사용할 수 있다. quota(1) 명령은 각 유저나 그룹 쿼타와 디스크 사용량을 체크할 수 있다. 유저는 자신의 쿼타만 그리고 그룹 쿼타는 유저가 속한 그룹의 쿼타만 확인할 수 있다. 슈퍼 유저는 모든 유저와 그룹 쿼타를 볼 수 있다. repquota(8) 명령은 쿼타가 활성화된 파일시스템의 모든 쿼타와 디스크 사용량을 요약해서 볼 수 있다.

다음은 두 개의 파일시스템에 쿼타 제한을 가지고 있는 유저의 샘플 quota -v 명령의 결과다.

```
Disk quotas for user test (uid 1002):
     Filesystem  blocks   quota  limit   grace   files   quota  limit   grace
           /usr      65*      50     75  5days       7      50     60
       /usr/var       0      50     75              0      50     60
```

위 예제의 /usr 파일시스템에서 이 유저는 50 블록의 소프트 제한에 현재 15 블록을 초과하여 유예 기간이 5 일 남았다. 별표(*)는 유저가 현재 쿼타 제한을 초과했음을 보여준다.

유저가 전혀 사용하지 않는 일반적인 파일시스템은 이 파일시스템에 쿼타 제한이 적용되었더라도 quota(1) 명령의 출력에 아무것도 나오지 않는다. -v 옵션은 위 예제의 /usr/var 파일시스템처럼 이들 파일시스템을 보여줄 것이다.

16.14.4 NFS 에서 쿼타

쿼타 서브시스템으로 NFS 서버에 쿼타가 강제로 적용된다. rpc.rquotad(8) 데몬은 NFS 클라

이언트에서 quota(1) 명령으로 유저가 자신의 쿼타 통계를 볼 수 있도록 한다.

/etc/inetd.conf 에서 rpc.rquotad 를 활성화하려면 다음 내용을 추가한다:

```
rquotad/1        dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

이제 inetd 를 다시 시작한다:

```
# kill -HUP `cat /var/run/inetd.pid`
```

16.15 디스크 파티션 암호화

FreeBSD 는 인증 받지 않은 데이터에 접근하는 것을 막기 위해 훌륭한 온라인 보호를 제공한다. 파일 퍼미션과 Mandatory Access Control (MAC)이(15 장을 본다) 컴퓨터가 켜져 있고 운영체제가 운용 중일 때 인증 받지 않은 어플리케이션으로부터 데이터 접근을 방지한다. 그러나 공격자가 물리적으로 컴퓨터에 접근하여 간단히 컴퓨터의 하드 드라이브를 다른 시스템에 복사해서 귀중한 데이터를 분석한다면 운영체제가 적용한 퍼미션은 의미가 없다.

공격자가 하드 드라이브를 소유하거나 컴퓨터의 전원을 끄고 중요한 자원에 높은 자극을 주더라도 **GEOM 기반 디스크 암호화(gbde)**는 컴퓨터 파일시스템의 데이터를 보호할 수 있다. 몇 개의 파일에만 암호화할 수 있는 귀찮은 방법과 달리 **gbde** 는 전체 파일시스템을 완벽하게 암호화한다.

16.15.1 커널에서 **gbde** 를 활성화하기

[예제: 커널에서 gbde 를 활성화하기]

① **root** 가 된다.

gbde 설정은 슈퍼 유저 권한이 필요하다.

```
% su -
Password:
```

② 운영체제 버전을 확인한다.

gbde(4)는 FreeBSD 5.0 또는 새로운 버전이 필요하다.

```
# uname -r
5.0-RELEASE
```

③ 커널 설정파일에 gbde(4) 지원을 추가한다.

좋아하는 텍스트 에디터로 커널 설정파일에 다음 라인을 추가한다:

```
options GEOM_BDE
```

설정을 추가해서 다시 컴파일 한 후 FreeBSD 커널을 설치한다. 이 절차는 8 장에서 설명하였다. 마지막으로 새로운 커널로 재 부팅한다.

16.15.2 암호화할 하드 드라이브 준비

다음 예제는 새로운 하드 드라이브를 추가하여 파티션을 암호화한다고 가정한다. 이 파티션은 /private 에 마운트 할 것이다. **gbde** 는 /home 과 /var/mail 을 암호화 하는데도 사용할 수 있지만 이 소개서의 범위를 초과하는 더 복잡한 설명이 필요하다.

[예제: 하드 드라이브를 추가하여 파티션 암호화하기]

① 새로운 하드 드라이브 추가.

16.3 장에서 설명했듯이 새로운 드라이브를 시스템에 설치한다. 이 예제의 목적에 따라 새로운 하드 드라이브 파티션이 /dev/ad4s1c 에 추가되었다. /dev/ad0s1 * 장치는 이 예제 시스템의 표준 FreeBSD 파티션을 나타낸다.

```
# ls /dev/ad*
/dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
/dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
/dev/ad0s1a     /dev/ad0s1d     /dev/ad4
```

② **gbde 잠금(Lock) 파일을 저장할 디렉터리를 생성한다.**

```
# mkdir /etc/gbde
```

gbde 잠금 파일은 **gbde** 가 암호화 파티션에 접근하기 위해 필요한 정보를 가지고 있다. 잠금 파일이 없는 **gbde** 는 암호화된 파티션에서 소프트웨어로 지원되지 않는 엄청난 수작업 없이 암호화된 데이터를 해석하지 못한다. 암호화된 파티션들은 각자의 잠금 파일을 사용한다.

③ **gbde 파티션 초기화**

gbde 파티션을 사용하기 전에 초기화해야 된다. 초기화는 한번만 수행하면 된다:

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c
```

gbde(8)은 에디터를 열고 템플릿에서 다양한 옵션을 설정할 수 있게 한다. UFS1 이나 UFS2 를 사용하려면 sector_size 를 2048 로 설정한다:

```
$FreeBSD: src/sbin/gbde/template.txt,v 1.1 2002/10/20 11:16:13 phk Exp $
#
# Sector size is the smallest unit of data which can be read or written.
# Making it too small decreases performance and decreases available space.
# Making it too large may prevent filesystems from working.  512 is the
# minimum and always safe.  For UFS, use the fragment size
#
sector_size     =       2048
[...]
```

gbde(8)은 데이터 보안에 사용할 패스워드를 두 번 입력하도록 요구한다. 패스워드는 두 번다 동일해야 된다. **gbde** 의 데이터 보호능력은 선택한 패스워드의 난이도에 따라 달라진다.

이 예제에서는 /etc/gbde/ad4s1c 에 저장하였듯이 **gdbe init** 명령은 **gbde** 파티션에 잠금 파일을 생성한다.

경고: **gbde** 잠금 파일은 암호화된 파티션의 내용과 함께 백업해야 된다. 잠금 파일만 삭제하면 의지를 가진 공격자가 **gbde** 파티션을 해독할 수도 있지만, 잠금 파일이 없으면 정당한 주인도 gbde(8)가 전혀 지원하지 못하는 엄청난 작업 없이 암호화된 파티션의 데이터에 접근하지 못한다.

④ **암호화된 파티션을 커널에 붙인다.**

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c
```

암호화된 파티션을 초기화하는 동안 선택한 패스워드를 입력하도록 한다. 암호화된 새로운 장치는 /dev 에서 /dev/device_name.bde 로 나타난다:

```
# ls /dev/ad*
/dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
/dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
/dev/ad0s1a     /dev/ad0s1d     /dev/ad4        /dev/ad4s1c.bde
```

⑤ **암호화된 장치에 파일시스템 생성하기**

커널에 암호화된 장치가 올라오면 장치에 파일시스템을 생성할 수 있다. 암호화된 장치에 파일시스템을 생성하려면 newfs(8)를 사용한다. newfs(8)에 *-O2* 옵션을 사용하면 예전 UFS1 파일시스템을 초기화하는 것보다 새로운 UFS2 파일시스템을 초기화할 때 더 빠르기 때문에 권장한다.

Note: *-O2* 옵션은 FreeBSD 5.1-릴리즈와 새로운 버전에서 기본이다.

```
# newfs -U -O2 /dev/ad4s1c.bde
```

Note: newfs(8) 명령은 *.bde 확장자가 장치 이름에 붙은 **gbde** 파티션에서 실행해야 된다.

⑥ **암호화된 파티션 마운트하기**

암호화된 파일시스템의 마운트 포인트를 생성한다

```
# mkdir /private
```

암호화된 파일시스템을 마운트한다.

```
# mount /dev/ad4s1c.bde /private
```

⑦ **암호화된 파일시스템을 사용할 수 있는지 확인한다**

이제 암호화된 파일시스템을 df(1)로 보고 사용할 수 있다

```
% df -H
Filesystem          Size    Used   Avail Capacity  Mounted on
/dev/ad0s1a        1037M     72M    883M     8%    /
/devfs             1.0K    1.0K      0B   100%    /dev
/dev/ad0s1f        8.1G     55K    7.5G     0%    /home
/dev/ad0s1e        1037M    1.1M    953M     0%    /tmp
/dev/ad0s1d         6.1G    1.9G    3.7G    35%    /usr
/dev/ad4s1c.bde     150G    4.1K    138G     0%    /private
```

암호화된 파티션

16.15.3 암호화된 파일시스템 마운트하기

부팅 후 파일시스템을 사용하기 전에 암호화된 파일시스템을 커널에 다시 붙이고 에러 체크 후 마운트해야 된다. 필요한 명령은 root 로 실행한다.

[예제: 암호화된 파일시스템 마운트하기]

① **커널에 gbde 파티션 붙이기**

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c
```

암호화된 gbde 파티션을 초기화 동안 선택한 패스워드를 입력한다.

② **파일시스템 체크하기**

암호화된 파일시스템을 자동으로 마운트할 수 있도록 /etc/fstab 에 추가하지 않았기 때문에 마운트하기 전에 수동으로 fsck(8)를 실행하여 이 파일시스템의 에러를 체크

한다.

```
# fsck -p -t ffs /dev/ad4s1c.bde
```

③ 암호화된 파일시스템 마운트하기

```
# mount /dev/ad4s1c.bde /private
```

이제 암호화된 파일시스템을 사용할 수 있다.

16.15.3.1 암호화된 파티션 자동으로 마운트하기

암호화된 파티션을 자동으로 붙여서 체크한 후 마운트하는 스크립트를 만드는 것이 가능하지만 보안을 위해 이 스크립트에 gbde(8) 패스워드는 입력하지 않는다. 대신 콘솔이나 ssh(1)를 통해 패스워드를 입력하여 스크립트를 직접 실행하는 것이 권장된다.

16.15.4 gbde 에 의한 암호 보호

gbde(8)는 CBC 모드에서 섹터당 128 비트의 AES 를 사용하여 암호화한다. 디스크에서 각 섹터는 다른 AES 키로 암호화된다. 유저가 입력한 패스워드에서 섹터 키가 파생되는 과정을 포함하여 **gbde** 의 암호화 디자인에 대한 더 많은 정보는 gbde(4)를 본다.

16.15.5 호환성 문제

sysinstall(8)은 **gbde** 로 암호화된 장치와 호환되지 않는다. 모든 *.bde 장치는 sysinstall(8)을 시작하기 전에 커널에서 분리시켜야 되고 그렇지 않으면 장치를 최초로 검색할 때 문제가 발생한다. 예제에서 사용한 암호화된 장치를 분리시키기 위해 다음 명령을 사용한다:

```
# gbde detach /dev/ad4s1c
```

또한 vinum(4)은 geom(4) 서브시스템을 사용하지 않기 때문에 vinum 볼륨에 gbde 를 사용하지 못한다.