우분투 리눅스

시스템 & 네트워크

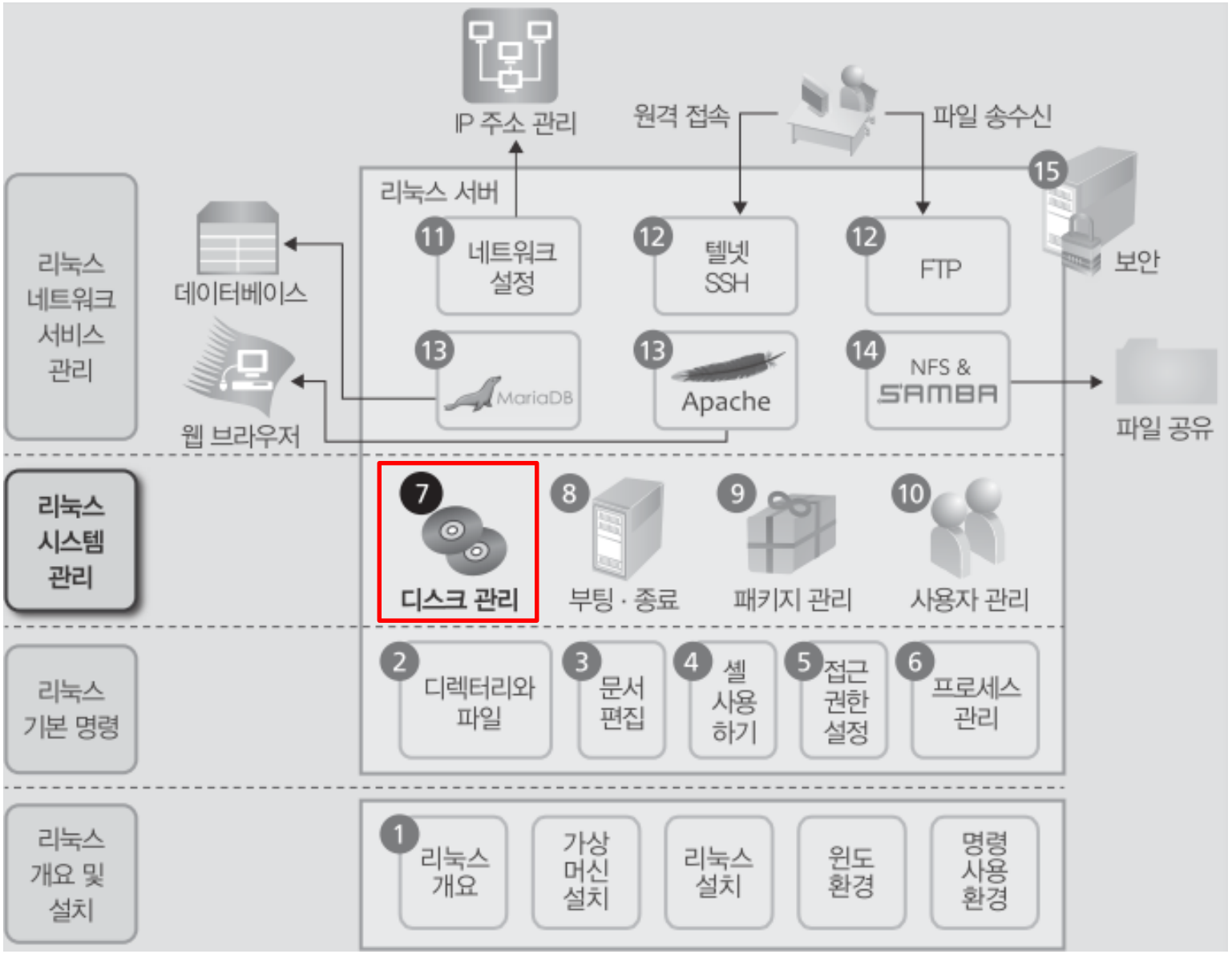
Chapter 07. 파일시스템과 디스크 관리하기

목차

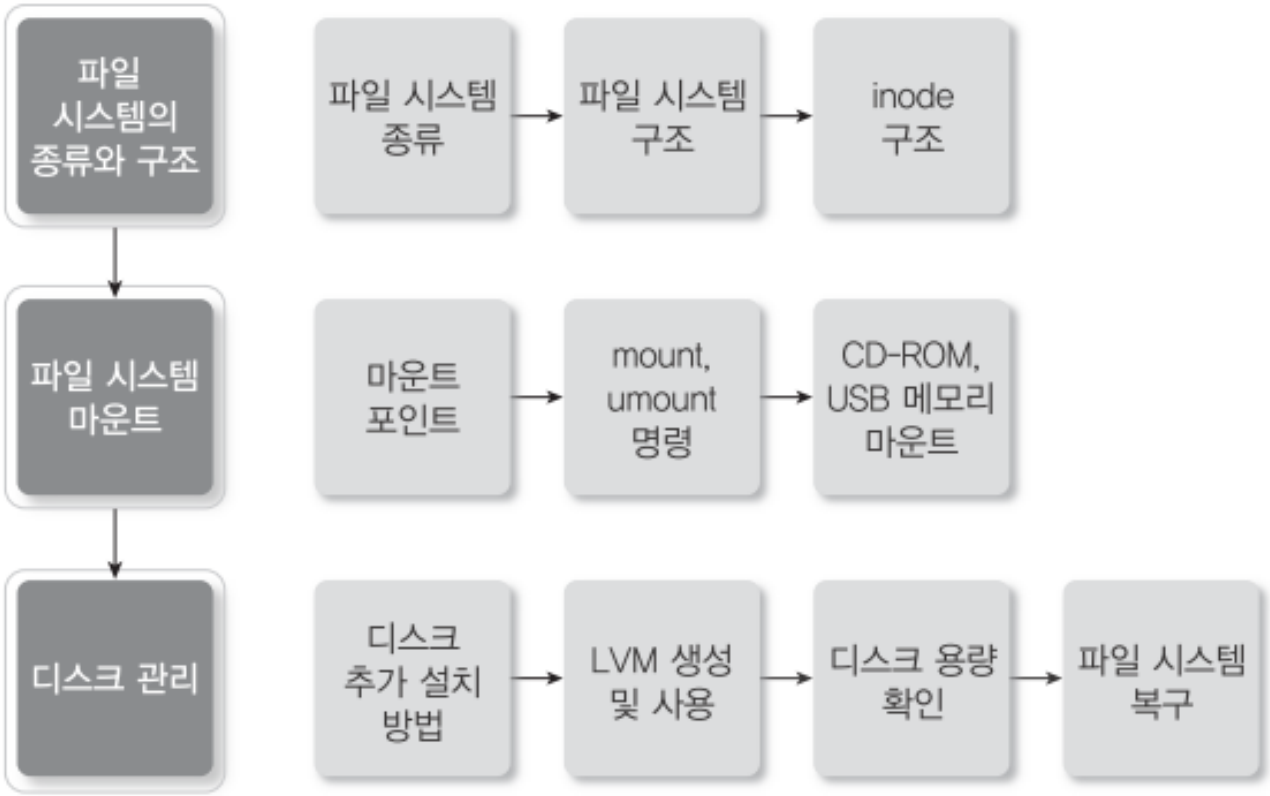
학습목표

- 파일 시스템이 무엇인지 설명할 수 있다.
- 리눅스에서 지원하는 파일 시스템의 종류와 구조를 설명할 수 있다.
- 마운트의 개념을 이해하고 설명할 수 있다.
- CD-ROM이나 USB 메모리 같은 이동식 장치를 마운트하여 사용할 수 있다.
- 새 디스크의 파티션을 나누고 파일 시스템을 생성할 수 있다.
- LVM의 개념을 이해하고 설명할 수 있다.
- 여러 디스크를 묶어서 LVM을 생성하고 마운트할 수 있다.
- 전체 파일 시스템의 사용량과 특정 사용자의 디스크 사용량을 확인할 수 있다.
- 배드 블록을 검사하고, 백업 수퍼블록을 이용하여 파일 시스템을 복구할 수 있다.

리눅스 실습 스터디 맵


IP 주소 관리
원격 접속
파일 송수신
리눅스 서버
15
보안
리눅스 네트워크 서비스 관리
데이터베이스
웹 브라우저
11 네트워크 설정
12 텔넷 SSH
12 FTP
13 MariaDB
13 Apache
14 NFS & SAMBA
파일 공유
리눅스 시스템 관리
7 디스크 관리
8 부팅 · 종료
9 패키지 관리
10 사용자 관리
리눅스 기본 명령
2 디렉터리와 파일
3 문서 편집
4 셸 사용하기
5 접근 권한 설정
6 프로세스 관리
리눅스 개요 및 설치
1 리눅스 개요
가상 머신 설치
리눅스 설치
윈도 환경
명령 사용 환경

00 개요



[그림 7-1] 7장의 내용 구성

01 리눅스 파일시스템의 종류

파일시스템

- 파일과 디렉터리의 집합을 구조적으로 관리하는 체계
- 어떤 구조를 구성하여 파일이나 디렉터리를 관리하느냐에 따라 다양한 형식의 파일 시스템이 존재

리눅스 고유의 디스크 기반 파일 시스템

- ext(ext1)
  - Extended File System'의 약자로 1992년 4월 리눅스 0.96c에 포함되어 발표
  - 파일 시스템의 최대 크기는 2GB, 파일 이름의 길이는 255바이트까지 지원
  - inode 수정과 데이터의 수정 시간 지원이 안 되고, 파일 시스템이 복잡해지고 파편화되는 문제
  - 현재 리눅스에서는 ext 파일 시스템을 사용하지 않음
- ext2
  - ext 파일 시스템이 가지고 있던 문제를 해결하고, 1993년 1월 발표
  - ext2는 ext3 파일 시스템이 도입되기 전까지 사실상 리눅스의 표준 파일 시스템으로 사용
  - 이론적으로 32TB까지 가능
- ext3
  - ext3는 ext2를 기반으로 개발되어 호환이 가능하며 2001년 11월 공개
  - ext3의 가장 큰 장점은 저널링(journaling) 기능을 도입 복구기능 강화
  - 파일 시스템의 최대 크기는 블록의 크기에 따라 2~32TB까지 지원
- ext4
  - ext4 파일 시스템은 1EB(엑사바이트, 1EB=1,024×1,024TB) 이상의 볼륨과 16TB 이상의 파일을 지원
  - ext2 및 ext3와 호환성을 유지하며 2008년 12월 발표

01 리눅스 파일시스템의 종류

■ 리눅스에서 지원하는 다른 디스크 기반 파일 시스템

[표 7-1] 리눅스에서 지원하는 다른 파일 시스템

파일 시스템	기능
msdos	MS-DOS 파티션을 사용하기 위한 파일 시스템이다.
iso9660	CD-ROM, DVD의 표준 파일 시스템으로 읽기 전용으로 사용된다.
nfs	Network File System으로 원격 서버의 디스크를 연결할 때 사용된다.
ufs	Unix File System으로 유닉스의 표준 파일 시스템이다.
vfat	윈도 95, 98, NT를 지원하기 위한 파일 시스템이다.
hpfs	HPFS를 지원하기 위한 파일 시스템이다.
ntfs	윈도의 NTFS를 지원하기 위한 파일 시스템이다.
sysv	유닉스 시스템 V를 지원하기 위한 파일 시스템이다.
hfs	맥의 hfs 파일 시스템을 지원하기 위한 파일 시스템이다.

01 리눅스 파일시스템의 종류

■ 특수 용도의 가상 파일 시스템

[표 7-2] 리눅스의 가상 파일 시스템

파일 시스템	기능
swap	• 스왑 영역을 관리하기 위한 스왑 파일 시스템이다.
tmpfs	• Temporary File System으로 메모리에 임시 파일을 저장하기 위한 파일 시스템이며, 시스템이 재시작할 때마다 기존 내용이 없어진다. • 예를 들면 /tmp 디렉터리이다(페도라 18부터).
proc	• proc 파일 시스템으로 /proc 디렉터리이다. • 커널의 현재 상태를 나타내는 파일을 가지고 있다.
ramfs	• 램디스크를 지원하는 파일 시스템이다.
rootfs	• Root File System으로 / 디렉터리이다. • 시스템 초기화와 관리에 필요한 내용을 관리한다.

01 리눅스 파일시스템의 종류

현재 시스템이 지원하는 파일 시스템 확인하기

- /proc/filesystems는 현재 커널이 지원하는 파일 시스템의 종류를 알려줌

```
nodev   sysfs
nodev   rootfs
nodev   bdev
nodev   proc
nodev   cgroup
nodev   cpuset
nodev   tmpfs
nodev   devtmpfs
nodev   debugfs
nodev   securityfs
nodev   sockfs
nodev   pipefs
nodev   anon_inodefs
nodev   configfs
nodev   devpts
        ext3
        ext4
nodev   ramfs
nodev   hugetlbfs
        vfat
nodev   ecryptfs
        fuseblk
nodev   fuse
nodev   fusectl
nodev   pstore
nodev   mqueue
        iso9660
```

nodev: 해당 파일 시스템이 블록 장치(예 : 디스크)와 연결되어 있지 않다는 것으로 가상 파일 시스템임을 뜻

[그림 7-2] /proc/filesystems의 내용

02 리눅스 파일 시스템의 구조

리눅스의 모든 파일 시스템의 기본 개념

- 파일은 inode로 관리된다.
- 디렉터리는 단순히 파일의 목록을 가지고 있는 파일일 뿐이다.
- 특수 파일을 통해 장치에 접근할 수 있다.

ext4 파일 시스템의 구조

- 효율적으로 디스크를 사용하기 위해 저장 장치를 논리적인 블록의 집합(블록 그룹)으로 구분
- 일반적으로 블록은 4KB이고 실제 크기는 시스템의 설정에 따라 변경 가능
- 블록 그룹 유형
  - 블록 그룹 0 : 파일 시스템의 첫 번째 블록 그룹으로 특별하게 그룹 0 패딩과 수퍼블록, 그룹 디스크립터를 가지고 있다.
  - 블록 그룹 a : 파일 시스템에서 첫 번째 블록 그룹이 아닌 블록 그룹으로 그룹 0 패딩이 없으나 수퍼블록과 그룹 디스크립터에 대한 복사본을 가지고 있다.
  - 블록 그룹 b : 파일 시스템에서 첫 번째 블록 그룹이 아닌 블록 그룹으로 그룹 0 패딩, 수퍼블록, 그룹 디스크립터가 없고 바로 데이터 블록 비트맵으로 시작한다.

02 리눅스 파일 시스템의 구조

ext4 파일 시스템의 구조

[그림 7-3] ext4 파일 시스템의 구조

02 리눅스 파일 시스템의 구조

ext4 파일 시스템의 구조

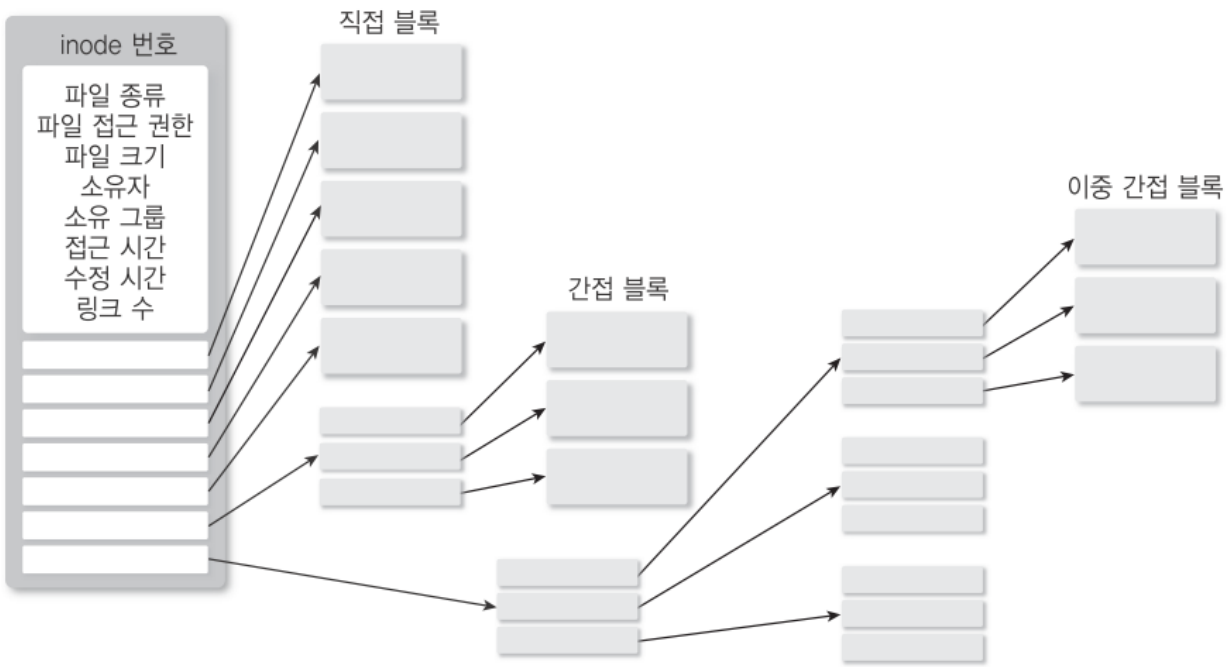
- 그룹 0 패딩
  - 블록 그룹 0의 첫 1,024바이트는 특별한 용도로 사용되는데, x86 부트 섹터와 부가 정보를 저장
- 수퍼블록
  - 파일 시스템과 관련된 다양한 정보가 저장
    - 전체 inode의 개수
    - 할당되지 않은 블록(free block)의 개수
    - 첫 번째 데이터 블록의 주소
    - 그룹당 블록의 개수
    - 파일 시스템의 생태
    - 전체 블록의 개수
    - 할당되지 않은 inode(free inode)의 개수
    - 블록의 크기
    - 마운트 시간
    - 그룹 디스크립터의 크기
  - 수퍼블록에 문제가 생길 경우 전체 파일 시스템을 사용할 수 없게됨
  - 수퍼블록을 다른 블록 그룹에 복사하고, 블록 그룹 0의 수퍼블록을 읽을 수 없을 경우 복사본을 사용하여 복구
- 그룹 디스크립터와 GDT 예약 블록
  - 그룹 디스크립터도 블록 그룹 0에 있는 것으로 수퍼블록의 다음에 위치
  - 그룹 디스크립터에 저장되는 정보
    - 블록 비트맵의 주소
    - inode 테이블의 주소
    - 할당되지 않은 inode의 개수
    - 블록 비트맵, inode 비트맵 체크섬
    - inode 비트맵의 주소
    - 할당되지 않은 블록의 개수
    - 디렉터리의 개수

02 리눅스 파일 시스템의 구조

ext4 파일 시스템의 구조

- 데이터 블록 비트맵과 inode 비트맵
  - 데이터 블록 비트맵은 블록 그룹에 포함된 데이터 블록의 사용 여부를 확인하는 데 사용
  - inode 비트맵은 inode 테이블의 항목(inode)이 사용 중인지를 표시
- inode 테이블과 데이터 블록
  - inode에는 파일 정보를 저장하고 데이터 블록에는 실제 데이터를 저장

inode의 구조



[그림 7-4] inode의 구조

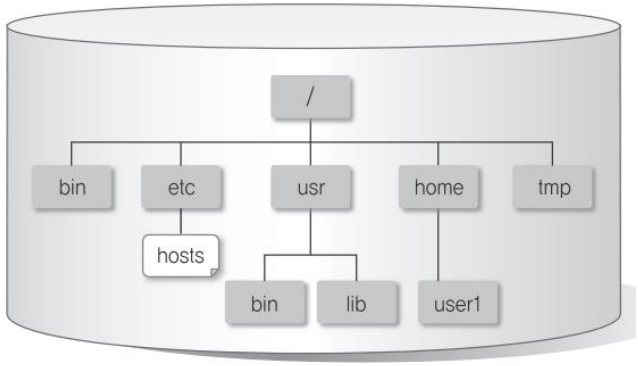
02 리눅스 파일 시스템의 구조

파일 시스템과 디렉터리 계층 구조

- 디렉터리 계층 구조는 리눅스의 전체 파일과 디렉터리를 어떤 구조로 정리하고 관리할 것인지를 정의한 것
- 실제 파일이 저장되어 있는 파일 시스템은 디렉터리 계층 구조에 연결되어야 사용자가 접근 가능

한 파일 시스템으로 구성하기

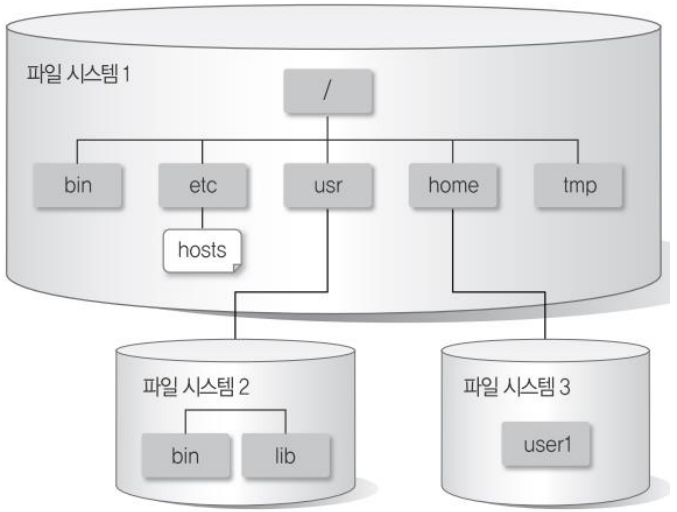
- 디렉터리 계층 구조에서 보이는 모든 디렉터리와 파일을 하나의 파일 시스템으로 구성



[그림 7-5] 한 파일 시스템으로 구성하기

여러 파일 시스템으로 구성하기

- 디렉터리 계층 구조를 여러 파일 시스템으로 구분하여 구성
- 여러 파일 시스템으로 나누어 디렉터리 계층 구조를 구성할 경우, 일부 파일 시스템에 문제가 생기더라도 다른 파일 시스템의 파일은 안전



[그림 7-6] 여러 파일 시스템으로 구성하기

03 파일 시스템 마운트

마운트

- 파일 시스템을 디렉터리 계층 구조의 특정 디렉터리와 연결하는 것

마운트 포인트

- 디렉터리 계층 구조에서 파일 시스템이 연결되는 디렉터리를 마운트 포인트

파일 시스템 마운트 설정 파일

- 리눅스에서 시스템이 부팅될 때 자동으로 파일 시스템이 마운트되게 하려면 /etc/fstab 파일에 설정
- /etc/fstab 파일의 기능: 파일 시스템의 마운트 설정 정보 저장

```
user1@myubuntu:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=4c270aab-6780-4f53-87a9-b8824b6d8b50 /   ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=0c662594-1f44-4da5-a063-fe4da3219ad1 none  swap    sw              0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8 0       0
user1@myubuntu:~$
```

03 파일 시스템 마운트

/etc/fstab 파일의 구조

[그림 7-7] /etc/fstab 파일의 구조

- /etc/fstab 파일의 예
  - 장치명 : UUID=4c270aab-6780-4f53-87a9-b8824b6d8b50
  - 마운트 포인트 : /
  - 파일 시스템의 종류 : ext4
  - 옵션 : errors=remount-ro
  - 덤프 관련 설정 : 1
  - 파일 점검 옵션 : 1
- UUID는 'universally unique identifier'의 약자로 로컬 시스템과 다른 시스템에서 파일 시스템을 유일하게 구분해주는 128비트의 숫자
  - UUID는 시스템의 하드웨어 정보와 시간 정보를 조합하여 랜덤으로 생성
  - UUID로 지정된 장치는 /dev/disk/by-uuid 디렉터리에서 찾을 수 있음

03 파일 시스템 마운트

/etc/fstab 파일의 구조

- 장치명: 파일시스템 장치명, 예를 들어 /dev/hda1, /dev/sda1과 같이 특정 디스크를 지정
- 마운트 포인트: 파일 시스템이 마운트될 마운트 포인트를 설정
- 파일 시스템의 종류: 파일 시스템의 종류를 설정, ext2, ext3, ext4 외
- 옵션: 파일 시스템의 속성을 지정
- 덤프 관련 설정: 0(dump 불가)
  1(dump 가능)
- 파일 점검 옵션
  - 0: 부팅시 fsck 안함
  - 1: 루트 파일시스템
  - 2: 루트 파일시스템 이외

[표 7-3] 파일 시스템 속성 설정 옵션

속성	의미
defaults	일반적인 파일 시스템에 지정하는 속성이다. rw, nouser, auto, exec, suid 속성을 모두 포함한다.
auto	부팅 시 자동으로 마운트된다.
exec	실행 파일이 실행되는 것을 허용한다.
suid	setuid, setgid의 사용을 허용한다.
ro	읽기 전용 파일 시스템이다.
rw	읽기, 쓰기가 가능한 파일 시스템이다.
user	일반 사용자도 마운트가 가능하다.
nouser	일반 사용자의 마운트가 불가능하다. root만 마운트할 수 있다.
noauto	부팅 시 자동으로 마운트하지 않는다.
noexec	실행 파일이 실행되는 것을 허용하지 않는다.
nosuid	setuid, setgid의 사용을 금지한다.
usrquota	사용자별로 디스크 쿼터 설정이 가능하다.
grpquota	그룹별로 디스크 쿼터 설정이 가능하다.

03 파일 시스템 마운트

마운트 관련 명령

mount

기능 파일 시스템을 마운트한다.

형식 mount [옵션] 장치명 마운트 포인트

옵션
-t 파일 시스템 종류 : 파일 시스템 종류를 지정한다.
-o 마운트 옵션 : 마운트 옵션을 지정한다.
-f : 마운트할 수 있는지 점검만 한다.
-r : 읽기만 가능하게 마운트한다(-o ro와 동일).

사용 예
```
mount
mount /dev/sdb1 /
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

umount

기능 파일 시스템을 언마운트한다.

형식 umount [옵션] 장치명 또는 마운트 포인트

옵션 -t 파일 시스템 종류 : 파일 시스템 종류를 지정한다.

사용 예
```
umount /dev/sdb1                    umount /mnt
```

03 파일 시스템 마운트

mount 명령만 사용하는 경우

- 옵션이나 인자를 지정하지 않고 mount 명령을 사용하면 현재 마운트 되어 있는 정보를 출력

```
user1@myubuntu:~$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
(생략)
user1@myubuntu:~$
```

- mount 명령으로 출력되는 정보는 /etc/mtab 파일의 내용과 동일
  - 장치명
  - 마운트 포인트
  - 파일 시스템의 종류
  - 마운트 옵션
  - 사용하지 않는 항목 두 개(0 0)

03 파일 시스템 마운트

mount 명령으로 장치를 연결하는 방법

- 하드디스크를 디렉터리 계층 구조에 연결할 때

```
mount /dev/sdb1 /mnt
```

[표 7-4] 다양한 장치 마운트의 예

장치	mount 명령 형식의 예
ext2 파일 시스템	mount -t ext2 /dev/sdb1 /mnt
ext3 파일 시스템	mount -t ext3 /dev/sdb1 /mnt
ext4 파일 시스템	mount -t ext4 /dev/sdb1 /mnt mount /dev/sdb1 /mnt
CD-ROM	mount -t iso9660 /dev/cdrom /mnt/cdrom
윈도 디스크	mount -t vfat /dev/hdc /mnt
USB 메모리	mount /dev/sdc1 /mnt → 리눅스용 USB 메모리의 경우 mount -t vfat /dev/sdc1 /mnt → 윈도용 USB 메모리의 경우
읽기 전용 마운트	mount -r /dev/sdb1 /mnt
읽기/쓰기 마운트	mount -w /dev/sdb1 /mnt
원격 디스크 마운트	mount -t nfs 서버 주소:/NFS 서버 측 디렉터리 /mnt

03 파일 시스템 마운트

USB 메모리 연결하기(리눅스용)

1. USB 메모리를 USB 슬롯에 꽂고 리눅스 시스템에 인식시킴
   - ① VMware Player의 Player 메뉴에서 ''Removable Devices→Removable Disk→Connect(Disconnect from host)'를 선택
   - ② USB 장치를 호스트 OS에서 분리하여 가상 머신에 연결한다는 메시지가 출력
   - ③ 기존에 사용하던 USB 메모리이면 자동으로 디렉터리에 마운트
   - ④ mount 명령을 실행해보면 마지막에 장치가 추가되었는지 확인 가능

```
user1@myubuntu:~$ mount
(생략)
/dev/sdb1 on /media/user1/5255-B26B type vfat (rw,nosuid,nodev,uid=1000,gid=
1000,shortname=mixed,dmask=0077,utf8=1,showexec,flush,uhelper=udisks2)
user1@myubuntu:~$
```

   - ⑤ 마운트를 해제하고 파일 시스템 생성 작업을 해야 함

```
user1@myubuntu:~$ umount /dev/sdb1
user1@myubuntu:~$
```

03 파일 시스템 마운트

USB 메모리 연결하기(리눅스용)

2. USB 메모리의 장치명을 확인: 장치명은 fdisk -l 명령으로 확인 가능(root 권한)

```
user1@myubuntu:~$ fdisk -l
user1@myubuntu:~$
user1@myubuntu:~$ sudo fdisk -l
(생략)
Disk /dev/sdb: 2056 MB, 2056257536 bytes
16 heads, 32 sectors/track, 7844 cylinders, total 4016128 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x4595dc7f
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1              32     4016127     2008048    6  FAT16
user1@myubuntu:~$
```

03 파일 시스템 마운트

USB 메모리 연결하기(리눅스용)

3. USB 메모리에 파티션을 생성

① fdisk 명령을 실행

```
user1@myubuntu:~$ sudo fdisk /dev/sdb
Command (m for help):
```

② 새로운 파티션을 생성하려는 것이므로 n을 입력: primary 선택

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p):
```

③ 생성할 파티션의 번호를 지정: 1번 (만약 이미 1번이 사용중이면 삭제하고 새로 생성)

```
Using default response p
Partition number (1-4, default 1):
Using default value 1
First sector (2048-4016127, default 2048):
```

03 파일 시스템 마운트

USB 메모리 연결하기(리눅스용)

3. USB 메모리에 파티션을 생성

④ 시작 섹터와 마지막 섹터를 지정

```
Using default value 2048
Last sector, +sectors or +sizeK,M,G (2048-4016127, default 4016127):
Using default value 4016127
Command (m for help):
```

⑤ p 명령을 사용하여 파티션이 제대로 설정되었는지 확인

```
Command (m for help): p
Disk /dev/sdb: 2056 MB, 2056257536 bytes
16 heads, 32 sectors/track, 7844 cylinders, total 4016128 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x4595dc7f
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4016127     2007040   83  Linux
Command (m for help):
```

⑥ w를 입력하여 설정한 파티션 정보를 파티션 테이블에 기록: USB 메모리에 파티션이 생성 완료

03 파일 시스템 마운트

USB 메모리 연결하기(리눅스용)

4. 생성한 파티션을 포맷하여 파일 시스템을 생성

```
user1@myubuntu:~$ sudo mke2fs -t ext4 /dev/sdb1
mke2fs 1.42.8 (20-Jun-2013)
Filesystem label=
(생략)
user1@myubuntu:~$
```

5. USB 마운트

```
user1@myubuntu:~$ sudo mount /dev/sdb1 /mnt
user1@myubuntu:~$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
(생략)
/dev/sdb1 on /mnt type ext4 (rw)
user1@myubuntu:~$
```

6. USB에 파일 복사

```
user1@myubuntu:~$ cd /mnt
user1@myubuntu:/mnt$ sudo cp /etc/hosts .
user1@myubuntu:/mnt$ ls
hosts  lost+found
user1@myubuntu:/mnt$
```

03 파일 시스템 마운트

장치 연결 해제하기

- USB 사용 완료후 제거하는 방법
  - 마운트 해제: 오류 발생 -> 'busy'라는 메시지가 출력되면 해당 디렉터리를 누군가가 사용하고 있어서 마운트를 해제할 수 없다는 뜻

```
user1@myubuntu:/mnt$ sudo umount /mnt
umount: /mnt: device is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
user1@myubuntu:/mnt$
```

  - /mnt 디렉터리에서 이동하여 umount 명령을 실행하면 정상적으로 마운트가 해제

```
user1@myubuntu:/mnt$ cd
user1@myubuntu:~$ sudo umount /mnt
user1@myubuntu:~$
```

  - USB 제거 가능

03 파일 시스템 마운트

윈도용 USB 메모리 연결하기

1. USB 메모리를 USB 슬롯에 꽂고 리눅스 시스템에 인식시킨다.
2. USB 메모리의 장치명을 확인한다.

```
user1@myubuntu:~$ sudo fdisk -l
(생략)
Disk /dev/sdb: 8004 MB, 8004304896 bytes
35 heads, 21 sectors/track, 21269 cylinders, total 15633408 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               32  15633407     7816688    b  W95 FAT32
user1@myubuntu:~$
```

3. USB 메모리를 /mnt 디렉터리에 마운트

```
user1@myubuntu:~$ sudo mount -t vfat /dev/sdb1 /mnt
user1@myubuntu:~$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
(생략)
/dev/sdb1 on /mnt type vfat (rw)
user1@myubuntu:~$
```

03 파일 시스템 마운트

윈도용 USB 메모리 연결하기

4. USB 메모리가 디렉터리에 연결되었으므로 사용이 가능하다.

```
user1@myubuntu:~$ cd /mnt
user1@myubuntu:/mnt$ ls
test.txt
user1@myubuntu:/mnt$
```

5. USB 메모리를 사용하고 나면 umount 명령으로 마운트를 해제한다.

```
user1@myubuntu:/mnt$ sudo cp /etc/hosts .
user1@myubuntu:/mnt$ ls
hosts  test.txt
user1@myubuntu:/mnt$
```

03 파일 시스템 마운트

CD-ROM 연결하기

1. CD-ROM 장치를 USB로 연결하여 리눅스 시스템에 인식시킨다.
2. CD-ROM를 마운트한다.

```
user1@myubuntu:~$ sudo mount -t iso9660 /dev/cdrom /mnt
[sudo] password for user1:
mount: block device /dev/sr0 is write-protected, mounting read-only
user1@myubuntu:~$
```

```
user1@myubuntu:~$ ls -l /dev/cdrom
lrwxrwxrwx 1 root root 3  2월 22 10:30 /dev/cdrom -> sr0
user1@myubuntu:~$
```

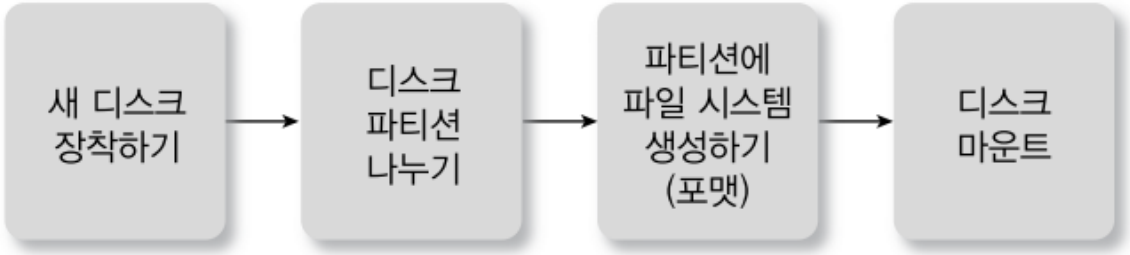
3. CD-ROM의 내용을 확인하고 사용할 수 있다.

```
user1@myubuntu:~$ ls /mnt
README.diskdefines  boot    dists    isolinux    pics  preseed  wubi.exe
autorun.inf         casper  install  md5sum.txt   pool  ubuntu
user1@myubuntu:~$
```

4. CD-ROM 장치를 사용하고 나면 umount 명령을 사용하여 마운트를 해제한다.

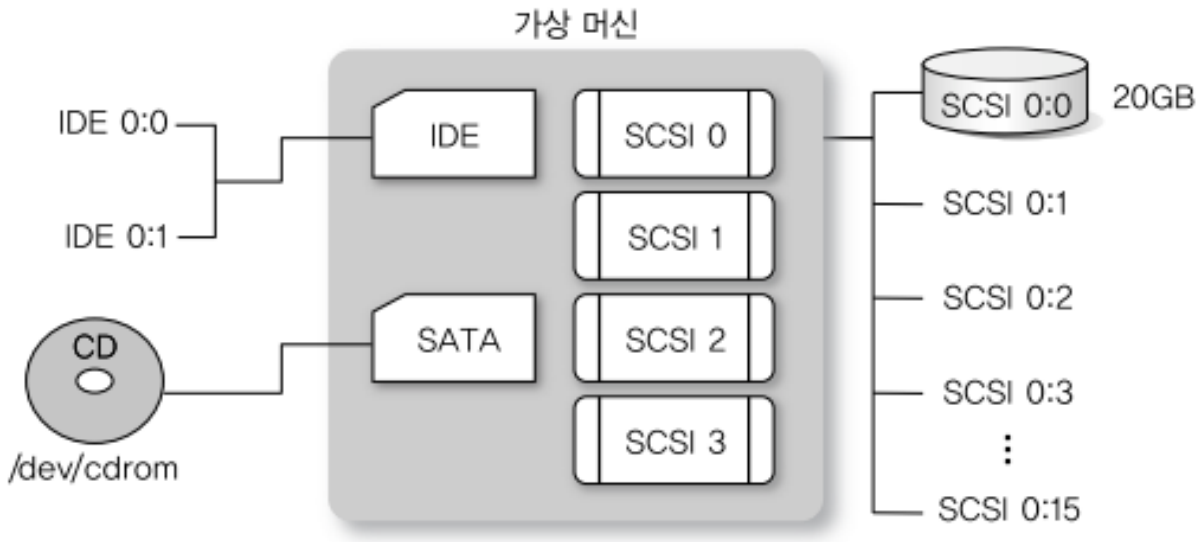
04 디스크 추가 설치

■ 디스크 추가 단계



[그림 7-10] 디스크 추가 단계

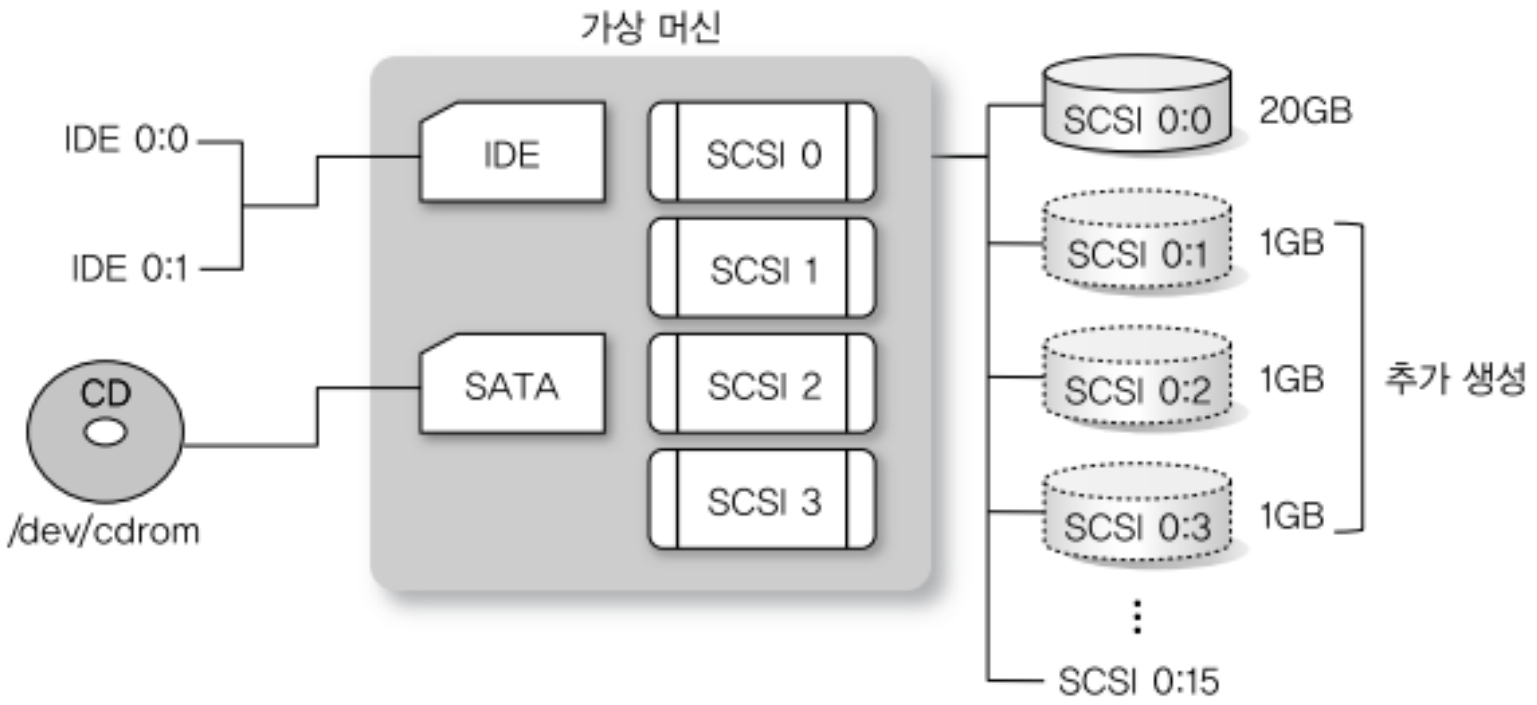
■ 가상 머신의 디스크 구성도



[그림 7-11] 가상 머신의 디스크 컨트롤러와 디스크 구성 개념도

04 디스크 추가 설치

- **가상 머신에 디스크 추가하기**



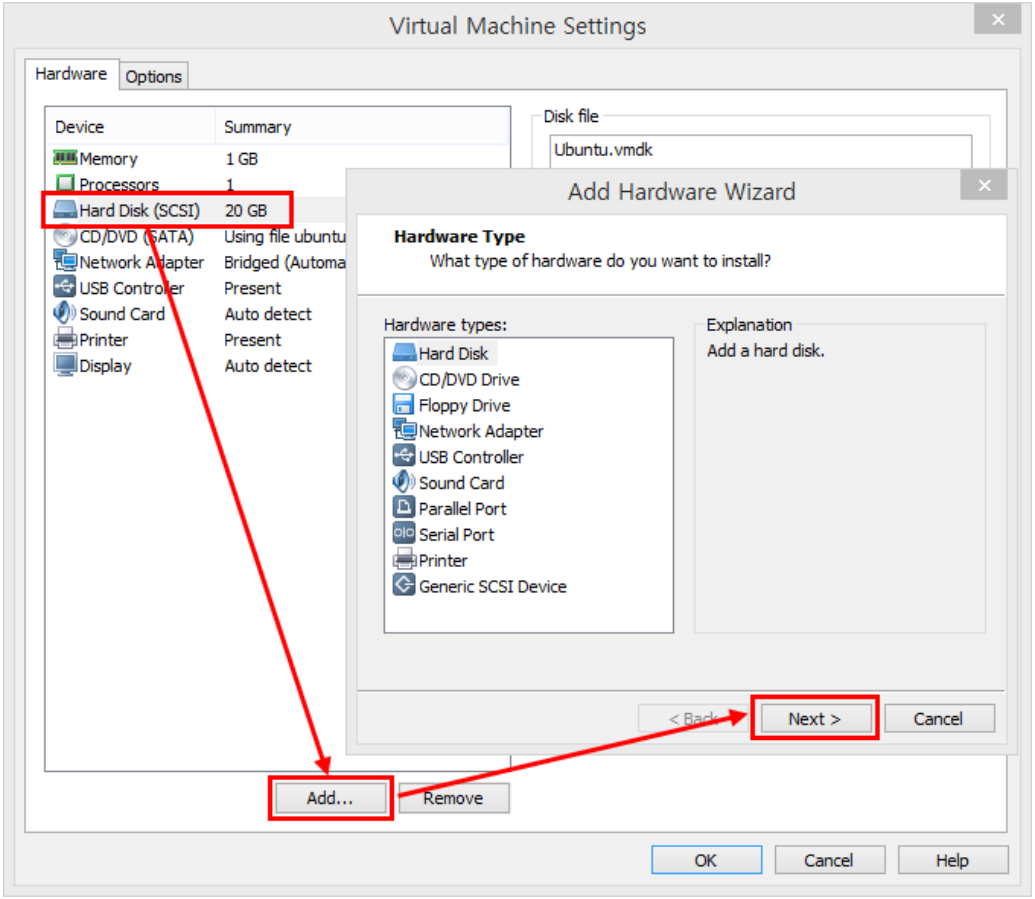
[그림 7-12] 가상 머신의 디스크 추가 구성 개념도

04 디스크 추가 설치

가상 머신에 디스크 추가하기

① Player 메뉴에서 'Manage→Virtual Machine Settings'를 선택하거나, 리눅스를 종료한 후에 VMware Player 메인 화면에서 'Edit virtual machine settings' 선택

② Virtual Machine Settings 창에서 Add...를 클릭 -> Add Hardware Wizard 창 -> Next



[그림 7-13]

Virtual Machine Settings 창에서 Add… 선택

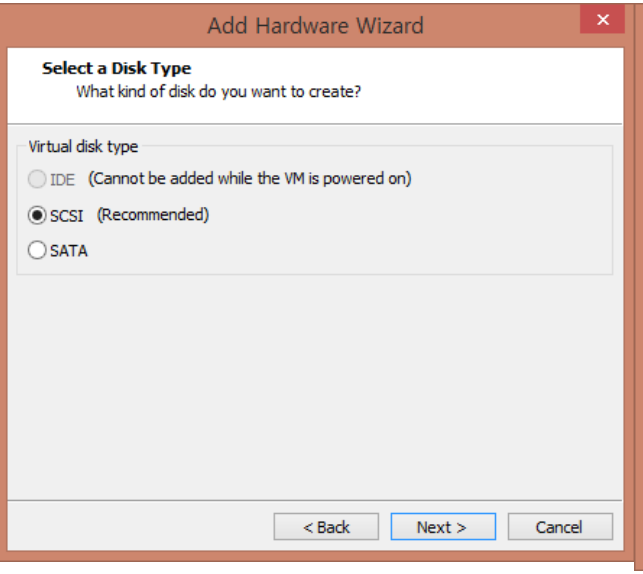
04 디스크 추가 설치

■ 가상 머신에 디스크 추가하기

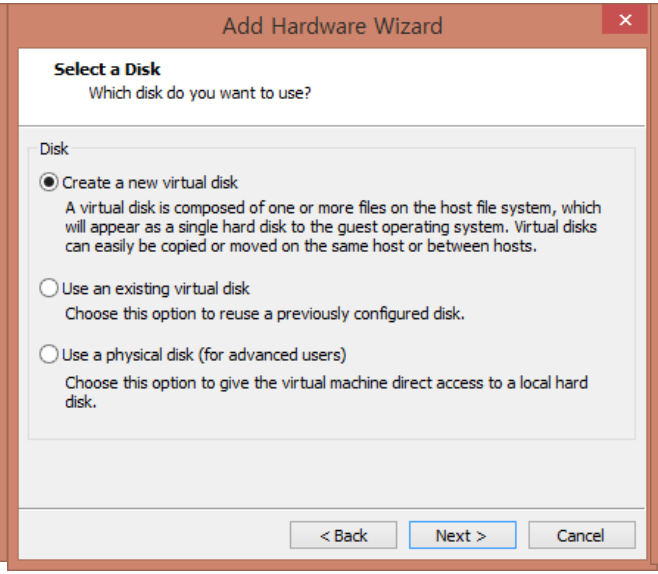
③ Select a Disk : 'Create a new virtual disk'를 선택하고 Next를 클릭

④ Select a Disk Type : 디스크의 종류는 SCSI를 선택한다.

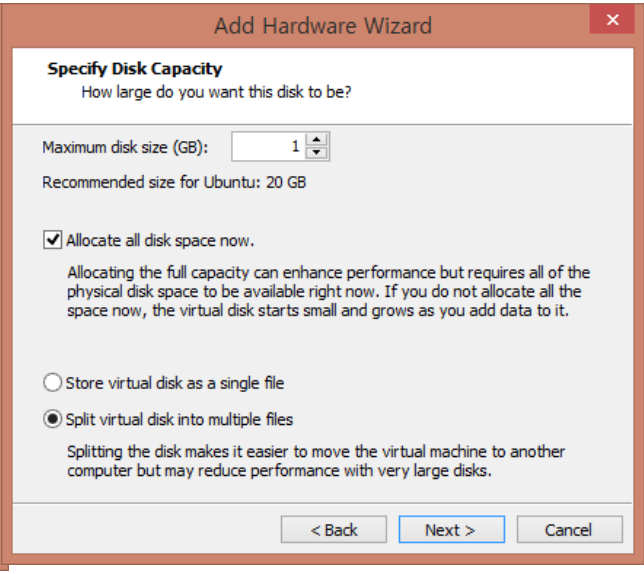
⑤ Specify Disk Capacity : 디스크의 용량을 설정한다. 테스트용이므로 1GB로 설정



[그림 7-14] Select a Disk 창



[그림 7-15] Select a Disk Type 창



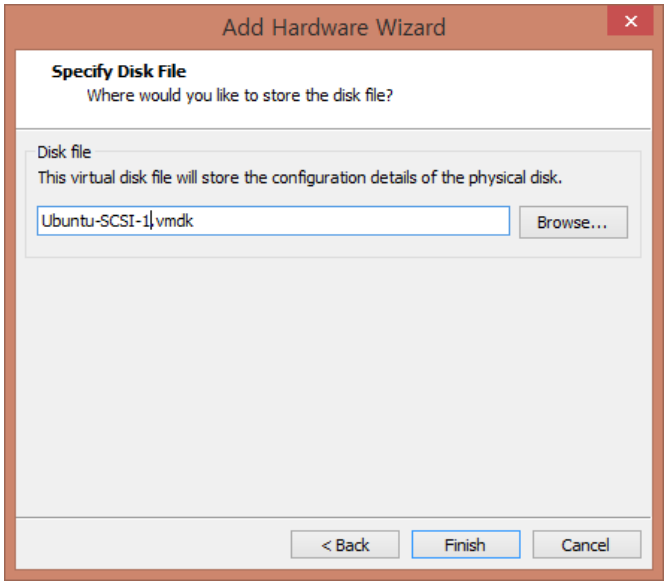
[그림 7-16] Specify Disk Capacity 창

04 디스크 추가 설치

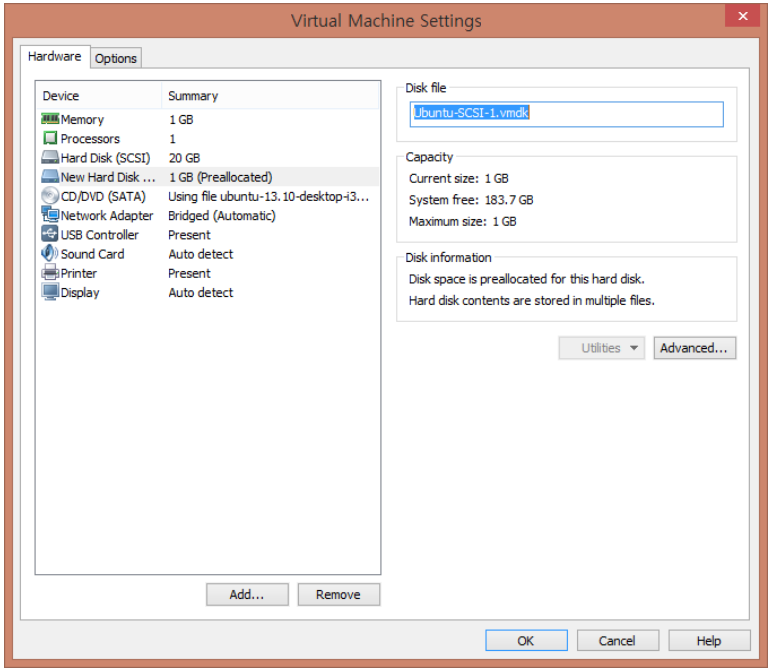
가상 머신에 디스크 추가하기

⑥ Specify Disk File : 가상 디스크의 파일명을 지정

⑦ 디스크 파일이 생성되고 디스크 추가 작업이 완료된다. Virtual Machine Settings 창에서 새로 생성된 디스크를 확인



[그림 7-17] Specify Disk File 창



[그림 7-18] 디스크 추가 완료

04 디스크 추가 설치

디스크 파티션 나누기

- 파티션이란 하나의 디스크를 독립된 영역으로 구분하는 것
- 디스크 전체를 하나의 파티션으로 사용할 수도 있고 여러 개의 파티션으로 나누어 사용할 수도 있음

디스크 장치의 이름

- 리눅스에서 IDE 컨트롤러에 연결된 디스크는 /dev/hd로 시작
- SCSI나 SATA 컨트롤러에 장착된 디스크는 /dev/sd로 시작하는 이름을 주로 사용
- 최근에는 IDE, SCSI 등 구분 없이 모두 /dev/sd로 시작하는 이름을 사용
- 컨트롤러에 연결되는 디스크의 순서에 따라 다음과 같이 알파벳이 추가 : /dev/sda, /dev/sdb

디스크 장치의 이름과 파티션 표시하기

- 하나의 디스크를 여러 개의 파티션으로 구분할 경우 파티션은 디스크 장치 이름의 뒤에 숫자를 붙여서 표시
  - /dev/sda : 첫째 디스크 전체를 의미하는 장치 이름
  - /dev/sda0 : 디스크의 첫째 파티션
  - /dev/sda1 : 디스크의 둘째 파티션

04 디스크 추가 설치

■ fdisk 명령

fdisk

기능 디스크의 파티션 생성, 삭제, 보기 등 파티션을 관리한다.

형식 fdisk [옵션] 장치명

옵션 -b <크기> : 섹터 크기를 지정한다(512, 1024, 2048, 4096).
-l : 파티션 테이블을 출력한다.

사용 예 fdisk /dev/sdb fdisk -l

[표 7-5] fdisk 내부 명령

내부 명령	기능	내부 명령	기능
a	부팅 파티션을 설정한다.	p	파티션 테이블을 출력한다.
b	BSD 디스크 라벨을 편집한다.	q	작업 내용을 저장하지 않고 종료한다.
c	도스 호환성을 설정한다.	s	새로운 빈 Sun 디스크 라벨을 생성한다.
d	파티션을 삭제한다.	t	파티션의 시스템 ID를 변경한다(파일 시스템 종류 변경).
l	사용 가능한 파티션 종류를 출력한다.	u	항목 정보를 변경 · 출력한다.
m	도움말을 출력한다.	v	파티션 테이블을 검사한다.
n	새로운 파티션을 추가한다.	w	파티션 정보를 디스크에 저장하고 종료한다.
o	새로운 빈 DOS 파티션을 생성한다.	x	실린더 개수 변경 등 전문가를 위한 부가적 기능이다.

04 디스크 추가 설치

- **fdisk로 파티션 정보 보기: fdisk -l**

```
user1@myubuntu:~$ sudo fdisk -l
[sudo] password for user1:

Disk /dev/sdd: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/sdd doesn't contain a valid partition table

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/sdb doesn't contain a valid partition table

(생략)
user1@myubuntu:~$
```

04 디스크 추가 설치

fdisk로 파티션 나누기

① fdisk 명령을 실행: 파티션 작업을 할 때는 장치명을 인자로 지정

```
user1@myubuntu:~$ sudo fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0x2fa4c807.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
Command (m for help):
```

② 새로운 파티션을 생성: n을 입력 -> 파티션의 종류를 선택하는 메뉴가 출력

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
```

③ 파티션의 종류를 선택: 기본 파티션을 선택 -> 파티션 번호를 선택하는 메뉴가 출력

```
Select (default p): p
Partition number (1-4, default 1):
```

04 디스크 추가 설치

fdisk로 파티션 나누기

④ 파티션의 번호를 선택: 1을 선택

```
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
```

⑤ 파티션의 크기를 설정: +500M을 입력

```
First sector (2048-2097151, default 2048):
Using default value 2048
Last sector, +sectors or +sizeK,M,G (2048-2097151, default 2097151): +500M
Command (m for help):
```

⑥ 파티션 설정 정보를 확인: p를 입력

```
Command (m for help): p
Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x2fa4c807
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     1026047      512000   83  Linux
Command (m for help):
```

04 디스크 추가 설치

fdisk로 파티션 나누기

⑦ n을 입력하여 두 번째 파티션을 생성: 기본 파티션(p), 파티션 번호는 2, 용량은 +500M을 선택

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 2):
Using default value 2
First sector (1026048-2097151, default 1026048):
Using default value 1026048
Last sector, +sectors or +sizeK,M,G (1026048-2097151, default 2097151): +500M
Command (m for help):
```

04 디스크 추가 설치

fdisk로 파티션 나누기

⑧ 파티션 설정 정보를 확인: p를 입력

```
Command (m for help): p
Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x2fa4c807
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     1026047      512000   83  Linux
/dev/sdb2         1026048     2050047      512000   83  Linux
Command (m for help):
```

⑨ w를 입력하여 파티션 설정 정보를 디스크에 기록하고 fdisk를 종료

```
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
user1@myubuntu:~$
```

04 디스크 추가 설치

■ 파일 시스템 생성하기

- 파일 시스템은 이 파티션에서 파일과 디렉터리를 관리하기 위한 구조를 만드는 것

■ 파일 시스템 생성 명령 : mkfs, mke2fs

mkfs

기능 리눅스 파일 시스템을 만든다.

형식 mkfs [옵션] 장치명

옵션 -t 종류 : 파일 시스템의 종류를 지정한다(기본 값은 ext2).

사용 예 mkfs /dev/sdb1 mkfs -t ext4 /dev/sdb1

mke2fs

기능 리눅스 개정판 확장 파일 시스템(ext2, ext3, ext4)을 만든다.

형식 mke2fs [옵션] 장치명

옵션
-t 종류 : 파일 시스템의 종류를 지정한다. 기본 값은 ext2이다.
-b 블록 크기 : 블록 크기를 바이트 수로 지정한다.
-c : 배드 블록을 체크한다.
-f 프래그먼트 크기 : 프래그먼트 크기를 바이트 수로 지정한다.
-i inode당 바이트 수 : inode당 바이트 수를 지정한다. 기본 값은 4,096바이트이다.
-m 예약 블록 퍼센트 : 슈퍼유저에게 예약해둘 블록의 퍼센트를 지정한다. 기본 값은 5%이다.

사용 예 mke2fs /dev/sdb1 mke2fs -t ext4 /dev/sdb1

04 디스크 추가 설치

- **mkfs 명령으로 파일 시스템 생성하기: /dev/sdb1 파티션에 ext2 파일 시스템 생성**

```
user1@myubuntu:~$ sudo mkfs /dev/sdb1
mke2fs 1.42.8 (20-Jun-2013)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128016 inodes, 512000 blocks
25600 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409
Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
user1@myubuntu:~$
```

04 디스크 추가 설치

- **mkfs.ext3 명령으로 /dev/sdb2 파티션에 ext3 파일 시스템을 생성**

```
user1@myubuntu:~$ sudo mkfs.ext3 /dev/sdb2
mke2fs 1.42.8 (20-Jun-2013)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128016 inodes, 512000 blocks
25600 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409
Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
user1@myubuntu:~$
```

04 디스크 추가 설치

- **mke2fs 명령으로 /dev/sdc2 파티션에 ext4 파일 시스템 생성**

```
user1@myubuntu:~$ sudo mke2fs -t ext4 -b 4096 /dev/sdc2
mke2fs 1.42.8 (20-Jun-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
128000 inodes, 128000 blocks
6400 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=134217728
4 block groups
32768 blocks per group, 32768 fragments per group
32000 inodes per group
Superblock backups stored on blocks:
        32768, 98304
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
user1@myubuntu:~$
```

04 디스크 추가 설치

디스크 마운트

- 파일 시스템을 디렉터리 계층 구조에 마운트

마운트 포인트 준비하기

```
user1@myubuntu:~$ sudo mkdir /mnt/hdd1
user1@myubuntu:~$ sudo mkdir /mnt/hdd2
user1@myubuntu:~$ sudo mkdir /mnt/hdd3
```

파일 시스템 마운트하기

- /dev/sdb1을 /mnt/hdd1 디렉터리에 마운트

```
user1@myubuntu:~$ sudo mount /dev/sdb1 /mnt/hdd1
user1@myubuntu:~$
```

- /dev/sdb2 파티션을 /mnt/hdd2 디렉터리에 마운트

```
user1@myubuntu:~$ sudo mount -t ext3 /dev/sdb2 /mnt/hdd2
user1@myubuntu:~$
```

- 마운트 결과

```
user1@myubuntu:~$ mount
(생략)
/dev/sdb1 on /mnt/hdd1 type ext2 (rw)
/dev/sdb2 on /mnt/hdd2 type ext3 (rw)
user1@myubuntu:~$
```

04 디스크 추가 설치

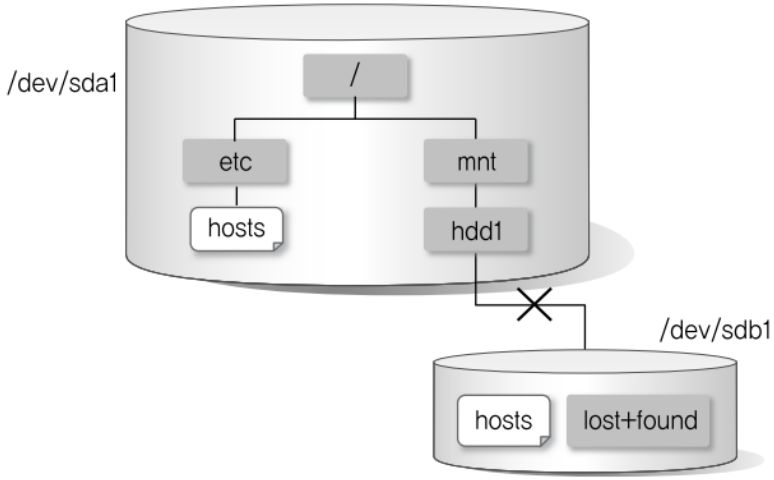
파일 시스템 사용하기

```
user1@myubuntu:~$ sudo cp /etc/hosts /mnt/hdd1
user1@myubuntu:~$ ls /mnt/hdd1
hosts  lost+found
user1@myubuntu:~$
```

이 상태에서 마운트를 해제하면

```
user1@myubuntu:~$ sudo umount /mnt/hdd1
user1@myubuntu:~$ ls /mnt/hdd1
user1@myubuntu:~$
```

- 마운트가 해제된 뒤 /mnt/hdd1 디렉터리에 아무 파일도 없음
- 파일 시스템의 마운트가 해제되면 이 파티션과의 연결이 끊어지므로 /mnt/hdd1에서 hosts 파일을 볼 수 없음



[그림 7-19] 디스크 마운트 해제와 파일의 관계

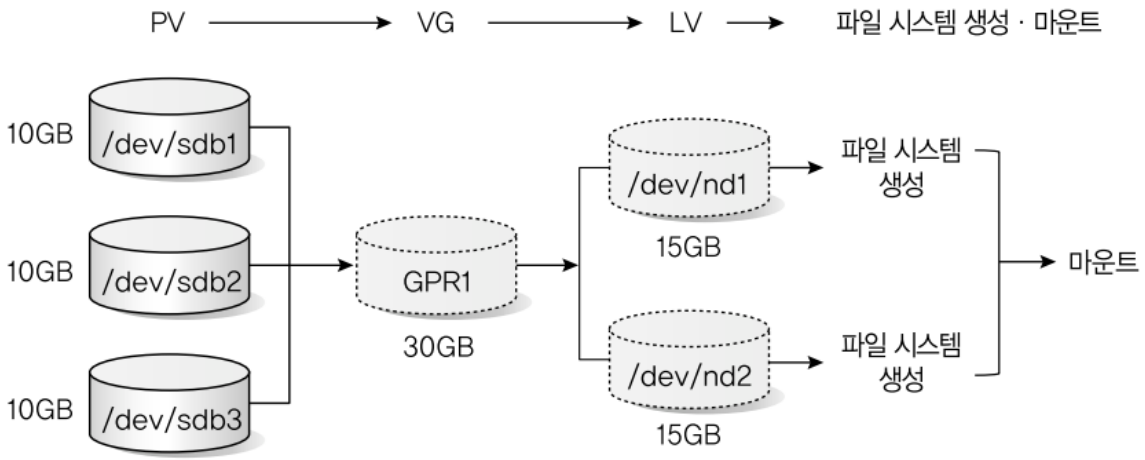
04 디스크 추가 설치

여러 디스크를 하나처럼 사용하기

- 디스크의 용량이 부족할 때 여러 개의 디스크를 하나의 디스크처럼 사용

LVM의 기본 개념

- LVM은 독립적으로 구성된 디스크 파티션을 하나로 연결하여 한 파티션처럼 사용할 수 있도록 해줌
- LVM 관련 용어
  - PV(physical volume, 물리 볼륨) : /dev/sdb1, /dev/sdb2 같은 실제 하드디스크의 파티션을 의미
  - VG(volume group, 볼륨 그룹) : 여러 개의 PV를 그룹으로 묶은 것을 뜻한다. 예를 들어 /dev/sdb1, /dev/sdb2가 GRP1이라는 그룹을 만들 때 GRP1을 VG라고 함
  - LV(logical volume, 논리 볼륨) : VG를 다시 적절한 크기의 파티션으로 나눌 때 각 파티션을 LV라고 함
  - PE(physical extent) : PV가 가진 일정한 블록을 의미
  - LE(logical extent) : LV가 가진 일정한 블록을 의미



[그림 7-20] LVM의 기본 개념도

04 디스크 추가 설치

[표 7-6] LVM 관련 명령

구분	기능	명령
PV	PV 생성	pvcreate 파티션 이름
	PV 상태 확인	pvscan
VG	VG 생성	vgcreate VG명 파티션(PV)명1 파티션(PV)명2 ...
	VG 활성화	vgchange -a y VG명
	VG 비활성화	vgchange -a n VG명
	VG 삭제	vgremove VG명
	VG 정보 확인	vgdisplay -v VG명
	VG에 PV 추가	vgextend VG명 PV명
	VG에서 PV 삭제	vgreduce VG명 PV명
	VG명 변경	vgrename 기존 VG명 새 VG명
LV	LV 생성	lvcreate -l PE 수 VG명 -n LV명
	LV 삭제	lvremove LV명
	LV 상태 확인	lvscan
	LV 용량 확대	lvextent -l +PE 수 LV명
	LV 용량 축소	lvextent -l -PE 수 LV명

04 디스크 추가 설치

- **LVM 생성 과정**

[그림 7-21] LVM 생성 흐름도

- **LVM 설치 : sudo apt-get install lvm2**

04 디스크 추가 설치

LVM 생성하기

- /dev/sdb1, /dev/sdb2는 크기가 각각 500MB이다. 이를 LVM으로 변환하고 1GB짜리 LV를 생성하여 마운트

① 파일 시스템 종류 83(Linux)에서 8e(Linux LVM)으로 변경: fdisk

```
user1@myubuntu:~$ sudo fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x2fa4c807
   Device Boot       Start           End  Blocks    Id  System
/dev/sdb1              2048  1026047  512000    83  Linux
/dev/sdb2           1026048  2050047  512000    83  Linux

Command (m for help):  t
Partition number (1-4): 1
Hex code (type L to list codes): 8e
Changed system type of partition 1 to 8e (Linux LVM)
(생략)
```

04 디스크 추가 설치

LVM 생성하기

```
Command (m for help):  p

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x2fa4c807

   Device Boot  Start  End  Blocks    Id  System
/dev/sdb1  2048  1026047  512000  8e  Linux LVM
/dev/sdb2  1026048  2050047  512000  8e  Linux LVM

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
user1@myubuntu:~$
```

04 디스크 추가 설치

LVM 생성하기

② /dev/sdb1, /dev/sdb2에 PV를 생성

```
user1@myubuntu:~$ sudo pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
user1@myubuntu:~$ sudo pvcreate /dev/sdb2
  Physical volume "/dev/sdb2" successfully created
user1@myubuntu:~$
```

③ pvscan 명령으로 PV의 상태를 확인

```
user1@myubuntu:~$ sudo pvscan
  PV /dev/sdb1                      lvm2 [500.00 MiB]
  PV /dev/sdb2                      lvm2 [500.00 MiB]
  Total: 2 [1000.00 MiB] / in use: 0 [0   ] / in no VG: 2 [1000.00 MiB]
user1@myubuntu:~$
```

④ 두 PV를 통합하여 VG를 생성: VG의 이름은 grp1

```
user1@myubuntu:~$ sudo vgcreate grp1 /dev/sdb1 /dev/sdb2
  Volume group "grp1" successfully created
user1@myubuntu:~$
```

04 디스크 추가 설치

LVM 생성하기

⑤ 생성된 VG grp1을 활성화

```
user1@myubuntu:~$ sudo vgchange -a y grp1
  0 logical volume(s) in volume group "grp1" now active
user1@myubuntu:~$
```

⑥ 활성화된 VG grp1의 상태를 vgdisplay 명령으로 확인

```
user1@myubuntu:~$ sudo vgdisplay -v grp1
    Using volume group(s) on command line
    Finding volume group "grp1"
  --- Volume group ---
  VG Name               grp1
  System ID
  Format                lvm2
  Metadata Areas        2
(생략)
  PV Name               /dev/sdb2
  PV UUID               h1Vt26-2QP0-fAbi-ABpR-s20x-6R63-2oPbEx
  PV Status             allocatable
  Total PE / Free PE    124 / 124
user1@myubuntu:~$
```

04 디스크 추가 설치

LVM 생성하기

⑦ 하나의 LV를 생성

```
user1@myubuntu:~$ sudo lvcreate -l 248 grp1 -n mylvm1
  Logical volume "mylvm1" created
user1@myubuntu:~$
```

⑧ 생성된 LV의 상태를 확인

```
user1@myubuntu:~$ sudo lvscan
  ACTIVE            '/dev/grp1/mylvm1' [992.00 MiB] inherit
user1@myubuntu:~$
```

04 디스크 추가 설치

LVM 생성하기

⑨ LV mylvm1에 ext4 파일 시스템을 생성

```
user1@myubuntu:~$ sudo mke2fs -t ext4 /dev/grp1/mylvm1
mke2fs 1.42.8 (20-Jun-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
63488 inodes, 253952 blocks
12697 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
7936 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks):
done
Writing superblocks and filesystem accounting information: done
user1@myubuntu:~$
```

04 디스크 추가 설치

LVM 생성하기

⑩ VG의 상태를 확인하여 LV의 정보가 수정되었는지 확인

```
user1@myubuntu:~$ sudo vgdisplay -v grp1
    Using volume group(s) on command line
    Finding volume group "grp1"
  --- Volume group ---
(생략)
  --- Logical volume ---
  LV Path                /dev/grp1/mylvm1
  LV Name                mylvm1
  VG Name                grp1
  LV UUID  m7ypbD-4PZZ-c8ZI-Ftrz-YV1T-dB04-szjKEI
  LV Write Access  read/write
  LV Creation host, time myubuntu, 2014-03-10 00:55:53 +0900
  LV Status  available
  # open                 0
  LV Size                992.00 MiB
  Current LE  248
  Segments  2
  Allocation  inherit
  --- Physical volumes ---
(생략)
user1@myubuntu:~$
```

04 디스크 추가 설치

LVM 생성하기

⑪ LV를 /mnt/lvm 디렉터리에 마운트하고 파일을 복사

```
user1@myubuntu:~$ sudo mkdir /mnt/lvm
user1@myubuntu:~$ sudo mount /dev/grp1/mylvm1 /mnt/lvm
user1@myubuntu:~$ sudo cp /etc/hosts /mnt/lvm
user1@myubuntu:~$ ls /mnt/lvm
hosts  lost+found
user1@myubuntu:~$
```

05 디스크 관리

■ 파일 시스템별 디스크 사용량 확인하기 : df

df

기능 디스크의 남은 공간에 대한 정보를 출력한다.

형식 df [옵션] [파일 시스템]

옵션
- -a : 모든 파일 시스템을 대상으로 디스크 사용량을 확인한다.
- -k : 디스크 사용량을 KB 단위로 출력한다.
- -m : 디스크 사용량을 MB 단위로 출력한다.
- -h : 디스크 사용량을 알기 쉬운 단위(GB, MB, KB 등)로 출력한다.
- -t 파일 시스템 종류 : 지정한 파일 시스템 종류에 해당하는 디스크 사용량을 출력한다.
- -T : 파일 시스템의 종류도 출력한다.

사용 예 df df -h

05 디스크 관리

df 명령만 사용하는 경우

```
user1@myubuntu:~$ df
Filesystem                1K-blocks      Used Available Use% Mounted on
/dev/sda1                  19478204   3482424  14983300  19% /
none                             4         0         4   0% /sys/fs/cgroup
udev                        504216         4    504212   1% /dev
tmpfs                       102588      1144    101444   2% /run
none                          5120         0      5120   0% /run/lock
none                        512920       152    512768   1% /run/shm
none                        102400        36    102364   1% /run/user
/dev/sr0                    916480    916480         0 100% /media/user1/
Ubuntu
13.10 i386
/dev/mapper/grp1-mylvm1     983448      1248    915028   1% /mnt/lvm
/dev/mapper/grp2-mylvm2     806288     17212    748116   3% /mnt/lvm2
/dev/mapper/grp2-mylvm3     186307      1550    170995   1% /mnt/lvm3
user1@myubuntu:~$
```

df 명령으로 출력되는 항목

- 파일 시스템 장치명
- 파일 시스템의 사용량
- 사용량을 퍼센트로 표시
- 파일 시스템의 전체 용량
- 파일 시스템의 사용 가능한 남은 용량
- 마운트 포인트

05 디스크 관리

■ 파일 시스템 사용량을 이해하기 쉬운 단위로 표시하기 : -h 옵션

- 각 파일 시스템별로 이해하기 쉬운 단위로 사용량을 표시

```
user1@myubuntu:~$ df -h
Filesystem                Size  Used  Avail  Use%  Mounted on
/dev/sda1                  19G  3.4G  15G  19%  /
none                      4.0K  0  4.0K  0%  /sys/fs/cgroup
udev                      493M  4.0K  493M  1%  /dev
tmpfs                     101M  1.2M  100M  2%  /run
none                      5.0M  0  5.0M    0%  /run/lock
none                      501M  152K  501M    1%  /run/shm
none                      100M  36K  100M    1%  /run/user
/dev/sr0                  895M  895M  0  100%  /media/user1/Ubuntu 13.10 i386
/dev/mapper/grp1-mylvm1  961M  1.3M   894M  1%  /mnt/lvm
/dev/mapper/grp2-mylvm2  788M  17M   731M  3%  /mnt/lvm2
/dev/mapper/grp2-mylvm3  182M  1.6M   167M  1%  /mnt/lvm3
user1@myubuntu:~$
```

05 디스크 관리

- **파일 시스템의 종류 정보 출력하기 : -T 옵션**

```
user1@myubuntu:~$ df -Th
Filesystem  Type  Size  Used  Avail  Use%  Mounted on
/dev/sda1   ext4  19G  3.4G     15G    19%  /
none                    tmpfs  4.0K  0    4.0K     0%  /sys/fs/cgroup
udev                    devtmpfs  493M  4.0K    493M     1%  /dev
tmpfs                   tmpfs  101M  1.2M    100M     2%  /run
none                    tmpfs  5.0M  0    5.0M     0%  /run/lock
none                    tmpfs  501M  152K    501M     1%  /run/shm
none                    tmpfs  100M  36K    100M     1%  /run/user
/dev/sr0                iso9660  895M  895M  0  100%  /media/user1/Ubuntu
13.10 i386
/dev/mapper/grp1-mylvm1 ext4  961M  1.3M  894M     1%  /mnt/lvm
/dev/mapper/grp2-mylvm2 ext3  788M  17M  731M     3%  /mnt/lvm2
/dev/mapper/grp2-mylvm3 ext4  182M  1.6M  167M     1%  /mnt/lvm3
user1@myubuntu:~$
```

05 디스크 관리

■ 디렉터리나 사용자별 디스크 사용량 확인하기 : du

du	
기능	디스크의 사용 공간에 대한 정보를 출력한다.
형식	du [옵션] [디렉터리]
옵션	-s : 특정 디렉터리의 전체 사용량을 출력한다. -h : 디스크 사용량을 알기 쉬운 단위(GB, MB, KB 등)로 출력한다.
사용 예	du du -s ~user1

■ du 명령만 사용하는 경우: 현재 디렉터리의 디스크 사용량을 출력

```
user1@myubuntu:~$ pwd
/home/user1
user1@myubuntu:~$ du
4       ./바탕화면
4       ./공개
4       ./비디오
8       ./.gconf/apps/nm-applet
8       ./.gconf/apps/gnome-terminal/profiles/Default
12      ./.gconf/apps/gnome-terminal/profiles
(생략)
17320   ./.mozilla
40388   .
user1@myubuntu:~$
```

05 디스크 관리

- **전체 디스크 사용량 출력하기 : -s 옵션**

```
user1@myubuntu:~$ du -s
40388   .
user1@myubuntu:~$ du -s /etc
du: '/etc/polkit-1/localauthority' 디렉터리를 읽을 수 없음: 허가 거부
du: '/etc/lvm/archive' 디렉터리를 읽을 수 없음: 허가 거부
du: '/etc/lvm/backup' 디렉터리를 읽을 수 없음: 허가 거부
du: '/etc/ssl/private' 디렉터리를 읽을 수 없음: 허가 거부
du: '/etc/cups/ssl' 디렉터리를 읽을 수 없음: 허가 거부
12996   /etc
user1@myubuntu:~$
```

- **특정 사용자의 디스크 사용량 출력하기**

```
user1@myubuntu:~$ du -sh ~user1
40M     /home/user1
user1@myubuntu:~$
```

05 디스크 관리

파일 시스템 검사하고 복구하기

- 파일 시스템은 부적절한 시스템 종료나 전원의 불안정, 소프트웨어 오류, 하드웨어 오작동 등 다양한 이유로 손상될 수 있음
- 손상된 파일 시스템의 용량을 확인할 뿐만 아니라 파일 시스템의 상태를 점검하고 문제가 있을 때 복구해야함

fsck 명령으로 파일 시스템 검사하기

- inode 및 블록, 디렉터리, 파일 링크 등을 검사하고 필요시 복구 작업도 수행

fsck

기능 리눅스의 파일 시스템을 검사한다.

형식 fsck [옵션] 장치명

옵션
-f : 강제로 검사한다.
-b 슈퍼블록 : 슈퍼블록으로 지정한 백업 슈퍼블록을 사용한다.
-y : 모든 질문에 yes로 대답하도록 한다.
-a : 파일 시스템 검사에서 문제가 발생했을 때 자동으로 복구한다.

사용 예 fsck /dev/sdb1 fsck -f /dev/sdb1

05 디스크 관리

fsck 명령으로 파일 시스템 검사하기

- 일반적인 파일 시스템 검사

```
user1@myubuntu:~$ sudo fsck /dev/sdd1
fsck from util-linux 2.20.1
e2fsck 1.42.8 (20-Jun-2013)
/dev/sdd1: clean, 11/76912 files, 11777/307200 blocks
user1@myubuntu:~$
```

- 파일 시스템 강제 검사

```
user1@myubuntu:~$ sudo fsck -f /dev/sdd1
fsck from util-linux 2.20.1
e2fsck 1.42.8 (20-Jun-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdd1: 11/76912 files (0.0% non-contiguous), 11777/307200 blocks
user1@myubuntu:~$
```

05 디스크 관리

fsck 명령으로 파일 시스템 검사하기

- 파일 시스템 종류를 지정해 검사

```
user1@myubuntu:~$ sudo fsck.ext4 /dev/sdd1
e2fsck 1.42.8 (20-Jun-2013)
/dev/sdd1: clean, 11/76912 files, 11777/307200 blocks
user1@myubuntu:~$
```

05 디스크 관리

e2fsck 명령으로 파일 시스템 검사하기

e2fsck

기능	리눅스의 확장 파일 시스템(ext2, ext3, ext4)을 검사한다.
형식	e2fsck [옵션] 장치명
옵션	-f : 강제로 검사한다. -b 슈퍼블록 : 슈퍼블록으로 지정한 백업 슈퍼블록을 사용한다. -y : 모든 질문에 yes로 대답하도록 한다. -j ext3/ext4 : ext3나 ext4 파일 시스템을 검사할 때 지정한다.
사용 예	e2fsck /dev/sdb1 e2fsck -f /dev/sdb1

- 일반적인 파일 시스템 검사

```
user1@myubuntu:~$ sudo e2fsck /dev/sdd1
e2fsck 1.42.8 (20-Jun-2013)
/dev/sdd1: clean, 11/76912 files, 11777/307200 blocks
user1@myubuntu:~$
```

05 디스크 관리

e2fsck 명령으로 파일 시스템 검사하기

- 파일 시스템 강제 검사

```
user1@myubuntu:~$ sudo e2fsck -f /dev/sdd1
e2fsck 1.42.8 (20-Jun-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdd1: 11/76912 files (0.0% non-contiguous), 11777/307200 blocks
user1@myubuntu:~$
```

05 디스크 관리

배드 블록 검사하기 : badblocks

badblocks

기능	장치의 배드 블록을 검색한다.
형식	badblocks [옵션] 장치명
옵션	-v : 검색 결과를 자세하게 출력한다. -o 출력 파일 : 검색한 배드 블록 목록을 지정한 출력 파일에 저장한다.
사용 예	badblocks -v /dev/sdb1 badblocks -v -o bad.out /dev/sdb1

- 배드 블록 검색하기

```
user1@myubuntu:~$ sudo badblocks -v /dev/sdd1
Checking blocks 0 to 307199
Checking for bad blocks (read-only test):
done
Pass completed, 0 bad blocks found. (0/0/0 errors)
user1@myubuntu:~$
```

05 디스크 관리

배드 블록 검사하기 : badblocks

- 검색 결과를 파일로 저장하기 : -o 옵션

```
user1@myubuntu:~$ sudo badblocks -v -o bad.out /dev/sdd1
Checking blocks 0 to 307199
Checking for bad blocks (read-only test):
done
Pass completed, 0 bad blocks found. (0/0/0 errors)
user1@myubuntu:~$
```

05 디스크 관리

■ 백업 수퍼블록을 이용해 파일 시스템 복구하기

- 파일 시스템의 기본 수퍼블록에 문제가 있으면 해당 파일 시스템을 사용할 수 없음
- 이 경우 백업 수퍼블록 중 하나를 사용하여 파일 시스템을 복구
- 백업 수퍼블록의 위치 파악하기 : dumpe2fs

dumpe2fs

기능 파일 시스템의 정보를 출력한다.

형식 dumpe2fs 장치명

사용 예 dumpe2fs /dev/sdb1

```
user1@myubuntu:~$ sudo dumpe2fs /dev/sdd1
dumpe2fs 1.42.8 (20-Jun-2013)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          c6cf16fe-9625-4a60-a7cd-d91ac4e8ceb7
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      ext_attr resize_inode dir_index filetype sparse_super
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
(생략)
```

05 디스크 관리

백업 수퍼블록을 이용해 파일 시스템 복구하기

- 수퍼블록에 관한 정보만 추출

```
user1@myubuntu:~$ sudo dumpe2fs /dev/sdd1 | grep superblock
dumpe2fs 1.42.8 (20-Jun-2013)
  Primary superblock at 1, Group descriptors at 2-3
  Backup superblock at 8193, Group descriptors at 8194-8195
  Backup superblock at 24577, Group descriptors at 24578-24579
  Backup superblock at 40961, Group descriptors at 40962-40963
  Backup superblock at 57345, Group descriptors at 57346-57347
  Backup superblock at 73729, Group descriptors at 73730-73731
  Backup superblock at 204801, Group descriptors at 204802-204803
  Backup superblock at 221185, Group descriptors at 221186-221187
user1@myubuntu:~$
```

05 디스크 관리

백업 수퍼블록을 이용해 파일 시스템 복구하기

- 파일 시스템 복구하기 : -b 옵션

```
user1@myubuntu:~$ sudo e2fsck -b 8193 -y /dev/sdd1
e2fsck 1.42.8 (20-Jun-2013)
/dev/sdd1 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdd1: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sdd1: 11/76912 files (0.0% non-contiguous), 11777/307200 blocks
user1@myubuntu:~$
```

우분투 리눅스

시스템 & 네트워크