File System & Disk Mangement

Linux File System 종류

- ❖ 디스크 기반 파일 시스템
  - ✓ 개요
    - 파일과 디렉토리를 관리하려면 파일 시스템이 필요
    - 리눅스는 초기에 Minix File System(MFS)을 이용했으나 ext 파일 시스템으로 알려진 리눅스 고유의 파일 시스템을 만들어 사용
    - ext 파일 시스템은 버전이 계속 업그레이드되면서 현재는 ext4까지 제공하고 있는데 대용량 파일 시스템을 위해 실리콘 그래픽스가 개발한 XFS 파일 시스템도 도입됨

Linux File System 종류

- ❖ 디스크 기반 파일 시스템
  - ✓ 종류
    - ● ext4
      - ◆ ext4 파일 시스템은 1EB(1EB=1,024 X 1,024TB) 이상의 볼륨과 16TB 이상의 파일을 지원
      - ◆ ext2 및 ext3과 호환성을 유지하고 있으며 ext3에서는 서브 디렉토리의 수가 32,000개로 제한되었으나 ext4에서는 64,000개로 늘어났으며 온라인 조각 모음 기능도 지원
    - ● XFS
      - ◆ XFS 파일 시스템(extended File System)은 1993년 실리콘 그래픽스가 개발한 고성능 저널링 파일 시스템으로 2000년 5월 GNU GPL로 공개되었고 2001년 리눅스에 이식되었고 현재 대부분의 리눅스 배포판에서 지원하고 있음
      - ◆ XFS는 64비트 파일 시스템으로 최대 16EB까지 지원
      - ◆ 우분투에서 XFS 파일 시스템을 이용하려면 xfsprogs 패키지가 설치되어 있어야 함

Linux File System 종류

- 리눅스에서 지원하는 다양한 파일 시스템
  - ✓ msdos: MS-DOS 파티션을 사용하기 위한 파일 시스템
  - ✓ iso9660: CD-ROM, DVD의 표준 파일 시스템이며 읽기 전용으로 사용
  - ✓ nfs(Network File System): 원격 서버의 디스크를 연결할 때 사용
  - ✓ ufs(Unix File System): 유닉스의 표준 파일 시스템
  - ✓ vfat: 윈도 95, 98. NT를 지원하기 위한 파일 시스템
  - ✓ hpfs: HPFS(High Performance File System - IBM이 OS/2 운영체제를 위해 만든 파일 시스템)를 지원하기 위한 파일 시스템
  - ✓ ntfs: 윈도의 NTFS를 지원하기 위한 파일 시스템
  - ✓ sysv: 유닉스 시스템 V를 지원하기 위한 파일 시스템
  - ✓ hfs: 맥 컴퓨터의 hfs 파일 시스템을 지원하기 위한 파일 시스템

Linux File System 종류

- 특수 용도의 가상 파일 시스템
  - ✓ swap
    - swap 영역을 관리하기 위한 swap 파일 시스템
    - RAM을 보완하여 부족한 메모리를 디스크 공간으로 확장하고 시스템이 안정적으로 동작하도록 돕는 역할
  - ✓ tmpfs
    - Temporary File System으로 메모리에 임시 파일을 저장하기 위한 파일 시스템이며 시스템이 재시작 할 때 기존 내용이 소멸됨
    - /tmp 디렉토리가 대표적
  - ✓ ramfs
    - 디스크 대신 메모리를 저장 공간처럼 활용하는 가상 파일 시스템
    - 임베디드 장치 나 테스트 용도로 사용하는데 tmpfs 와 다른 점은 tmpfs는 메모리를 사용하지만 필요 시 swap을 이용
  - ✓ rootfs
    - Root File System으로 /디렉토리
    - 시스템 초기화 및 관리에 필요한 내용을 저장

Linux File System 종류

- 특수 용도의 가상 파일 시스템
  - ✓ proc
    - proc 파일 시스템으로 /proc 디렉토리
    - 커널의 현재 상태를 나타내는 파일을 소유
    - 리눅스의 가상 파일 시스템(Virtual File System) 으로 커널과 프로세스 관련 정보를 사용자 공간에서 확인할 수 있도록 제공
    - 실제 디스크에 저장된 파일이 아니라 메모리에 존재하는 인터페이스
    - 주요 내용
      - /proc/cpuinfo → CPU 정보
      - /proc/meminfo → 메모리 사용량
      - /proc/uptime → 시스템 부팅 후 경과 시간
      - /proc/loadavg → 시스템 부하 평균
      - /proc/[PID]/status → 특정 프로세스 상태
      - /proc/sys/net/ipv4/ip_forward → 패킷 포워딩 여부 (0/1 변경 가능)

Linux File System 종류

- 현재 지원하는 파일 시스템 확인
  - ✓ cat /proc/filesystems

```
nodev   sysfs
nodev   tmpfs
nodev   bdev
nodev   proc
nodev   cgroup
nodev   cgroup2
nodev   cpuset
nodev   devtmpfs
nodev   configfs
nodev   debugfs
nodev   tracefs
nodev   securityfs
nodev   sockfs
nodev   bpf
nodev   pipefs
nodev   ramfs
nodev   hugetlbfs
nodev   devpts
 ext3
 ext2
 ext4
 squashfs
```

Linux File System 구조

- ❖ 리눅스 파일 시스템
  - ✓ 유닉스 운영체제에서 유래된 공통의 개념
    - 파일은 inode로 관리
    - 디렉토리는 단순히 파일 목록을 가지고 있는 파일
    - 특수 파일을 통해 장치에 접근

Linux File System 구조

- ext4 파일 시스템의 구조
  - ✓ 개요
    - ext4 파일 시스템은 효율적으로 디스크를 사용하기 위해 저장 장치를 논리적인 블록의 집합(블록 그룹) 으로 구분
    - 일반적으로 블록은 4KB이고 실제 크기는 시스템의 설정에 따라 달라질 수 있음
    - 블록 그룹의 개수는 장치의 크기를 블록 그룹의 크기로 나눈 값

Linux File System 구조

- ext4 파일 시스템의 구조
  - ✓ ext4 파일 시스템의 블록 그룹 유형
    - 블록 그룹 0: 파일 시스템의 첫 번째 블록 그룹으로 특별하게 그룹 0 패딩과 슈퍼 블록, 그룹 디스크립터를 가지고 있음
    - 블록 그룹 a: 파일 시스템에서 첫 번째 블록 그룹이 아닌 블록 그룹으로 그룹 0 패딩이 없으나 슈퍼 블록과 그룹 디스크립터의 복사본을 가지고 있음
    - 블록 그룹 b: 파일 시스템에서 첫 번째 블록 그룹이 아닌 블록 그룹으로 그룹 0 패딩, 슈퍼 블록, 그룹 디스크립터가 없고 바로 데이터 블록 비트맵으로 시작

Linux File System 구조

- ext4 파일 시스템의 구조
  - ✓ 파일 시스템을 구성하는 요소
    - 그룹 0 패딩
      - 블록 그룹 0의 첫 1024B는 x86 부트 섹터와 부가 정보를 저장하기 위한 것
    - 슈퍼 블록
      - 슈퍼 블록에는 파일 시스템과 관련된 다양한 정보를 저장하는데 문제가 생기면 전체 파일 시스템을 사용할 수 없게 되기 때문에 슈퍼 블록을 다른 블록 그룹에 복사해놓고 블록 그룹 0의 슈퍼 블록을 읽을 수 없을 경우 복사본을 사용하여 복구
      - 주요 정보
        - 전체 inode의 개수 와 free inode의 개수
        - free block 의 개수
        - 첫 번째 데이터 블록의 주소
        - 그룹 당 블록의 개수 와 전체 블록의 개수
        - 파일 시스템의 상태
        - 블록의 크기
        - 마운트 시간
        - 그룹 디스크립터의 크기

Linux File System 구조

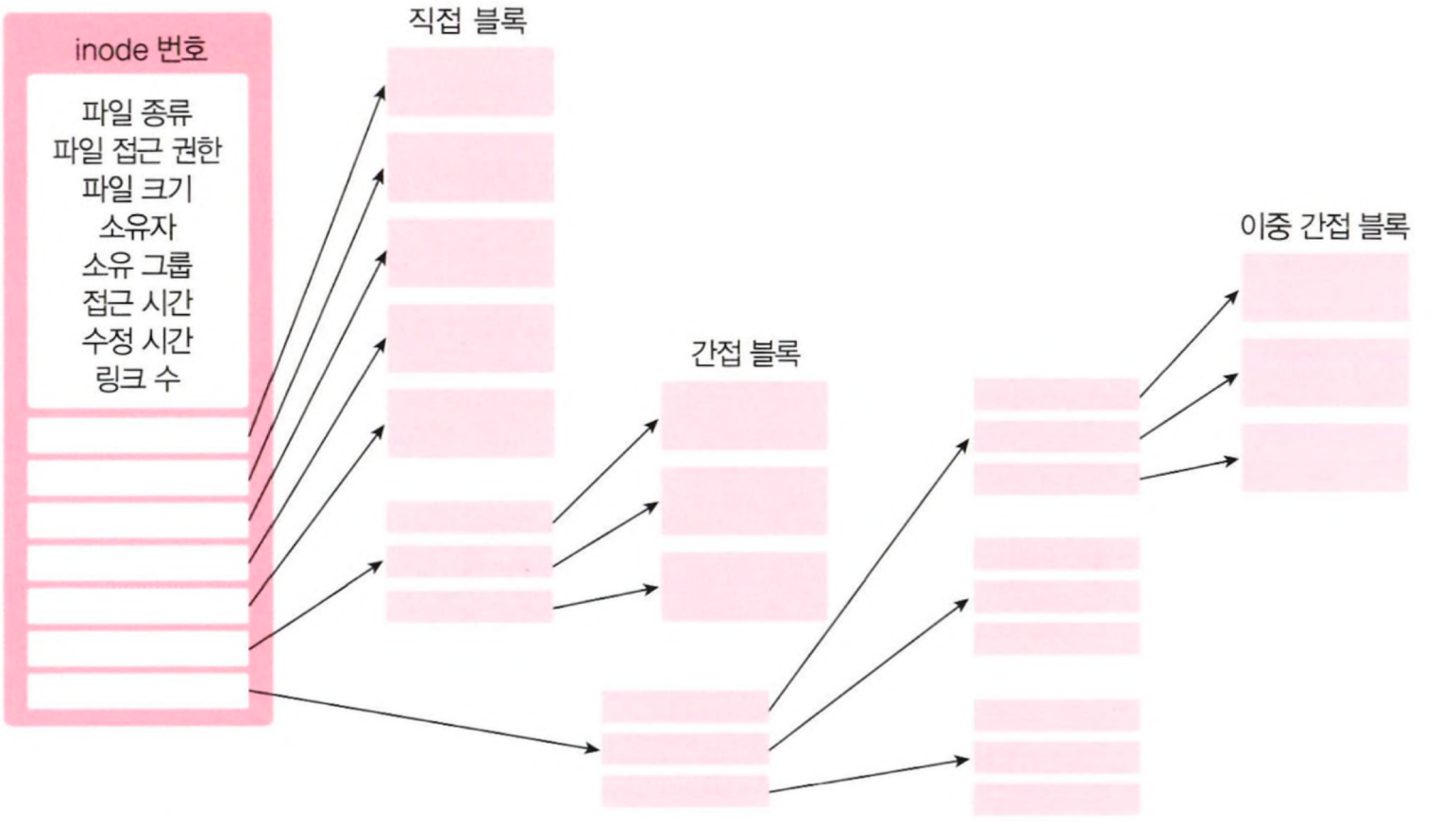
- ext4 파일 시스템의 구조
  - ✓ 그룹 디스크립터 와 GDT 예약 블록
    - 그룹 디스크립터도 블록 그룹 0에 있는 것으로 슈퍼 블록의 다음에 위치
    - 그룹 디스크립터에는 다음과 같은 정보를 저장
      - 블록 비트맵의 주소
      - inode 비트맵의 주소
      - inode 테이블의 주소
      - 할당되지 않은 블록의 개수
      - 할당되지 않은 inode의 개수
      - 디렉토리의 개수
      - 블록 비트맵
      - inode 비트맵 체크섬
    - 그룹 디스크립터도 슈퍼블록과 함께 다른 블록 그룹에 복사되어 블록 그룹 0에 문제가 있을 때 복구하는 데 사용
    - GDT 예약 블록은 그룹 디스크립터의 확장을 위한 예비 공간

Linux File System 구조

- ext4 파일 시스템의 구조
  - ✓ 데이터 블록 비트맵과 inode 비트맵
    - 데이터 블록 비트맵은 블록 그룹에 포함된 데이터 블록의 사용 여부를 확인하는 데 쓰이며 inode 비트맵은 inode 테이블의 항목이 사용 중인지를 표시
    - 비트맵에서 각 데이터 블록과 inode 테이블의 항목은 1비트로 표현
  - ✓ inode 테이블과 데이터 블록
    - 일반적인 유닉스처럼 리눅스에서도 inode에 파일 정보를 저장
    - 데이터 블록에는 실제 데이터가 저장
    - 일반 파일은 데이터 블록에 파일 내용을 저장하고 디렉토리는 데이터 블록에 해당 디렉토리에 있는 파일이나 서브 디렉토리의 정보(이름, inode)를 저장

Linux File System 구조

- Inode 구조
  - ✓ 전통적인 구조

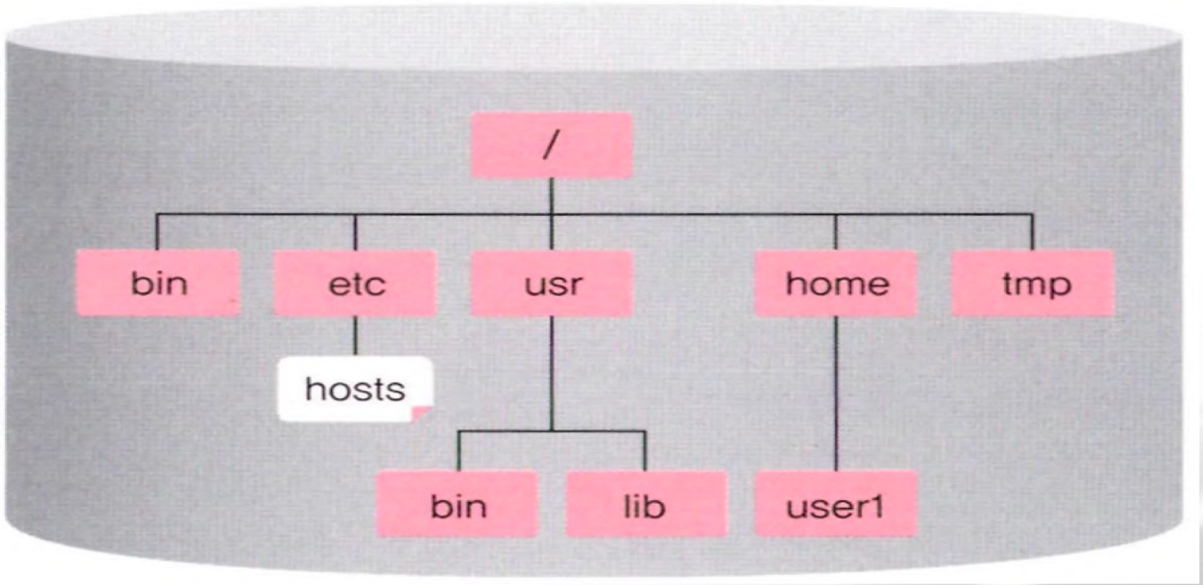


Linux File System 구조

- inode 구조
  - ✓ inode는 크게 두 부분 즉 파일 정보를 저장하는 부분과 실제 파일 내용이 저장된 데이터 블록의 주소를 저장하는 부분으로 나뉨
  - ✓ inode에 저장되는 파일 정보는 파일 종류, 파일 접근 권한, 파일 크기, 소유자, 접근 및 수정 시간 등으로 사용자가 ls – l 명령으로 확인하는 정보로 ls -l 명령은 inode에 저장된 파일 정보를 읽어서 출력하는 것
  - ✓ inode에서 데이터 블록의 주소를 저장하는 부분은 직접 블록, 간접 블록, 이중 간접 블록으로 구분
  - ✓ 직접 블록은 데이터 블록에 대한 주소를 직접 가지고 있고 간접 블록과 이중 간접 블록에는 데이터 블록에 대한 주소를 가지고 있는 블록의 주소가 저장됨
  - ✓ 데이터 블록의 크기는 시스템 설정에 따라 1~8KB까지 지정할 수 있음

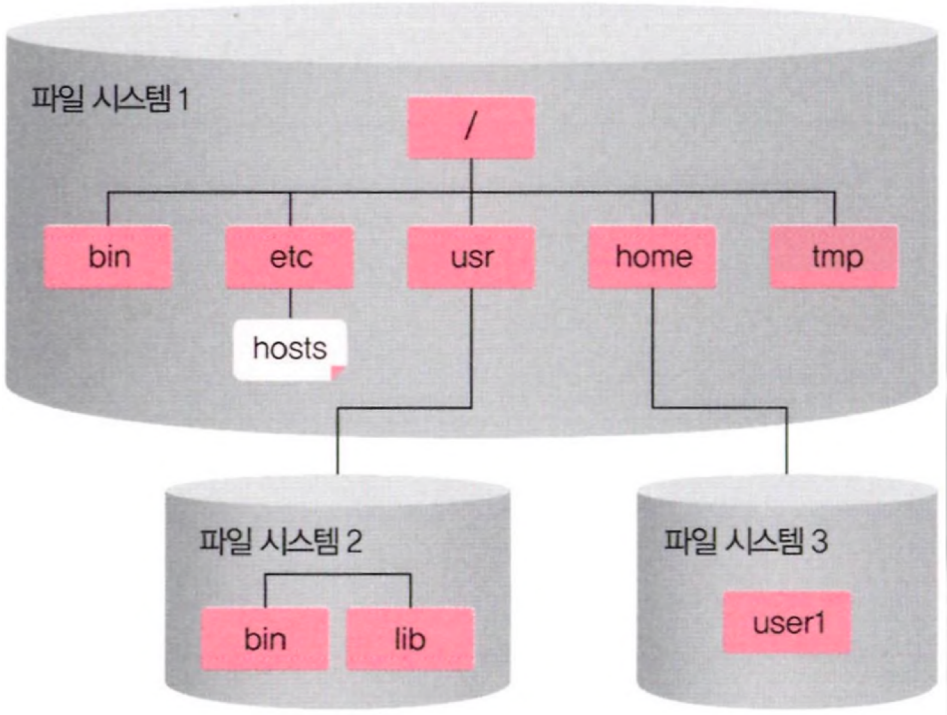
Linux File System 구조

- ❖ 파일 시스템 과 디렉토리 계층 구조
  - ✓ 하나의 파일 시스템으로 구성
    - ● 하나의 파일 시스템으로 구성할 경우 / 디렉토리에 파일 시스템을 연결



Linux File System 구조

- ❖ 파일 시스템 과 디렉토리 계층 구조
  - ✓ 여러 파일 시스템으로 구성: 여러 파일 시스템으로 구분하여 구성할 수 있음
    - 파일 시스템 1을 /디렉토리에 파일 시스템 2를 /usr 디렉토리에 파일 시스템 3을 /home 디렉토리에 연결
    - /디렉토리에 연결된 파일 시스템을 루트 파일 시스템이라하고 파일 시스템으로 나누어 디렉토리 계층 구조를 구성할 경우 일부 파일 시스템에 문제가 생기더라도 다른 파일 시스템의 파일은 안전



Linux File System Mount

- 파일 시스템 마운트
  - ✓ 개요
    - 파일 시스템이 디렉토리 계층 구조와 연결되지 않으면 사용자가 해당 파일 시스템에 접근할 수 없음
    - 파일 시스템을 디렉토리 계층 구조의 특정 디렉토리와 연결하는 것이 마운트
    - 이전 그림에서 파일 시스템 3은 /home 디렉토리에 마운트 한 것이고 파일 시스템 2는 /usr 디렉토리에 마운트 한 것
  - ✓ 마운트 포인트
    - 디렉토리 계층 구조에서 파일 시스템이 연결되는 디렉토리가 마운트 포인트
    - 이전 그림에서는 /usr, /home 디렉토리가 마운트 포인트인데 루트 파일 시스템을 마운트 한 / 디렉토리도 마운트 포인트

Linux File System Mount

- 파일 마운트 설정 파일
  - ✓ 리눅스에서 시스템이 부팅될 때 자동으로 파일 시스템이 마운트되게 하려면 /etc/fstab 파일에 관련 사항을 설정
  - ✓ /etc/fstab 파일의 기능
    - 리눅스 시스템은 부팅할 때 이 파일을 읽고 설정 내용에 따라 파일 시스템을 자동으로 마운트하며 이 파일에 오류가 있으면 시스템 부팅이 중지될 수도 있음
    - /etc/fstab 파일에는 파일 시스템의 장치 이름과 마운트 포인트 그리고 마운트 할 때 설정할 옵션을 지정
  - ✓ /etc/fstab 파일의 구조
    - 장치 이름
    - 마운트 포인트
    - 파일 시스템의 종류
    - 옵션
    - 덤프 관련 설정
    - 파일 점검 옵션

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ 파일 마운트 설정 파일
    - ● 실제 내용
      - ◆ 장치 이름: /dev/disk/by-id/dm-uuid-LVM-OPh3q8uzO0MHMG4KhpE91917VA5rShWqiCYY7t5z4DocjUMK6DQK2eo48Y461EHw
      - ◆ 마운트 포인트: /
      - ◆ 파일 시스템 종류: ext4
      - ◆ 옵션: defaults
      - ◆ 덤프 관련 설정: 0
        - ▪ 0: dump 명령으로 덤프되지 않은 파일 시스템
        - ▪ 1: 데이터 백업 등을 위해 dump 명령의 사용이 가능한 파일 시스템
      - ◆ 파일 점검 옵션: 1
        - ▪ 0: 부팅할 때 fsck 명령으로 파일 시스템을 점검하지 않도록 설정
        - ▪ 1: 루트 파일 시스템
        - ▪ 2: 루트 파일 시스템 이외의 파일 시스템

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ 파일 마운트 설정 파일
    - ● 실제 내용
      - ◆ 옵션 종류
        - defaults: 일반적인 파일 시스템에 지정하는 속성으로 rw, nouser, auto, exec, suid 속성을 모두 포함
        - auto: 부팅 시 자동으로 마운트
        - exec: 실행 파일이 실행되는 것을 허용
        - suid: setuid, setgid의 사용을 허용
        - ro: 읽기 전용 파일 시스템
        - rw: 읽기 및 쓰기가 가능한 파일 시스템
        - user: 일반 사용자도 마운트가 가능하다
        - nouser: 일반 사용자의 마운트가 불가능하고 root 만 마운트 가능
        - noauto: 부팅 시 자동으로 마운트하지 않음
        - noexec: 실행 파일이 실행되는 것을 허용하지 않음
        - nosuid: setuid, setgid 의 사용을 금지
        - usrquota: 사용자 별로 디스크 쿼터 설정이 가능
        - grpquota: 그룹 별로 디스크 쿼터 설정이 가능

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ 마운트 관련 명령
    - ● 장치를 디렉토리와 연결하려면 mount 명령을 사용
      - ◆ 형식: mount [옵션] [장치명 마운트 포인트]
      - ◆ 옵션
        - ▪ -t 파일 시스템 종류: 파일 시스템 종류를 지정
        - ▪ -o 마운트 옵션: 마운트 옵션을 지정
        - ▪ -f: 마운트를 할 수 있는지 점검만 수행
        - ▪ -r: 읽기만 가능하게 마운트(-o ro와 동일)
      - ◆ 사용 예
        - ▪ mount
        - ▪ mount /dev/sdb1 /
        - ▪ mount –t iso9660 /dev/cdrom /mnt/cdrom

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ 마운트 관련 명령
    - ● 장치를 디렉토리와 연결을 해제하고자 하는 경우에는 unmount 명령을 사용
      - ◆ 형식: umount [옵션] 장치명 또는 마운트 포인트
      - ◆ 옵션
        - ▪ -t 파일 시스템 종류: 파일 시스템 종류를 지정
      - ◆ 사용 예
        - ▪ unmount /dev/sdb1
        - ▪ unmount /mnt

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ mount 명령만 사용하는 경우
    - ● $ mount

      sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)

      proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)

      udev on /dev type devtmpfs (rw,nosuid,relatime,size=944028k,nr_inodes=236007,mode=755,inode64)

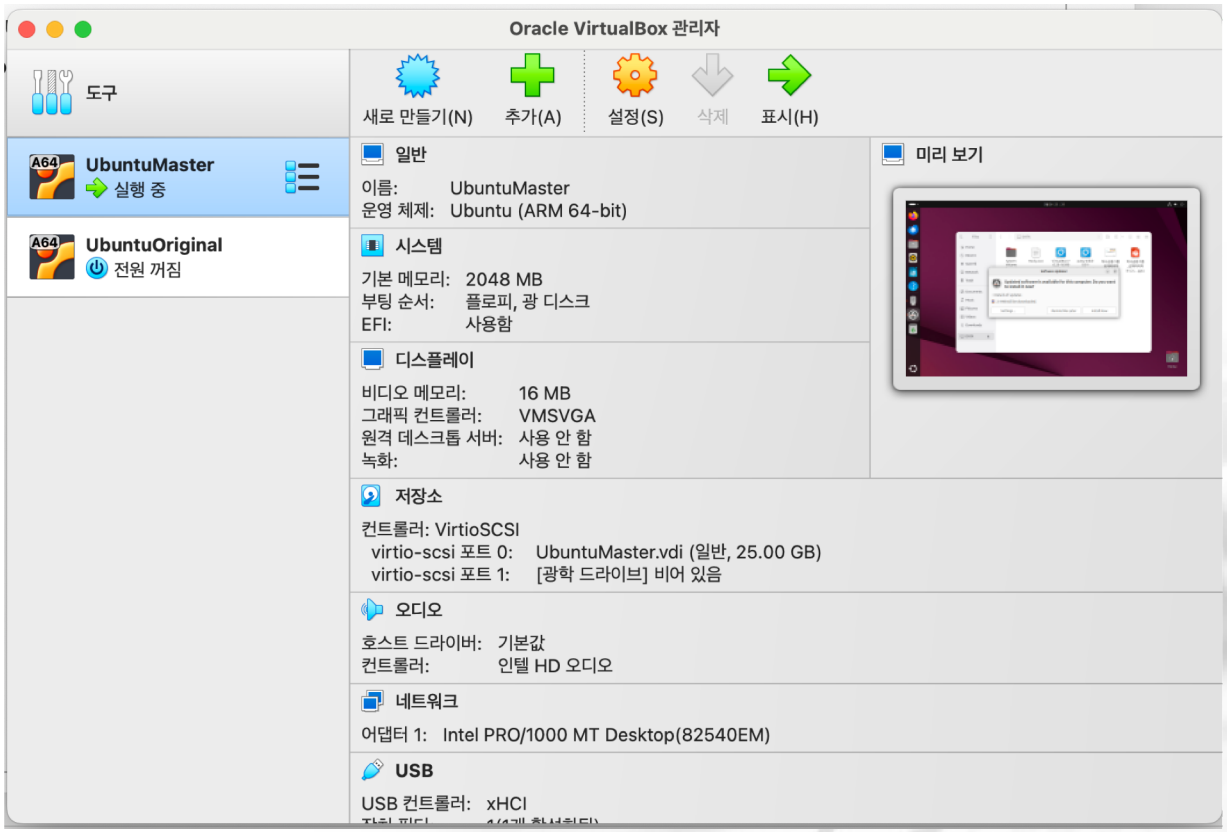
      - ◆ mount 명령으로 출력되는 정보는 /etc/mtab 파일의 내용과 동일
      - ◆ /etc/mtab 파일에는 현재 시스템에 마운트되어 있는 파일 시스템의 정보가 저장되어 있음
      - ◆ /proc/mounts에 대한 심볼릭 링크로 읽기 전용 파일이므로 내용을 수정할 수 없음
      - ◆ /etc/mtab 구성 항목
        - ▪ 장치명
        - ▪ 마운트 포인트
        - ▪ 파일 시스템의 종류
        - ▪ 옵션
        - ▪ 사용하지 않는 항목 두 개(0 0)

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ 장치 연결
    - ● $ mount /dev/sdb1 /mnt
      - ◆ /dev/sdb1은 디스크 장치의 이름이고 /mnt는 마운트 포인트
      - ◆ 다양한 장치 마운트의 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 -f vfat /dev/hdc /mnt
        - ▪ USB 메모리
          - mount /dev/sdc1 /mnt: 리눅스용 USB 메모리의 경우
          - mount -f 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 -f nfs 서버 주소:/NFS 서버 측 디렉토리 /mnt

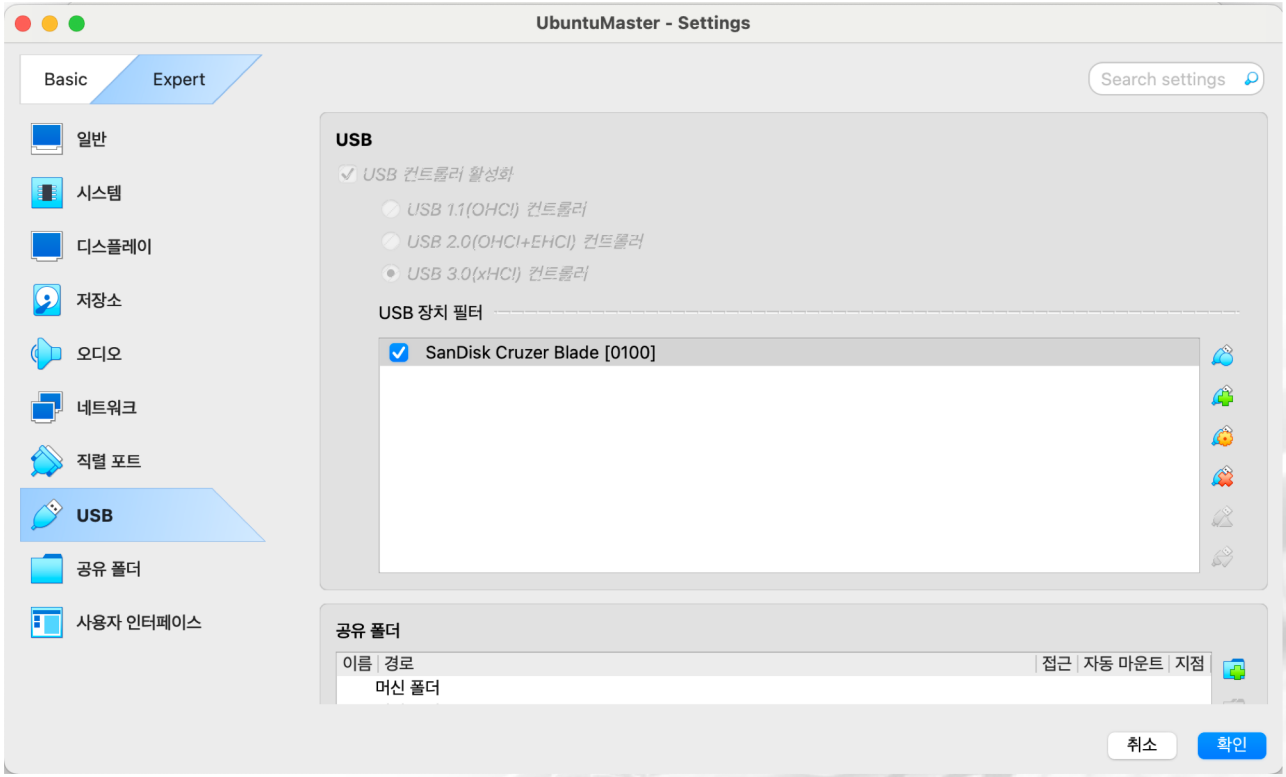
Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ VirtualBox를 선택하고 [설정]을 선택



Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ [USB]를 선택하고 오른쪽 화면에 [USB 장치 필터]를 추가해서 사용



Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 디스크 장치 확인: sudo fdisk -l

```
Disk /dev/sdb: 29.25 GiB, 31406948352 bytes, 61341696 sectors
Disk model: Cruzer Blade
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x120cde0e

Device     Boot Start      End  Sectors  Size Id Type
/dev/sdb1  *     2048 61341695 61339648 29.2G  c W95 FAT32 (LBA)
```

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 파티션 생성
      - ◆ ~$ sudo fdisk /dev/sdb

        Welcome to fdisk (util-linux 2.36.1).

        Changes will remain in memory only, until you decide to write them.

        Be careful before using the write command.
      - ◆ 사용할 수 있는 명령 확인 - Command (m for help) : m

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 파티션 생성
        - ▪ 파티션 생성: Command (m for help) n
        - ▪ 기존 파티션 삭제: Command (m for help) d
        - ▪ 파티션 생성: Command (m for help) n
        - ▪ 파티션 종류 선택

          Partition type

          p primary (0 primary, 0 extended, 4 free)

          e extended (container for logical partitions)

          Select (default p) : p
        - ▪ 파티션 번호: Command (m for help) 1
        - ▪ First sector (2048-15532031, default 2048): Enter
        - ▪ Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-61341695, default 61341695): Enter

        Created a new partition 1 of type 'Linux' and of size 29.2 GiB.

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 파티션 생성
        - ▪ 확인: p

```
Disk /dev/sdb: 29.25 GiB, 31406948352 bytes, 61341696 sectors
Disk model: Cruzer Blade
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x120cde0e
Device     Boot Start      End  Sectors  Size Id Type
/dev/sdb1        2048 61341695 61339648 29.2G 83 Linux
Filesystem/RAID signature on partition 1 will be wiped.
```

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 파티션 생성
        - ▪ w를 입력하여 설정한 파티션 정보를 파티션 테이블에 기록하는데 파티션 테이블에 기록되면 이전에 있던 정보가 없어지고 USB 메모리에 파티션이 생성됨

          The partition table has been altered.

          Calling ioctl() to re-read partition table.

          Syncing disks.

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 포맷 및 파일 시스템 생성: 사용 준비
        - ▪ sudo mke2fs -t ext4 /dev/sdb1

mke2fs 1.47.2 (1-Jan-2025)

Creating filesystem with 7667456 4k blocks and 1916928 inodes

Filesystem UUID: 69ff91ce-9830-4469-9641-8be2b2343636

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,

4096000

Allocating group tables: done

Writing inode tables: done

Creating journal (32768 blocks): Enter

done

Writing superblocks and filesystem accounting information: done

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ USB 파일 시스템 마운트
        - ▪ sudo mount /dev/sdb1 /mnt
        - ▪ 확인

          $ cd /mnt

          /mnt$ ls

          lost+found
        - ▪ 파일 복사

          /mnt$ sudo cp /etc/hosts .

          /mnt$ ls

          hosts lost+found

Linux File System Mount

- ❖ 파일 시스템 마운트
  - ✓ Virtual Box에서 디스크 추가
    - ● VirtualBox에서 USB 사용
      - ◆ 마운트 해제
        - ▪ cd
        - ▪ sudo umount /mnt

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 디스크 장치의 이름과 파티션 표시
    - 리눅스에서는 IDE 컨트롤러에 연결된 디스크는 /dev/hd로 시작하는 이름을 SCSI나 SATA 컨트롤러에 장착된 디스크는 /dev/sd로 시작하는 이름을 사용했으나 최근에는 IDE, SCSI 등의 구분없이 모두 /dev/sd로 시작하는 이름을 사용
    - 컨트롤러에 연결되는 디스크의 순서에 따라 다음과 같이 알파벳이 추가
      - ◆ /dev/sda: 첫 번째 디스크
      - ◆ /dev/sdb: 두 번째 디스크
      - ◆ /dev/sdc: 세 번째 디스크
    - 일반적으로 IDE 컨트롤러에 장착된 디스크가 SCSI에 장착된 디스크보다 앞에 배정됨
    - 파티션은 디스크 장치 이름 뒤에 숫자를 붙여서 표시하는데 장치명이 /dev/sda라면 파티션의 이름을 다음과 같이 표시
      - ◆ /dev/sda: 첫 번째 디스크 전체를 의미하는 장치명
      - ◆ /dev/sda0: 디스크의 첫 번째 파티션
      - ◆ /dev/sda1: 디스크의 두 번째 파티션

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - 디스크를 파티션을 관리하는 작업을 위한 명령어
    - 형식

    fdisk [옵션] [장치명]
    - 옵션
      - ◆ -b 크기: 섹터 크기를 지정(512, 1024, 2048, 4096).
      - ◆ -l: 파티션 테이블을 출력

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - ● 내부 명령
      - ◆ a: 부팅 파티션을 설정
      - ◆ p: 파티션 테이블을 출력
      - ◆ b: BSD 디스크 라벨을 편집
      - ◆ q: 작업 내용을 저장하지 않고 종료
      - ◆ c: 도스 호환성을 설정
      - ◆ s: 새로운 빈 Sun 디스크 라벨을 생성
      - ◆ d: 파티션을 삭제
      - ◆ t: 파티션의 시스템 ID를 변경(파일 시스템 종류 변경)
      - ◆ l: 사용 가능한 파티션의 종류를 출력
      - ◆ u: 항목 정보를 변경 및 출력
      - ◆ m: 도움말을 출력
      - ◆ v: 파티션 테이블을 검사
      - ◆ n: 새로운 파티션을 추가
      - ◆ w: 파티션 정보를 디스크에 저장하고 종료
      - ◆ o: 새로운 빈 DOS 파티션을 생성
      - ◆ x: 실린더 개수 변경 등 전문가를 위한 부가적 기능

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - ● 파티션 정보 보기
      - ◆ sudo fdisk –l

```
Disk /dev/sda: 25 GiB, 26843545600 bytes, 52428800 sectors
Disk model: HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 0FE992DA-2D23-49E9-9E92-EAAD38C50FE2

Device       Start      End  Sectors  Size Type
/dev/sda1     2048  2203647  2201600    1G EFI System
/dev/sda2  2203648  6397951  4194304    2G Linux filesystem
/dev/sda3  6397952 52426751 46028800 21.9G Linux filesystem


Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - ● 파티션 만들기
      - ◆ sudo fdisk /dev/sdb

```
Welcome to fdisk (util-linux 2.40.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS (MBR) disklabel with disk identifier 0x675b7da4.

Command (m for help):
```

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - ● 파티션 만들기
      - ◆ sudo fdisk /dev/sdb

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-20971519, default 20971519):
+500M

Created a new partition 1 of type 'Linux' and of size 5 GiB.
```

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - ● 파티션 만들기
      - ◆ 동일한 방식으로 3개 더 생성

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ fdisk
    - ● 파티션 만들기
      - ◆ sudo fdisk /dev/sdb

```
Command (m for help): p
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x675b7da4

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1          2048 1026047 1024000  500M 83 Linux
/dev/sdb2       1026048 2050047 1024000  500M 83 Linux
/dev/sdb3       2050048 3074047 1024000  500M 83 Linux
/dev/sdb4       3074048 4098047 1024000  500M 83 Linux

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

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - 디스크의 파티션 작업이 끝나면 다음으로 파티션에 파일 시스템을 생성해야 함
    - 파티션이 디스크를 독립적인 영역으로 구분하는 작업이라면 파일 시스템 생성은 파티션에서 파일과 디렉토리를 관리하기 위한 구조를 만드는 것
    - 생성 명령은 mkfs 와 mke2fs
    - mkfs
      - ◆ 형식: mkfs [옵션] [장치명]
      - ◆ 옵션: -t 종류 형식으로 설정해서 파일 시스템의 종류를 지정(기본은 ext2)
      - ◆ mkfs 명령은 생성하는 파일 시스템의 종류에 따라 mkfs.ext2, mkfs.ext3, mkfs.ext4 명령도 제공하는데 이 명령들은 mkfs -t ext2, mkfs -t ext3, mkfs -t ext4와 동일하게 동작

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - mke2fs
      - ◆ 리눅스 확장 파일 시스템(ext2, ext3, ext4)을 생성
      - ◆ 형식: mke2fs [옵션] [장치명]
      - ◆ 옵션
        - -t 종류: 파일 시스템의 종류를 지정(기본값은 ext2)
        - -b: 블록 크기: 블록 크기를 바이트 수로 지정
        - -c: 배드 블록을 체크
        - -f 프래그먼트 크기: 프래그먼트 크기를 바이트 수로 지정
        - -i inode당 바이트 수: inode당 바이트 수를 지정(기본값은 4,096B)
        - -m 예약 블록 퍼센트: 슈퍼유저에게 예약해 둘 블록의 퍼센트를 지정(기본값은 5%)
      - ◆ mke2fs 명령은 /etc/mke2fs.conf 파일로 파일 시스템의 종류에 따라 기본적으로 설정할 값이 정의되어 있음

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - ● sbin/mk 로 시작하는 명령어(ls /sbin/mk*)

```
adam@master:~$ ls /sbin/mk*
/sbin/mkdosfs      /sbin/mkfs.ext2    /sbin/mkfs.msdos   /sbin/mkhomedir_helper
/sbin/mke2fs       /sbin/mkfs.ext3    /sbin/mkfs.ntfs    /sbin/mkinitramfs
/sbin/mkfs         /sbin/mkfs.ext4    /sbin/mkfs.ubifs   /sbin/mklost+found
/sbin/mkfs.btrfs   /sbin/mkfs.fat     /sbin/mkfs.vfat    /sbin/mkntfs
/sbin/mkfs.exfat   /sbin/mkfs.jffs2   /sbin/mkfs.xfs     /sbin/mkswap
```

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - sudo mkfs /dev/sdb1: ext2 파일 시스템 생성

      mke2fs 1.47.2 (1-Jan-2025)

      Creating filesystem with 262144 4k blocks and 65536 inodes

      Filesystem UUID: 88f9a640-8992-45a0-9128-45f645d0703a

      Superblock backups stored on blocks:

      32768, 98304, 163840, 229376

      Allocating group tables: done

      Writing inode tables: done

      Writing superblocks and filesystem accounting information: done

      - ◆ mkfs 명령으로 파일 시스템을 만들 때 출력되는 내용은 블록의 크기와 inode의 개수, 블록의 개수, 슈퍼 블록의 백업 위치 등 파일 시스템의 구성 요소에 대한 정보
      - ◆ dev/sdb1 파티션은 사용 준비가 완료
      - ◆ 마운트만 하면 바로 사용할 수 있음

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - sudo mkfs.ext3 -v /dev/sdb2: ext3 파일 시스템 생성

```
fs_types for mke2fs.conf resolution: 'ext3'
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Filesystem UUID: f356b401-d3ee-44ee-a3f7-97ff91d5a33f
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - sudo mke2fs -t ext3 /dev/sdb3: ext3 파일 시스템 생성

```
mke2fs 1.47.2 (1-Jan-2025)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 4b1caabf-79e3-4ab9-b9ea-0007730a6a30
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

디스크 관리

- ❖ 디스크 파티션 나누기
  - ✓ 파일 시스템 생성
    - sudo mke2fs -t ext4 /dev/sdb4: ext4 파일 시스템 생성

```
mke2fs 1.47.2 (1-Jan-2025)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 571a9605-1333-43f6-ab19-dee1a1d83d02
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

디스크 관리

- ❖ 디스크 마운트
  - ✓ 파티션을 나누고 파일 시스템을 생성한 후에는 디렉토리 계층 구조에 마운트하면 됨
    - 마운트 포인트 준비
      - sudo mkdir /mnt/hdd1
      - sudo mkdir /mnt/hdd2
      - sudo mkdir /mnt/hdd3
    - 마운트
      - sudo mount /dev/sdb1 /mnt/hdd1
      - sudo mount -t ext3 /dev/sdb2 /mnt/hdd2
      - mount

        /dev/sdb1 on /mnt/hdd1 type ext2 (rw,relatime)

        /dev/sdb2 on /mnt/hdd2 type ext3 (rw,relatime)
    - 마운트 해제
      - sudo cp /etc/hosts /mnt/hdd1
      - ls /mnt/hdd1

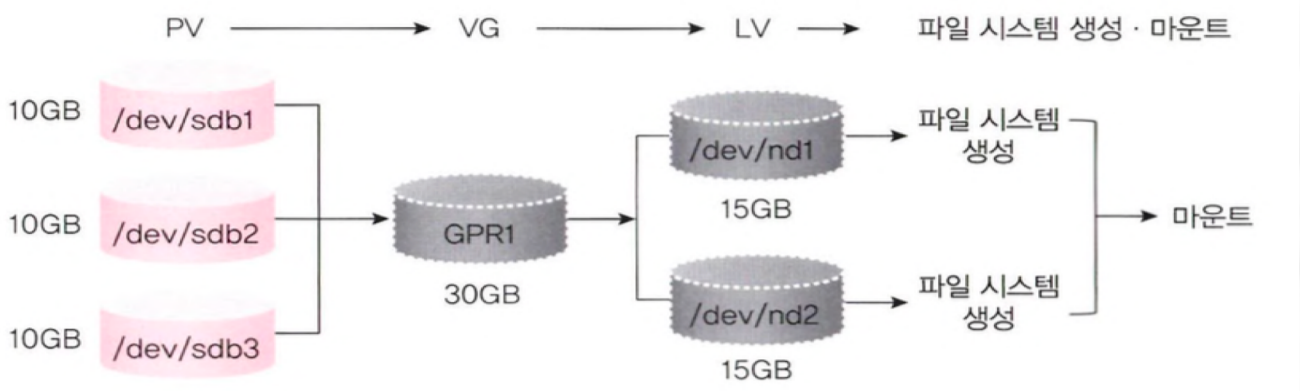
        hosts lost+found
      - sudo umount /mnt/hdd1
      - ls /mnt/hdd1

디스크 관리

- ❖ LVM
  - ✓ 개요
    - LVM(Logical Volume Manager)는 리눅스의 저장 공간을 효율적이고 유연하게 관리하기 위한 커널의 한 부분
    - LVM의 주요 용도는 여러 개의 하드디스크를 합쳐서 한 개의 파티션으로 구성한 후 다시 필요에 따라 나누는 것으로 한 개의 하드디스크를 LVM으로 구성하고 다시 파티션을 구분할 수도 있음
    - LVM vs 일반 disk partitioning
      - ◆ LVM이 아닌 기존 방식의 경우 하드 디스크를 파티셔닝 한 후 OS 영역에 마운트하여 read/wirte를 수행하는데 이 경우 저장 공간의 크기가 고정되어서 증설/축소가 어려움
      - ◆ LVM은 파티션 대신에 volume이라는 단위로 저장 장치를 다루는데 스토리지의 확장, 변경에 유연하며 크기를 변경할 때 기존 데이터의 이전이 필요없음
    - LVM 사용의 장점
      - ◆ 유연한 용량 조절
      - ◆ 크기 조절이 가능한 storage pool
      - ◆ 편의에 따른 장치 이름 지정
      - ◆ disk striping, mirror volume 등을 제공

디스크 관리

- LVM
  - ✓ 기본 개념
    - PV(Physicai Volume, 물리 볼륨): /dev/sdbl, /dev/sdb2 같은 실제 하드디스크의 파티션을 의미
    - VG(Volume Group, 볼륨 그룹): 여러 개의 PV를 그룹으로 묶은 것으로 /dev/sdb1, /dev/sdb2가 GRP1 이라는 그룹을 만든다면 GRP1이 VG
    - LV(Logical Volume, 논리 볼륨): VG를 다시 적절한 크기의 파티션으로 나눌 때 각 파티션이 LV
    - PE(Physical Extent): PV가 가진 일정한 블록
    - LE(Logical Extent): LV가 가진 일정한 블록



디스크 관리

❖ LVM

✓ 관련 명령

구분	기능	명령
PV	생성	pvcreate 파티션이름
	상태 확인	pvscan
VG	생성	vgcreate VG 명 파티션 (PV) 이름1 파티션 (PV)이름2
	활성화	vgchange -a y VG이름
	비활성화	vgchange -a n VG이름
	삭제	vgremove VG이름
	정보확인	vgdisplay –v VG이름
	PV 추가	vgextend VG이름 PV이름
	PV 삭제	vgreduce VG이름 PV이름
	이름 변경	vgrename 기존VG이름 새로운VG이름
LV	생성	lvcreate –l PE개수 VG이름 -n LV 이름
	삭제	lvremove LV 이름
	상태 확인	lvscan
	용량 확대	lvextent –l +PE개수 LV 이름
	용량 축소	lvcreate –l -PE개수 LV 이름

디스크 관리

- LVM
  - ✓ 생성 과정
    1. 기존 파일 시스템의 종류 변경: fdisk
    2. PV 생성: pvcreate
    3. VG 생성: vgcreate
    4. VG 활성화: vgchange -a y
    5. LV 생성: lvcreate
    6. LV에 파일 시스템 생성: mkfs, mke2fs
    7. LV 마운트: mount

디스크 관리

- ❖ LVM
  - ✓ 생성
    - 설치: sudo apt install lvm2
    - 파일 시스템 종류 변경
      - ◆ sudo fdisk /dev/sdb

        Welcome to fdisk (util-linux 2.40.2).
        Changes will remain in memory only, until you decide to write them.
        Be careful before using the write command.

        This disk is currently in use - repartitioning is probably a bad idea.
        It's recommended to umount all file systems, and swapoff all swap
        partitions on this disk.

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● 파일 시스템 종류 변경
      - ◆ Command (m for help): p

```
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb7479af2

Device     Boot   Start     End Sectors Size Id Type
/dev/sdb1          2048 2099199 2097152   1G 83 Linux
/dev/sdb2       2099200 4196351 2097152   1G 83 Linux
/dev/sdb3       4196352 6293503 2097152   1G 83 Linux
/dev/sdb4       6293504 8390655 2097152   1G 83 Linux
```

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● 파일 시스템 종류 변경
      - ◆ Command (m for help): t
      - ◆ Partition number (1-4, default 4): 1
      - ◆ Hex code or alias (type L to list all): 8e

      Changed type of partition 'Linux' to 'Linux LVM'.

      - ◆ Command (m for help): t
      - ◆ Partition number (1-4, default 4): 2
      - ◆ Hex code or alias (type L to list all): 8e

      Changed type of partition 'Linux' to 'Linux LVM'.

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● 파일 시스템 종류 변경
      - ◆ Command (m for help): p

```
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb7479af2

Device     Boot   Start     End Sectors Size Id Type
/dev/sdb1          2048 2099199 2097152   1G 8e Linux LVM
/dev/sdb2       2099200 4196351 2097152   1G 8e Linux LVM
/dev/sdb3       4196352 6293503 2097152   1G 83 Linux
/dev/sdb4       6293504 8390655 2097152   1G 83 Linux
```

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● 파일 시스템 종류 변경
      - ◆ Command (m for help): w

        The partition table has been altered.

        Syncing disks.

디스크 관리

- LVM
  - ✓ 생성
    - PV 생성
      - sudo pvcreate /dev/sdb1
      - sudo pvcreate /dev/sdb2
      - D-Bus 경고 메시지가 나올 경우 apt install lvm2-dbusd 명령으로 패키지를 업데이트
    - PV 상태 확인
      - sudo pvscan

      PV /dev/sdb1 lvm2 [1.00 GiB]

      PV /dev/sdb2 lvm2 [1.00 GiB]
    - PV를 통합하여 VG를 생성
      - sudo vgcreate grp1 /dev/sdb1 /dev/sdb2

      Volume group "grp1" successfully created
    - VG grp1을 활성화
      - sudo vgchange -a y grp1

      0 logical volume(s) in volume group "grp1" now active

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● VG grp1의 상태 확인
      - ◆ sudo vgdisplay -v grp1

```
--- Volume group ---
VG Name               grp1
System ID
Format                lvm2
Metadata Areas        2
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                2
Act PV                2
VG Size               1.99 GiB
PE Size               4.00 MiB
Total PE              510
Alloc PE / Size       0 / 0
Free  PE / Size       510 / 1.99 GiB
VG UUID               o3R8pB-MnP7-Pm5n-rlRh-WyIX-NywF-GI6mJ3
```

디스크 관리

- LVM
  - ✓ 생성
    - LV 생성
      - sudo lvcreate -l 510 grp1 -n mylvm1

      Logical volume "mylvm1" created.
    - LV 확인
      - sudo lvscan

      ACTIVE '/dev/grp1/mylvm1' [1.99 GiB] inherit

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● LV에 ext4 파일 시스템 생성
      - ◆ sudo mke2fs -t ext4 /dev/grp1/mylvm1

```
mke2fs 1.47.2 (1-Jan-2025)
Creating filesystem with 522240 4k blocks and 130560 inodes
Filesystem UUID: 48e0d229-aa3f-46dd-8d66-27d3b019ded7
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

디스크 관리

- ❖ LVM
  - ✓ 생성
    - ● VG 상태를 확인해서 LV 정보가 수정되었는지 확인
      - ◆ sudo vgdisplay -v grp1

```
--- Logical volume ---
  LV Path                /dev/grp1/mylvm1
  LV Name                mylvm1
  VG Name                grp1
  LV UUID                f3MUek-nbOm-UyiW-aEQQ-vxDf-2Xfz-6x8Ly6
  LV Write Access        read/write
  LV Creation host, time master, 2025-06-12 23:21:31 +0000
  LV Status              available
```

디스크 관리

- LVM
  - ✓ 생성
    - LV를 /mnt/lvm 디렉토리에 마운트 하고 파일을 복사
      - sudo mkdir /mnt/lvm
      - sudo mount /dev/grp1/mylvm1 /mnt/lvm
      - sudo cp /etc/hosts /mnt/lvm
      - $ ls /mnt/lvm

      hosts lost+found

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 파일 시스템 별 디스크 사용량 확인: df
    - ● 명령어
      - ◆ 디스크의 남은 공간에 대한 정보를 출력
      - ◆ 형식: df [옵션] [파일 시스템]
      - ◆ 옵션
        - ▪ -a: 모든 파일 시스템을 대상으로 디스크 사용량을 확인
        - ▪ -k: 디스크 사용량을 KB 단위로 출력
        - ▪ -m: 디스크 사용량을 MB 단위로 출력
        - ▪ -h: 디스크 사용량을 알기 쉬운 단위(GB, MB, KB 등)로 출력
        - ▪ -t 파일 시스템 종류: 지정한 파일 시스템 종류에 해당하는 디스크의 사용량을 출력
        - ▪ -T: 파일 시스템 종류를 출력

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 파일 시스템 별 디스크 사용량 확인: df
    - ● 옵션 없이 사용: df

```
Filesystem                         1K-blocks     Used Available Use% Mounted on
tmpfs                                 399428     1048    398380   1% /run
efivarfs                                 256        4       253   2% /sys/firmware/efi/efivars
/dev/mapper/ubuntu--vg-ubuntu--lv   11218472  8087624   2539184  77% /
tmpfs                                1997128        0   1997128   0% /dev/shm
tmpfs                                   5120        0      5120   0% /run/lock
/dev/sda2                            1992552   101568   1769744   6% /boot
/dev/sda1                            1098632     6516   1092116   1% /boot/efi
tmpfs                                 399424       12    399412   1% /run/user/1000
```

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 파일 시스템 별 디스크 사용량 확인: df
    - ● 옵션 없이 사용: df
      - ◆ 출력
        - ▪ 파일 시스템 장치명
        - ▪ 파일 시스템의 전체 용량
        - ▪ 파일 시스템의 사용량
        - ▪ 파일 시스템의 사용 가능한 남은 용량
        - ▪ 퍼센트로 나타낸 사용량
        - ▪ 마운트포인트

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 파일 시스템 별 디스크 사용량 확인: df
    - ● 파일 시스템 사용량을 이해하기 쉬운 단위로 표시: df -h

```
Filesystem                         Size  Used Avail Use% Mounted on
tmpfs                              391M  1.1M  390M   1% /run
efivarfs                           256K  3.6K  253K   2% /sys/firmware/efi/efivars
/dev/mapper/ubuntu--vg-ubuntu--lv   11G  7.8G  2.4G  77% /
tmpfs                              2.0G     0  2.0G   0% /dev/shm
tmpfs                              5.0M     0  5.0M   0% /run/lock
/dev/sda2                          2.0G  100M  1.7G   6% /boot
/dev/sda1                          1.1G  6.4M  1.1G   1% /boot/efi
tmpfs                              391M   12K  391M   1% /run/user/1000
```

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 파일 시스템 별 디스크 사용량 확인: df
    - ● 파일 시스템의 종류 출력: df -Th

```
Filesystem                Type      Size  Used Avail Use% Mounted on
tmpfs                     tmpfs     391M  1.1M  390M   1% /run
efivarfs                  efivarfs  256K  3.6K  253K   2%
/sys/firmware/efi/efivars
/dev/mapper/ubuntu--vg-ubuntu--lv ext4       11G  7.8G  2.4G  77% /
tmpfs                     tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs                     tmpfs     5.0M     0  5.0M   0% /run/lock
/dev/sda2                 ext4      2.0G  100M  1.7G   6% /boot
/dev/sda1                 vfat      1.1G  6.4M  1.1G   1% /boot/efi
tmpfs                     tmpfs     391M   12K  391M   1% /run/user/1000
```

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 디렉토리나 사용자 별 사용량 확인: du
    - 기능: 디스크의 사용 공간에 대한 정보를 출력
    - 형식: du [옵션] [디렉토리]
    - 옵션
      - ◆ 옵션없이 사용하면 현재 디렉토리의 사용량을 출력
      - ◆ -s 디렉토리: 특정 디렉터리의 전체 사용량을 출력
      - ◆ -h: 디스크 사용량을 알기 쉬운 단위(GB, MB, KB 등)로 출력

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 디렉토리나 사용자 별 사용량 확인: du
    - ● 현재 디렉토리의 사용량 확인: du

      4 ./.cache

      4 ./.ssh

      36 .
    - ● 특정 디렉토리의 사용량 확인: sudo du -s /etc

      6400 /etc
    - ● 특정 사용자의 사용량 확인: du -sh ~adam

      36K /home/adam

디스크 관리

- ❖ 디스크 사용량 확인
  - ✓ 연습
    - 파일 시스템의 디스크 사용량을 MB 단위로 출력
    - ext4 파일 시스템의 디스크 사용량 출력
    - 전체 파일 시스템의 디스크 사용량을 출력하는데 용량이 0인 파일 시스템의 정보도 출력
    - /usr 디렉터리가 사용하고 있는 디스크 사용량을 출력
    - ext3 파일 시스템의 디스크 사용량을 출력
    - ext4 파일 시스템의 디스크 사용량과 파일 시스템의 종류를 함께 출력
    - /tmp 디렉터리의 서브 디렉터리까지 디스크 사용량을 출력

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 파일 시스템 검사: fsck
    - ● 명령어 형식
      - ◆ 기능: 리눅스의 파일 시스템을 점검
      - ◆ 형식: fsck [옵션] [장치명]
      - ◆ 옵션
        - ▪ -f: 강제로 점검
        - ▪ -b 슈퍼블록: 슈퍼블록으로 지정한 백업 슈퍼블록을 사용
        - ▪ -y: 모든 질문에 yes로 대답
        - ▪ -a: 파일 시스템 검사에서 문제를 발견했을 때 자동으로 복구

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 파일 시스템 검사: fsck
    - ● 파일 시스템 강제 검사
      - ◆ sudo fsck -f /dev/sdb3

```
fsck from util-linux 2.39.3
e2fsck 1.47.0 (5-Feb-2023)
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/sdb3: 11/128000 files (0.0% non-contiguous), 12214/128000 blocks
```

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 파일 시스템 검사: fsck
    - ● 파일 시스템 종류를 지정한 검사
      - ◆ sudo fsck.ext4 /dev/sdb4

        e2fsck 1.47.0 (5-Feb-2023)

        /dev/sdb4: clean, 11/128000 files, 12302/128000 blocks

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 파일 시스템 검사: e2fsck
    - fsck처럼 inode 및 블록, 디렉토리, 파일 링크 등을 검사하고 필요시 복구 작업도 수행
    - e2fsck 명령으로 파일 시스템을 점검할 때는 해당 파일 시스템의 마운트를 해제해야 하는데 마운트되어 있는 상태에서 e2fsck 명령을 수행할 경우 예상치 않은 오류가 발생할 수도 있으므로 주의
    - 명령어 형식
      - ◆ 기능: 리눅스의 확장 파일 시스템(ext2, ext3, ext4)을 점검
      - ◆ 형식: e2fsck [옵션] [장치명]
      - ◆ 옵션
        - -f: 강제로 점검
        - -b 슈퍼블록: 슈퍼블록으로 지정한 백업 슈퍼블록을 사용
        - -y: 모든 질문에 yes로 응답
        - -j: ext3/ext4: ext3나 ext4 파일 시스템을 검사할 때 지정

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 파일 시스템 검사: e2fsck
    - 일반적인 파일 시스템 검사: sudo e2fsck /dev/sdb1

      e2fsck 1.47.0 (5-Feb-2023)

      /dev/sdb1: clean, 11/128000 files, 8113/128000 blocks
    - 파일 시스템 강제 검사: sudo e2fsck -f /dev/sdb1

      e2fsck 1.47.0 (5-Feb-2023)

      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/sdb1: 11/128000 files (0.0% non-contiguous), 8113/128000 blocks

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 배드 블록 검사: badblocks
    - ● 명령어 형식
      - ◆ 기능: 장치의 배드 블록을 검색
      - ◆ 형식: badblocks [옵션] [장치명]
      - ◆ 옵션
        - ▪ -v: 검색 결과를 자세하게 출력
        - ▪ -o 출력 파일: 검색한 배드 블록 목록을 지정한 출력 파일에 저장

디스크 관리

- ❖ 파일 시스템 검사 및 복구
  - ✓ 배드 블록 검사: badblocks
    - sudo badblocks -v /dev/sdb1

      Checking blocks 0 to 511999

      Checking for bad blocks (read-only test):

      done

      Pass completed, 0 bad blocks found. (0/0/0 errors)

디스크 관리

- RAID
  - ✓ RAID(Redundant Array of Independent Disks / Redundant Array of Inexpensive Disks)는 여러 개의 하드디스크를 하나의 하드디스크처럼 사용하는 방식으로 비용을 절감하면서도 신뢰성을 높이며 성능까지 향상시킬 수 있음
  - ✓ RAID의 종류는 크게 하드웨어 RAID와 소프트웨어 RAID로 나눌 수 있음
  - ✓ 하드웨어 RAID
    - 하드웨어 RAID는 하드웨어 제조업체에서 여러 개의 하드디스크를 연결한 장비를 만들어 그 자체를 공급하는 것
    - 하드웨어 RAID는 좀 더 안정적이고 각 제조업체에서 기술 지원을 받을 수 있기 때문에 많이 선호하는 방법
    - 최근에는 저렴한 가격의 제품도 출시되고 있지만 안정적이고 성능이 좋은 제품은 여전히 매우 고가
    - 하드웨어 RAID는 대개 고가의 경우 SA-SCSI 하드디스크를 중저가의 경우 SATA 하드디스크를 사용해 만듬
  - ✓ 소프트웨어 RAID
    - 고가 하드웨어 RAID의 대안으로 하드디스크만 여러 개 있으면 운영체제에서 지원하는 방식
    - 하드웨어 RAID와 비교하면 신뢰성이나 속도 등이 떨어질 수 있지만 저렴한 비용으로 좀 더 안전하게 데이터를 저장할 수 있다는 점에서 적극 고려해볼 수 있는 방식

디스크 관리

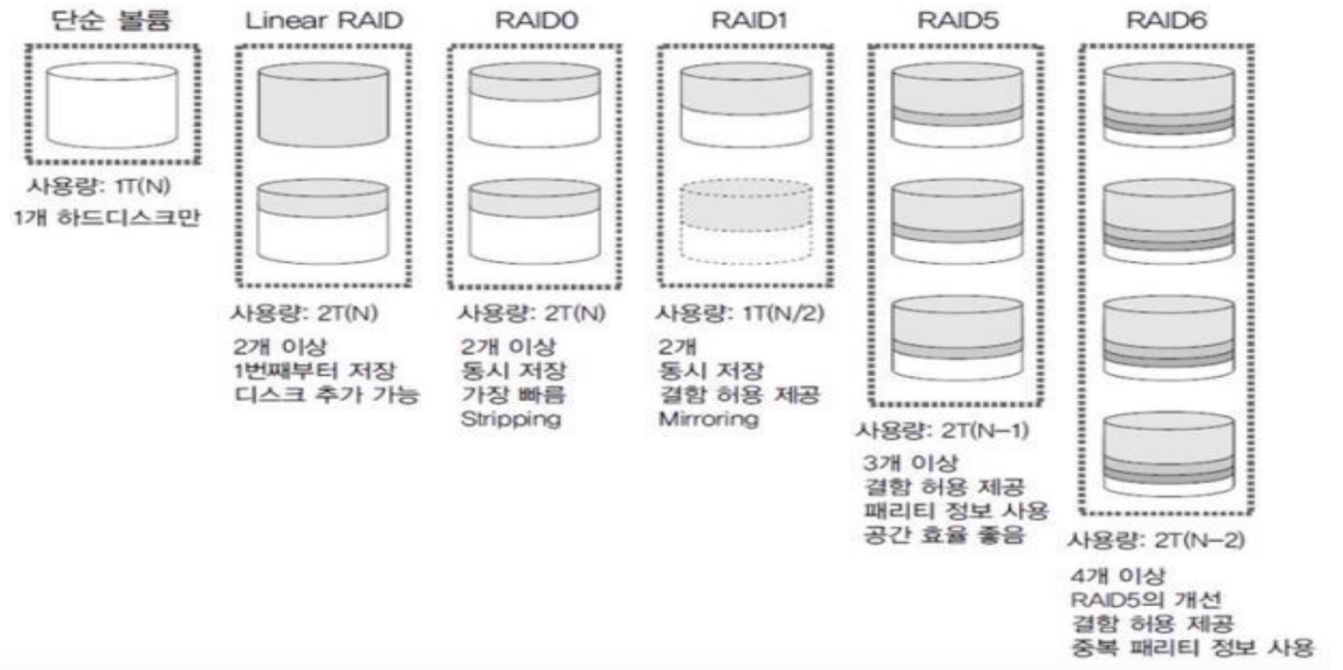
- RAID
  - ✓ 개요
    - 네트워크를 구성하려면 많은 트래픽을 견뎌야 하고, 어마어마한 용량을 필요로 함
    - RAID(Redundant Array of Independent Disks / Redundant Array of Inexpensive Disks)는 여러 개의 하드디스크를 하나의 하드디스크처럼 사용하는 방식으로 비용을 절감하면서도 신뢰성을 높이며 성능까지 향상시킬 수 있음
    - 기대 효과
      - 대용량의 단일 볼륨을 사용하는 효과
      - 디스크 I/O 병렬화로 인한 성능 향상
      - 데이터 복제로 인한 안정성 향상
    - 주 사용 목적은 무정지(고가용성을 보장하는) 구현과 고성능 구현으로 구분되며 무정지 구현은 RAID 1, 고성능은 RAID 0로 대표됨
    - 하드디스크의 느린 속도를 보완하기 위해 만든 기술로 RAID의 구성 방식에 따라 성능, 용량이 바뀜
    - RAID의 종류는 크게 하드웨어 RAID와 소프트웨어 RAID로 나눌 수 있음

디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - RAID는 기본적으로 구성하는 방식에 따라 Linear RAID, RAID 0, RAID 1, RAID 2, RAID 3, RAID 4, RAID 5까지 일곱 가지
    - Standard RAID 레벨에는 대표적으로 Linear RAID, RAID 0, 1, 5, 6이 있음
    - 주로 사용되는 방식은 Linear RAID, RAID 0, RAID 1, RAID 5 와 RAID 5의 변형 인 RAID 6, 그리고 RAID 1 과 0의 혼합인 RAID 1+0 등

디스크 관리

- RAID
  - ✓ RAID 레벨



디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● Linear RAID
      - ◆ 1TB 하드가 3개라면 정보를 저장할 때 위에서부터 순차적으로 저장하여 3TB를 사용하는 방식으로 가장 고전적인 방식

디스크 관리

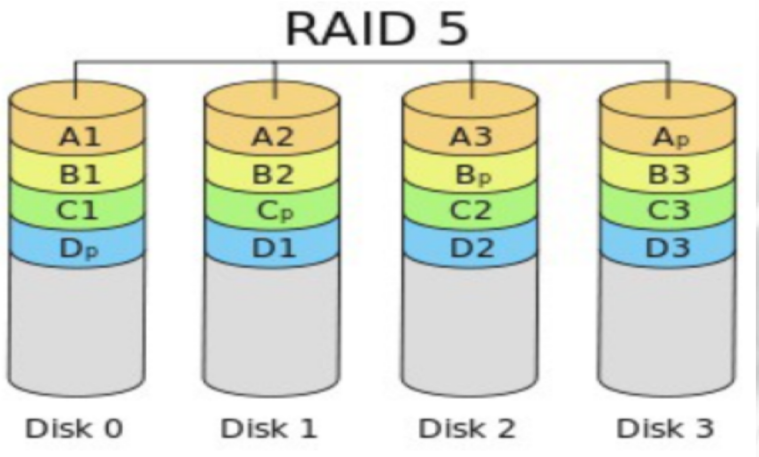
- RAID
  - ✓ RAID 레벨
    - RAID 0
      - ◆ 최소 2개의 하드디스크를 사용해서 하나의 정보를 2개 이상의 디스크에 나눠서 저장 - Stripping
      - ◆ Linear RAID보다 저장 속도의 효율이 좋음
      - ◆ 이론적으로 단일 디스크를 사용하는 것에 비해 N배의 성능을 보장하지만 정보를 나누어서 저장한 하나의 디스크라도 고장나면 모든 정보를 사용할 수 없게되므로 신뢰성이 낮음
      - ◆ 오류 검출 기능이 없어 멤버 디스크를 늘릴수록 안정성이 떨어지며 RAID 시키는 하드디스크 용량이 각각 다르게 되면 공간 효율이 100%가 될 수 없는데 가장 작은 하드디스크의 용량에 맞춰 저장되기 때문
      - ◆ 이 방식을 사용하게 되면 서로 호환이 되는 동일 제조사, 동일 용량의 디스크를 사용하는 것이 일반적
      - ◆ 이미지 프로세싱, 데이터베이스 캐싱 등 빠른 입출력 성능을 필요로 하며 데이터 손실이 문제되지 않는 환경에서 쓰일 수 있지만 상용 환경에서는 권장하지 않음

디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 1
      - ◆ 미러링 방식이라고 부르며 동일한 데이터를 2개 이상의 하드디스크에 동일하게 저장하므로 저장 효율이 나쁨
      - ◆ 공간 효율은 좋지 않지만 하나의 디스크가 고장나도 다른 디스크는 문제 없이 사용가능 하므로 정보 저장의 신뢰성이 높음
      - ◆ 멤버 디스크가 늘더라도 저장 공간은 증가하지 않지만 가용성이 크게 증가하며 서버에서 끊김없이 지속적으로 서비스를 제공하기 위해 사용
      - ◆ 쓰기 속도는 이론적으로 소폭 하락하는데 이는 하향 평준화가 기준이기 때문인데 동일 제품으로 구성한 경우에는 차이가 없을 수 있음
      - ◆ 백업의 목적보다는 가용성에 초점을 맞추고 사용하는 것이 옳음
      - ◆ 디스크 고장 상황에서 백업과 유사한 보호 기능을 제공하며 가장 최신화 된 데이터를 항상 유지할 수 있으나 랜섬웨어 등에 대응할 수 없다는 점에서 반쪽짜리 백업에 불과
      - ◆ 비용 효율이 좋지 않음

디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 5
      - ◆ RAID 0 과 RAID 1의 장점을 조합해서 사용하는 방식.
      - ◆ RAID 1의 데이터 안전성 + RAID 0의 빠른 저장 속도와 공간 효율
      - ◆ 제일 사용 빈도가 높음
      - ◆ 패리티(Parity)를 사용하게 되는데 일반적으로 하드디스크 7~10개를 연결
      - ◆ 하나의 하드디스크가 고장나도 패리티를 이용해서 데이터를 복구할 수 있음
      - ◆ RAID 2, 3, 4 방식도 있지만 5보다 효율이 떨어지므로 잘 사용되지 않음



디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 5
      - ◆ 저장 및 복구 원리
        - ▪ Block 단위로 스트라이핑을 하고 error correction을 위해 패리티를 한개의 디스크에 저장하는데, 패리티를 저장하는 디스크를 고정하지 않고 매 번 다른 디스크에 저장
        - ▪ 패러티를 이용해서 하나의 디스크가 문제가 생겨도 잃어버린 데이터를 복구 할 수 있는데 패리티를 한 디스크에 넣지 않고 각 멤버 디스크에 순환적으로 저장해서 입출력 병목 현상을 해결할 수 있음
        - ▪ 용량 및 성능이 단일 디스크 대비 N-1 배 증가
        - ▪ 하나의 멤버 디스크 고장에는 견딜 수 있지만 두개 이상 고장나면 데이터가 모두 손실
        - ▪ DB 서버 등 큰 용량과 무정지 복구 기능을 동시에 필요로 하는 환경에서 주로 사용하는데 매 쓰기 작업에 패리티 연산 과정이 추가되어 성능을 보장하려면 패리티 연산 전용 프로세서와 메모리를 사용해야하고 멤버 디스크도 3개 이상 사용해야하므로 초기 구축 비용이 비싸다는 단점이 있음
        - ▪ 최소 3개의 디스크로 구성 가능

디스크 관리

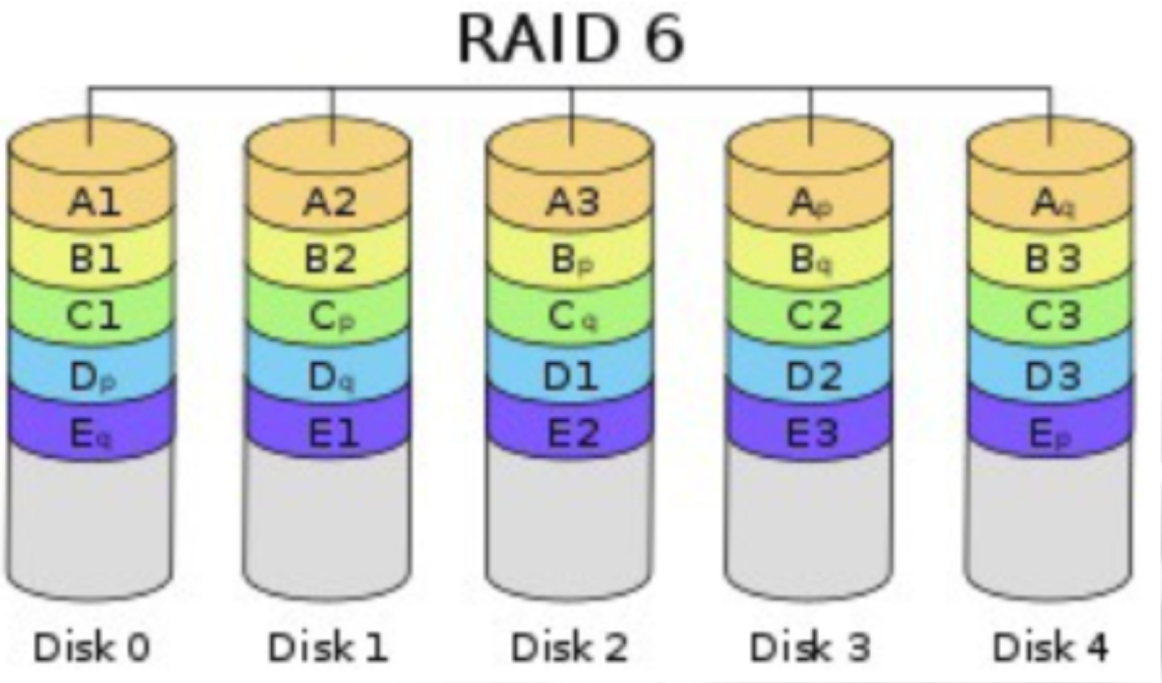
- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 5
      - ◆ 저장 및 복구 원리
        - ▪ 읽기 작업은 전체 디스크에 분산되어 속도가 향상되지만 쓰기 작업은 성능이 약간 떨어지는데 적어도 둘 이상의 디스크(데이터+패리티)에서 진행해야하기 때문
        - ▪ 가용성은 높은 편이나 패리티 연산을 통해 데이터를 저장한다는 특징 때문에 까다로움
        - ▪ 짝수 패리티 방식
          - 3개의 디스크에 정보를 나눠서 저장한 후 하나의 빈 정보를 따로 남은 디스크에 저장
          - 남은 디스크의 저장 공간을 패리티라고 부르며 하나의 디스크가 고장나게 되면 패리티를 포함한 데이터의 값을 짝수로 만들어야 하므로 이를 통해 패리티에 들어갈 숫자를 인식해서 데이터를 복구

디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 6
      - ◆ RAID 5에서 성능 및 용량을 좀 더 줄이고 안정성을 높인 레벨
      - ◆ Block 단위로 스트라이핑을 하고 error correction을 위해 패리티를 두 개의 디스크에 저장하는데 패리티를 저장하는 디스크를 고정하지 않고 매 번 다른 디스크에 저장
      - ◆ 용량 및 성능이 단일 디스크 대비 N-2배 증가
      - ◆ 패리티를 2개 사용하므로 RAID 5 보다 가용성이 높지만 정보를 저장하지 않은 디스크 2개나 생겨 공간 효율과 성능이 떨어지므로 활용 빈도는 조금 떨어짐
      - ◆ 5가 1개까지의 고장을 허용했다면 6은 2개까지는 허용
      - ◆ RAID 5보다 초기 구축 비용이 증가하는데 최소 4개의 디스크로 구성되어야 하기 때문

디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 6



디스크 관리

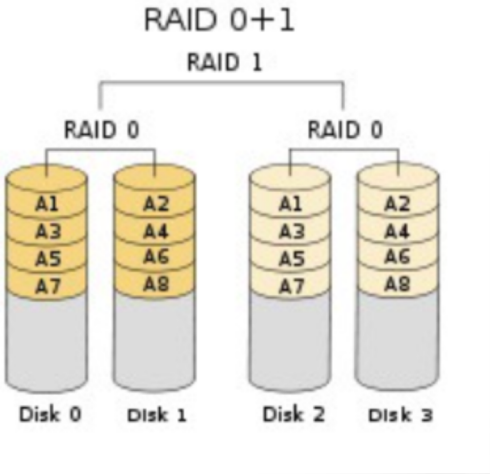
- ❖ RAID
  - ✓ RAID 레벨
    - ● RAID 6
      - ◆ RAID 5에서 성능 및 용량을 좀 더 줄이고 안정성을 높인 레벨
      - ◆ Block 단위로 스트라이핑을 하고 error correction을 위해 패리티를 두 개의 디스크에 저장하는데 패리티를 저장하는 디스크를 고정하지 않고 매 번 다른 디스크에 저장
      - ◆ 용량 및 성능이 단일 디스크 대비 N-2배 증가
      - ◆ 패리티를 2개 사용하므로 RAID 5 보다 가용성이 높지만 정보를 저장하지 않은 디스크 2개나 생겨 공간 효율과 성능이 떨어지므로 활용 빈도는 조금 떨어짐
      - ◆ 5가 1개까지의 고장을 허용했다면 6은 2개까지는 허용
      - ◆ RAID 5보다 초기 구축 비용이 증가하는데 최소 4개의 디스크로 구성되어야 하기 때문

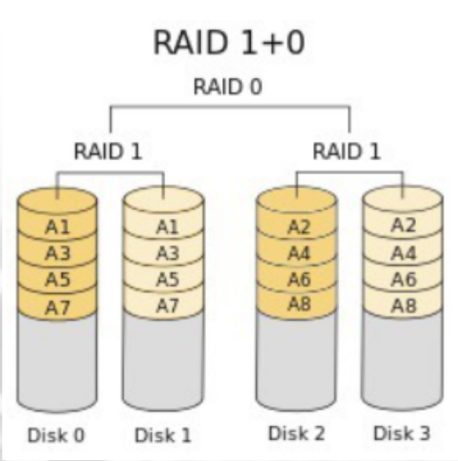
디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● Nested RAID (복합)
      - ◆ 0~6 레벨의 Standard RAID를 여러 개 중첩하여 사용
      - ◆ RAID 볼륨의 멤버로 다른 레이드 볼륨을 사용하는 형태
      - ◆ 0 과 1 조합 및 5 와 0 조합 그리고 5 와 1 조합 등을 사용

디스크 관리

- ❖ RAID
  - ✓ RAID 레벨
    - ● Nested RAID (복합)
      - ◆ RAID 1+ RAID 0
        - ▪ RAID 6보다 더 안정적이면서 RAID 5의 장점을 극대화하기 위해 RAID 1과 RAID 0의 장점을 조합하여 사용하는 방식
        - ▪ RAID 5가 패리티를 사용해 공간 효율이 떨어졌지만 이 방법은 1과 0의 방식을 직접 조합시켜 그러한 단점을 보완한 방식
        - ▪ 신뢰성과 성능 두 가지 모두 만족하는 방법





디스크 관리

- ❖ RAID
  - ✓ RAID 구현
    - ● Hardware RAID
      - ◆ 별도의 RAID 카드를 장착하여 구현하는 방법
      - ◆ RAID 카드에는 RAID를 관리하는 컨트롤러 칩셋과 방열판, 캐시로 사용하기 위한 메모리가 달려 있고 디스크를 연결하기 위한 인터페이스(주로 SAS)가 달려있음
      - ◆ 장애 시 캐시의 내용을 잠시나마 유지시키기 위해 배터리가 달려 있기도 함
      - ◆ RAID 카드는 대부분 PCI-E 슬롯에 꽂아서 사용하며 OS부팅 이전 RAID 카드의 설정 페이지로 진입하여 관리
      - ◆ RAID 구성 후 부팅을 하게 되면 OS에서는 구성이 완료된 RAID 볼륨만 확인할 수 있는데 RAID를 구성하고 있는 단일 디스크의 정보는 RAID 카드를 통해서만 알 수 있음
      - ◆ 카드에 달려 있는 별도의 컨트롤러 칩셋이 RAID를 관리하기 때문에 다른 방식에 비해 성능이 월등히 높은데 특히 별도의 패리티 연산이 필요한 5 나 6에서 매우 높은 성능을 보여주며 RAID를 구성할 수 있는 최대 디스크 개수도 월등히 높음
      - ◆ RAID 카드와 연결되어 있는 디스크를 그대로 제거하여 다른 PC, 서버로 이동시키면 기존 운영하던 RAID 볼륨을 거의 100% 그래도 유지할 수 있다는 것도 장점
      - ◆ 별도의 RAID 카드를 사야하기 때문에 구축 비용이 많이 들고 RAID를 구성하는 단일 디스크의 분석을 거의 할 수 없다는 것이 단점

디스크 관리

- ❖ RAID
  - ✓ RAID 구현
    - ● 펌웨어(드라이버) RAID
      - ◆ 드라이버 RAID, On-board RAID, 임베디드 RAID 등으로 불림.
      - ◆ RAID 카드 대신 기능을 간략화한 RAID 칩을 탑재하고 펌웨어(드라이버)로 제어하여 구현하는 방법
      - ◆ OS 진입 전 BIOS 메뉴에서 RAID를 구현
      - ◆ OS에 관계없이 작동하며 OS에서는 원래 장착한 디스크 대신 가상의 BIOS RAID 하드웨어가 표시되므로 별도의 드라이버 소프트웨어를 통한 관리가 불가능
      - ◆ 펌웨어 RAID는 OS부팅 전에 RAID를 구성하기 때문에 OS 변경에는 영향을 미치지 않기 때문에 OS를 바꿔도 RAID는 유효한 대신 메인보드를 바꾸게 되면 더 이상 사용하지 못할 가능성이 큼
      - ◆ 원래 디스크를 가상 디스크가 대체하는 방식이다 보니 용량이 다른 두 하드웨어를 묶었을 때 남은 공간은 활용 못하고 버려지는 단점이 있음
      - ◆ 전통적으로는 별도 RAID 컨트롤러를 사용하는 것이 안정성이 좋고 유지보수 등의 강점이 있는 것으로 알려져 있으나 최근에는 메인보드 내장 RAID 컨트롤러 또한 상당한 성능을 보여주고 있음
      - ◆ 안정성 면에서도 별도의 RAID 컨트롤러에 비해 부실하기 때문에 RAID Array가 깨졌을 시 데이터 복구는 힘들 수 있음
      - ◆ 하드웨어 RAID와 소프트웨어 RAID의 중간

디스크 관리

- ❖ RAID
  - ✓ RAID 구현
    - ● 소프트웨어 RAID
      - ◆ OS 수준의 RAID
      - ◆ RAID 소프트웨어(프로그램)을 이용하여 RAID를 구성하는 방식으로 OS부팅 이후 관련 프로그램을 실행하여 RAID를 구축
      - ◆ OS가 인식하고 있는 블록 디바이스를 사용해서 구축하게 되며 OS의 디스크 관리 메뉴에서 RAID를 구현
      - ◆ 하드웨어 RAID와 다르게 RAID를 구성하고 있는 단일 디스크에 대한 분석 가능
      - ◆ 단점으로는 OS 위에서 동작하는 프로그램이 관리하기 때문에 상대적으로 속도가 매우 느리고 다른 프로그램들과 리소스(CPU, 메모리 등)을 같이 사용하기 때문에 전체적인 시스템의 성능이 떨어질 수도 있음
      - ◆ OS RAID는 메인보드를 바꾸더라도 해당 디스크만 제대로 꽂아주면 계속 RAID를 사용할 수 있지만 OS를 바꾸면 보통 사용하지 못함
      - ◆ OS에서 관리하므로 다양한 방법으로 RAID를 구성할 수 있으며 용량이 다른 두 제품이 경우 RAID를 구성하고 남는 공간에 단일 파티션 또는 또 다른 RAID Array를 구성할 수도 있음
      - ◆ 파일 시스템은 언제나 해당 OS에서 네이티브로 잘 지원되는 것을 사용하는 것이 안전하기 때문에 호환성 측면에서는 메인보드 RAID 보다는 OS RAID가 좀 더 좋음