우선 리눅스 디렉터리는 최상위(/ root)를 기준으로 하위 디렉터리들이 존재하는 계층적 트리구조이다. /root는 루트 관리자 홈 디렉터리이다. 헷갈리면 안 된다.

우선 이 4가지가 제일 중요하다.

/bin : 필수 실행 파일(기본적인 명령어 : ls, cp, mv, cat 등)의 기본 명령어가 저장된 디렉터리다.

/sbin : 시스템 관리 명령어(shutdown, reboot 등, 관리자용 시스템 표준 명령어)가 저장되어 있다.

/dev : 디바이스(장치) 파일이 저장되어 있다. (하드디스크/sda, 터미널/tty)

/etc : 시스템 환경 파일 저장(passwd, hosts, shadow)

키워드 : 가상 파일, 프로세스 하드웨어 - /proc

키워드 : 가변 자료, 시스템 로그, 메일 등 - /var

리눅스 라이선스 - 카피라이트 표기법 : Copyright 2024, ⓒ 2024, (C) 2024

GNU GPL : copyleft 저작권은 없고, 사용은 자유 재배포 가능, 외부에 공표, 배포할 시에 소스코드를 전체적으로 다 공개해야 한다. 자유 소프트웨어이다.

LGPL : 1991년 프리 소프트웨어 재단(FSF)의 리처드 스톨먼은 GPL 라이선스 강력 카피레프트 조건을 절충하여 단순 사용은 소스코드 공개 비의무를 허가 발표를 했다. 라이브러리에 응용프로그램 링크 시킬 경우 소스코드 공개할 의무가 없기 때문에 독점 소프트웨어로 사용 가능하다. 이 라이선스로 개발 후 GPL로 변경이 가능하다.

MPL : GPL과 BSD의 혼합적 성격이다. 2차 저작물부터 소스코드 공개해야 하지만 MPL 소스코드와 혼합하여 다른 프로그램을 개발할 경우 수정된 코드만 공개하면 된다. 전체 프로그램은 비공개가 가능하다. 원 저작자에게 수정한 부분을 알려줘야 한다.

BSD, Apache, MIT : 관련 라이선스가 적용된 소스코드를 수정하여 만든 2차적 저작물에 대해 소스코드 미공개 허용, 해당 라이선스가 적용된 소프트웨어를 다운해서 부분 또는 전체를 개인적이나 상업용 목적으로 사용 가능

BSD : 캘리포니아 대학교 버클리 캠퍼스에서 배포했고, 수정본 무료로 재배포하는 것은 의무적인 사항이 아니므로 상용 소프트웨어에서도 사용한다.

MIT : MIt대학교에서 배포했다. 원작자 저작권 정보와 라이선스 내용을 제거해서는 안된

다. X windows, node.js, Jquery등에서 적용했다.

Apache : 재배포 시 아파치 라이선스 2.0 전문을 포함시키며, 관련 소프트웨어임을 밝혀야 한다.

수정 소스코드 공개 : GPL, LGPL, MPL – PL로 끝나는 건 일부든 전체든 공개해야 한다.

수정 소스코드 미공개 : BSD, Apache, MIT 보통 라이선스만 밝히면 되는 듯 하다.

BSD, Apache, MIT는 수정본 재배포가 의무가 아니기 때문에 상용 소프트웨어로 사용 가능하다.

성격이 다른 라이선스 문제는 공개 미공개로 분류한 것들을 잘 기억해서 분류하면 된다.

명령어 usermod의 -l 은 계정명을 변경하는데 변경 ID가 먼저 나오고 기존 ID가 뒤에 나와야 한다. 디테일적인 요소 기억하기

명령어 userdel은 계정을 삭제하는 명령어이다. 홈디렉터리 및 모든 정보를 삭제하는 -r 옵션이 있고 사용중이어도 삭제하는 -f 옵션이 있다. 하지만 고려해야 하는 사항이 있다. 홈디렉터리를 삭제하지 않고 계정을 삭제한다면 새로운 계정이 사용하던 uid를 받으면 정보를 볼 수 있는 문제가 발생한다. 하지만 잠금만 한다면 uid를 삭제하지 않는다.

삭제될 계정이 소유하는 파일이 있는지 확인도 해야한다.

삭제 전 : find / -user username -ls 해당 계정이 소유하는 모든 파일을 검색한다.

삭제 후 : find / -uid UID -ls or find /-nouser -ls로 확인해야 한다.

su : switch user – 모든 옵션은 해당 사용자의 환경변수를 적용하여 로그인하는 것이다.

chage : 사용자 계정의 패스워드 만료 정책을 변경하는 명령어이다. /etc/shadow 의 날짜 관련 필드에 모두 설정할 수 있는 명령어이다. 패스워드에 대한 정보를 보여준다.

cp 명령어는 파일 및 디렉터리를 복사한다. 하지만 mv는 사용 방식에 따라 파일 및 디렉터리 이동과 이름 변경이 가능하다. mv 뒤 파일 및 디렉터리 이름이 나오고 뒤에 경로가 나오면 이동, 이름이 동일하게 나오면 이름 변경이다.

chown : 파일 및 디렉터리 소유자, 그룹 변경 명령어이다. :이 들어가는 게 특징이며 콜론의 위치에 따라 변경하는 것이 다르다. 변경 파일 앞에 변경할 이름을 적어주고 콜론을 사용하지 않으면 소유자만 변경, 콜론을 이름: 이렇게 주면 모두 변경, :이름 이런 식으로 콜론을 주면 그룹만 변경한다.

chgrp : 이 명령어는 root 사용자가 파일 및 디렉터리의 그룹을 변경하는 명령어이다. Chown :root 와 동일하다.

-rwxr-xr-x 이런 형태는 파일 및 디렉터리의 종류 그리고 권한을 나타낸다.

- 하이픈은 -을 뜻하며 해당 권한이 없는 것이다. Chmod -R 755 dir1 이렇게 하면 이 디렉터리에 대해 소유자는 모든 권한을 갖고 그룹은 읽기 및 실행, 기타도 동일한 권한을 갖는다. -R 옵션은 하위 모든 파일과 서브 디렉터리까지 권한을 변경하는 것이다.

앞에서부터 종류 rwx 한쌍을 이루어 소유자 그룹 기타 이런 형태를 나타낸다.

-는 일반 파일, d는 디렉터리, l은 심볼릭링크(원본 파일을 카리킨다.), b는 블록디바이스파일, c는 문자디바이스 파일로 하드디스크나 주변 장치를 파일로 이용하는 것들이다.

마지막 s는 소켓 파일로 PC간 통신(IPC)나 네트워크 간 통신 파일을 뜻한다.

Chmod 명령어에 기호모드라는 것이 있다. 형식은 ugoa 라는 명령어이다. 앞에서부터 user, group, other, all을 의미하며 생략 시 all이 적용된다. 여기서 all은 모든 사용자 즉 u g o를 포함한다. +-=는 순서대로 권한의 추가 제거 지정을 의미한다.

+ 추가, - 제거, = 지정한 권한만 설정 기존 권한 제거 후 설정한다.

Chmod u+x file1 => 소유자에게 실행 권한을 추가한다.

chmod u=rwx, g=rx, o=x file2 -> 소유자에게 모든 권한 지정, 그룹은 읽기와 실행만, 기타는 실행 권한만 지정한다.

umask : 생성될 때 기본적을 권한을 설정하는 값이다. 파일은 666, 디렉터리는 777이 바람직한 듯 하다. Umask 002로 쓰면 기타 사용자에게 2 즉 쓰기 권한을 주지 않는다는 것이다.

이 밑에 글은 chmod 포함 내용이다.

002는 디렉터리에 775라는 접근권한을 갖게 되는 것이고 파일은 664가 된다. 파일은 기본적으로 6부터 시작하는 듯하다.

명령어 뒤 지정한 숫자는 원래의 접근 권한에서 뺄셈한 결과이다. 4자리 표기가 일반적이며 0022일 때 첫 번째는 특수 비트이고 022가 소유자 그룹 기타 사용자에 대한 권한 설정 부분이다. 동일하게 계산하면 된다.

특수 비트 4는 파일의 소유자 권한으로 프로세스 수행할 수 있게 만들어준다. 소유자 실행 권한에 S가 들어간다.

2는 실행자의 그룹 권한이 아닌 파일 소유자의 그룹 권한으로 실행된다. 동일하게 S가 들어간다. 디렉터리에 설정된 경우 그룹 구성원의 공유 공간으로 사용할 수 있다.

1은 기타 사용자의 실행 권한에 t가 표시된다. 주로 공유 디렉터리(/tmp)에 설정되며, 누

구나 그 디렉터리에 파일 생성 가능하지만 파일의 소유자만 삭제할 수 있다.

Tmp는 모든 사용자들이 공동으로 사용하는 특성을 갖고 있다. 그렇기에 sticky-bit의 공용 디렉터리 설정 시 사용한다.

파일 권한 r – 파일 내용 보기 w 수정 삭제 이름 변경 등 x 실행

디렉터리 권한 r – 하위 파일 서브 디렉터리 목록 보기, w 파일과 동일, x 디렉터리 내부 접근 권한 내부 파일 실행 권한

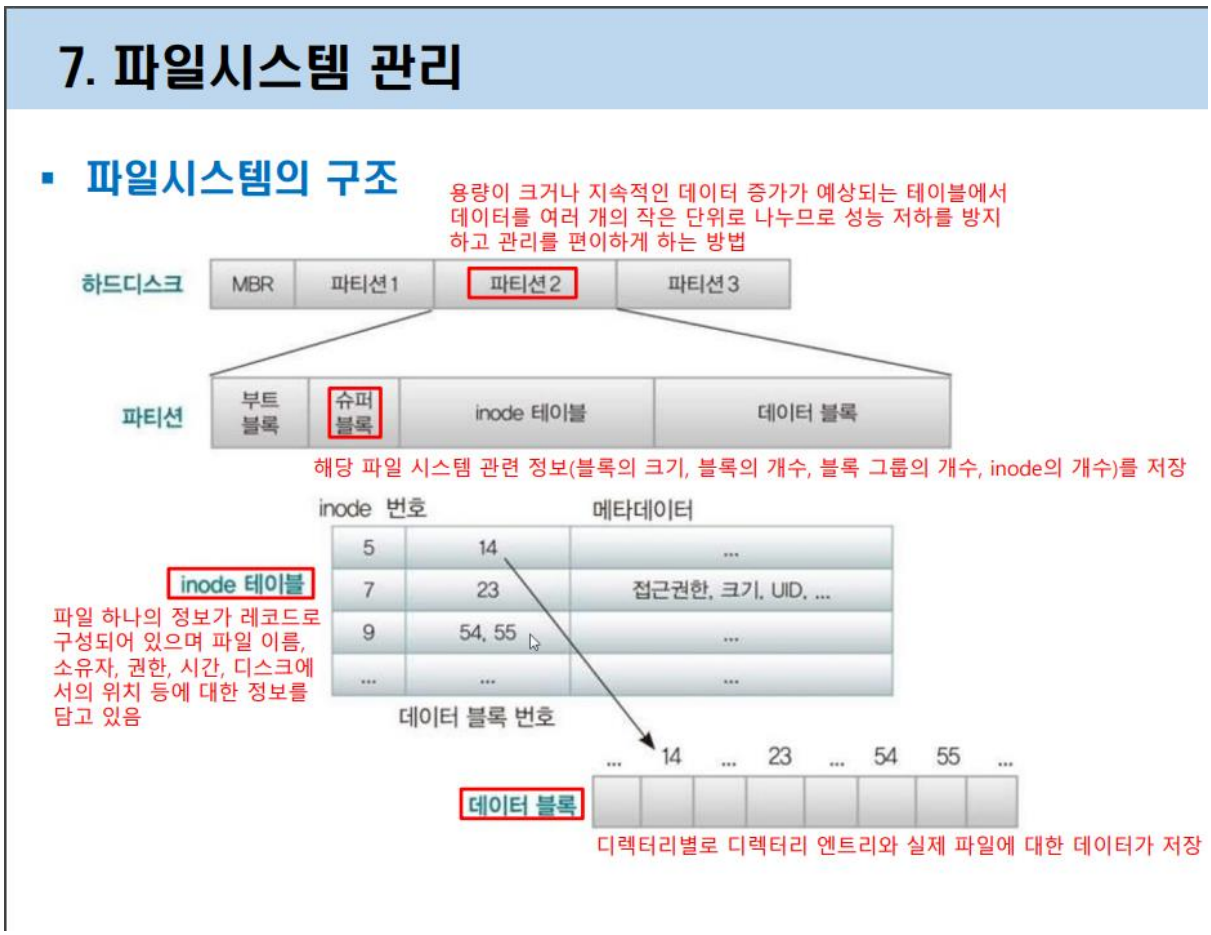
그리고 모든 곳에서 권한이 없는 것은 – 하이픈이다.

보안에 가장 좋은 것은 소유자에게만 모든 권한을 주는 것이다. Umask 077

리눅스 파일시스템

데이터를 효율적으로 저정 및 관리하기 위해 계층적 구조로 구성된다.

용량이 크거나 데이터 증가가 예상되는 테이블에서 데이터를 여러 개의 작은 단위로 나눠 저장하여 성능 저하를 방지하고 관리를 편하게 하기 위한 방식이다.



📊 주요 구성 요소

구성 요소	설명
MBR (Master Boot Record)	디스크의 첫 번째 섹터, 부트로더 및 파티션 테이블 정보 저장
파티션	디스크를 논리적으로 나눈 영역
부트 블록	부트로더를 저장
슈퍼블록 (Super Block)	해당 파일시스템의 정보 (블록 크기, 블록 수, 블록 그룹 수, inode 수 등) 저장
inode 테이블	파일 하나하나에 대한 정보 (파일 이름, 소유자, 권한, 생성시간, 수정시간, 디스크 블록 위치 등)를 저장
데이터 블록	실제 데이터나 디렉터리 엔트리를 저장하는 공간

파티션은 지속적으로 용량이 증가되는 테이블을 위해 성능 저하 및 관리 용이 목적으로 나누는 것이다.

슈퍼블록은 파일 시스템에 대한 정보가 나타난다.

Inode는 파일 하나하나에 대한 정보가 레코드로 저장된다. 이 개수는 슈퍼블록에 포함되어 있다.

데이터 블록은 실제 데이터나 디렉터리 엔트리를 저장한다.

파일 시스템의 종류

들어가기 앞서 저널링 기술에 대해 알아야 한다. 데이터를 디스크에 쓰기 전 별도 로그에 데이터를 남겨 놓는 기술이다.

이 기술은 파일 시스템을 검사하고 수리하는 fsck(파일시스템 무결성 검사 도구)에 걸리는 시간을 단축하기 위해 로그에 데이터를 남겨 비정상적 종료 및 에러에도 로그에 남은 데이터를 사용해 빠르고 안정적인 복구 기술을 제공하는 기술이다.
Ext1 : 레미 카드가 개발한 것 1992년 4월 리눅스 커널 0.96c에 포함되어 발표, 초기 리눅스 파일 시스템 inode 수정, 데이터 수정 시간 지원 x, 링크드 리스트로 파일 시스템 구성하여 복잡해지고 파편하되는 문제, 현재는 사용되지 않는다.

Ext2 : ext1 후속 파일 시스템, 안정성 있는 파일 시스템 제공, 레미 카드가 1993년 1월에 개발하여 발표했다. 최대 크기는 2TB이지만 현재는 이론적으로 32TB까지 지원된다.

지금도 부팅 가능한 usb 플래시 드라이브와 기타 ssd 장치에서 사용되고 있다. Ext3 도입 전까지는 표준 파일 시스템으로 사용되었다.

Ext3 : 저널링 기술이 적용된 2001년 11월에 리눅스 커널 2.4.15에 추가 되었다.

Ext2를 기반으로 개발하여 호환이 가능했다. Ext2 파일 시스템을 별도 변경 없이 바로 ext3 파일 시스템에 이식할 수 있었다. ACL(Access Control List)를 통한 접근 제어를 지원했다. 단점은 inode 동적 할당, 다양한 블록 크기와 같은 최신 파일 시스템 기능 부족, 온라인 조각 모음 기능이 없다는 것. 2~32TB까지 지원된다.

Ext4 : 2008년 12월 25일 리눅스 커널 2.6.28에 포함되어 공개, 1EB 이상의 볼륨과 16TB 이상의 파일을 지원하며 2, 3과도 호환성을 유지하고 있다. 서브 디렉터리 수가 32,000개에서 2배 늘어났고, 온라인 조각 모음 기능도 지원한다.

Xfs : 1993년 실리콘 그래픽스가 개발한 고성능 저널링 파일 시스템으로 2000년 6월 GNU GPL로 공개되었다. 2001년 리눅스에 이식되었고, 64비트 파일 시스템으로 최대 16EB까지 지원한다. 로키 리눅스는 이 파일 시스템을 기본으로 사용하고 있다.

7. 파일시스템 관리

▪ 저널링 기술의 특징

① 데이터를 디스크에 쓰기 전에 별도의 로그에 데이터를 남겨 놓는 기술

② 저널링 기술이 적용된 파일시스템은 ext3, ext4, XFS, JFS, ResierFS 등이 있음

. XFS(eXetended File Syatem) : 고성능 저널링. SGI에서 개발. 신속한 복구 기능

. JFS(Journaling File Syatem) : IBM사에서 개발한 저널링 파일시스템

. ReiserFS : 독일의 한스라이저가 개발. 리눅스용 파일시스템 중에 가장 안정적

③ 전원공급 문제나 시스템 오류와 같은 상황에 복구가 가능함

④ fsck(파일시스템 무결성 검사도구) 로 복구하는 것 보다 속도가 빠르고, 복구의 안정성도 뛰어남 ※ fsck : 파일 시스템을 검사하고 수리

저널링 기술은 기존의 fsck에 걸리는 시간을 단축하기 위해 데이터를 디스크에 쓰기 전에 로그(log)에 데이터를 남겨 시스템의 비정상적인 종료에도 로그를 사용해 빠르고 안정적인 복구 기능을 제공하는 기술

Mount -t 옵션은 파일시스템의 유형을 결정한다. 이 점 기억해야 한다. mkfs에서도 동일

-o 옵션은 추가 설정으로 noatime은 파일이 변경되기 전까지 access time이 변경되지 않는다.

▪ 다양한 장치 mount의 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 SUB 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

fdisk : 새로운 파티션 생성, 기존 파티션 삭제, 타입 결정 등의 작업을 수행한다. 한번에 한 디스크에 대해서만 작업을 수행한다. 어떤 디스크의 파티션을 변경할지 알려주어야 한다. 여러 개의 하드디스크를 설치 할 수 있기 때문이다.

▪ 명령어 fdisk

- **새로운 파티션의 생성, 기존 파티션의 삭제, 파티션의 타입 결정 등의 작업을 수행하며, 한번에 한 디스크에 대해서만 작업을 수행**
- **1대의 서버에 여러 개의 하드디스크가 설치될 수 있으므로 어떤 디스크의 파티션을 변경할 것인지 알려 주어야 함**

[형식] fdisk [옵션] [장치명]
[옵션] -v : fdisk 버전 정보 표시
-l : 현재 디스크의 파티션 테이블 정보 표시
-s : 지정된 파티션의 크기를 블록 단위로 표시
[주요 명령어] p : 디스크 정보 표시
n : 파티션 생성
t : 파티션 속성(Swap, RAID 등) 지정
d : 파티션 삭제
w : 변경된 파티션 정보 저장
q : 파티션 설정 작업 종료

Mkfs : fdisk로 하드디스크 파티션을 나눈 후 해당 파티션에 맞는 파일시스템을 생성한다.

해당 파티션의 마운트 정보와 파일 시스템 정보를 확인한다. -t 옵션 뒤에는 파일시스템 타입이 나와야 한다.

- **명령어 mkfs**
  - fdisk로 하드디스크 파티션을 나눈 후 해당 파티션에 맞는 파일시스템 생성
  - 해당 파티션의 마운트 정보와 파일 시스템 정보를 확인

  [형식] mkfs [옵션] 장치명

  [옵션] -V : 자세한 정보 보기
  -t 파일시스템 : 생성할 파일시스템 타입(ext2, ext3, ext4 등)을 지정
  -c : 파일시스템을 생성하기 전에 배드블록(Bad Block)을 검사
  -l 파일명 : 지정된 파일명으로부터 배드블록(Bad Block) 목록 읽기
  -v : 작업상태와 결과를 자세히 보기

- **명령어 mke2fs**
  - ext2, ext3, ext4 타입의 리눅스 파일시스템을 생성하는 명령어
  - fdisk로 파티션 작업을 한 후에 mke2fs 또는 mkfs 명령어로 파일시스템을 생성해야 함

  [형식] mke2fs [옵션] 장치명

  [옵션] -t : 파일시스템 타입 지정(ext2, ext3, ext4 등)
  -b : 블록 크기를 바이트 수로 지정
  -f : 프래그먼트 크기 지정
  -i : inode 당 바이트 수를 지정
  -j : 파티션을 저널링 파일시스템 ext3으로 지정
  -R : RAID4 장치를 포맷할 때 사용하는 특수 옵션

  [예시] 파티션 /dev/sda2을 파일 시스템 ext3로 생성
  # mke2fs -j /dev/sda2
  # mkfs.ext3 /dev/sda2
  # mke2fs -t ext3 /dev/sda2
  # mkfs -t ext3 /dev/sda2

7. 파일시스템 관리

- **명령어 fsck**
  - 파일시스템의 무결성을 점검하고 대화식으로 복구하는 명령어
  - 디렉터리 /lost+found는 fsck에서 사용하는 디렉터리
  - fsck 명령은 손상된 디렉터리나 파일을 수정할 때 임시로 /lost+found 디렉터리에 작업을 수행하고 정상적인 복구가 되면 사라짐

  [형식] fsck [옵션] 장치명

  [옵션] -A : /etc/fstab에서 모든 파일시스템을 점검
  -a : 질의 없이 자동 복구 (오류 발견 시 자동으로 복구를 시도)
  -r : 질의 후 복구 (복구 시도 전에 확인을 요청)
  -s : fsck 동작을 시리얼화, 대화형 모드에서 여러 파일시스템 점검 시 유용
  -t 파일시스템 : 점검할 파일시스템 유형 지정

du(Disk Usage) 디스크 별 사용량 확인하는 명령어

7. 파일시스템 관리

- **명령어 du (Disk Usage)**
  - 디렉터리별로 디스크 사용량을 확인

  [형식] du [옵션] [파일 및 디렉터리명]

  [옵션] -h : 용량 단위(KB, MB, GB)로 표시
  -a : 디렉터리가 아닌 모든 파일에 대한 정보 표시
  -m : 결과 값을 MB 단위로 표시
  -k : 결과 값을 KB 단위로 표시 (기본값)
  -s : 사용량의 총 합계만 표시 (파일의 전체 크기를 합한 값으로 표시)
  -c : 모든 파일의 디스크 사용 정보를 보여주고 나서 합계를 표시

  [예시] du -sh /* : 디렉터리별 크기를 KB, MB, GB 등의 단위로 출력

```
[linux@localhost ~]$ du -sh /home/linux
58M     /home/linux
```

▪ 명령어 df (Disk Free)

- 시스템에 마운트된 하드 디스크의 용량을 파티션 단위로 확인하는 명령어
- 기본적으로 1KB 블록 단위로 출력하며 옵션을 통해 변경이 가능

[형식] df [옵션] [파일명]

[옵션] -h : 용량 단위(KB, MB, GB)로 표시
-T : 파일시스템의 종류(유형과 파티션 정보)를 출력
-t : 표시되는 파일시스템 유형을 지정
-a : 0 블록의 파일시스템을 포함하여 모든 파일시스템을 출력
-k : --block-size=1K와 같은 의미
-i : inode 사용률 확인, 사용 공간, 사용 퍼센트를 출력

파일시스템 종류별 용량 표시

```
[linux@localhost ~]$ df -hT
Filesystem          Type      Size  Used Avail Use% Mounted on
devtmpfs            devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs               tmpfs     1.8G     0  1.8G   0% /dev/shm
tmpfs               tmpfs     726M   12M  715M   2% /run
/dev/mapper/rl-root xfs        36G  5.0G   31G  15% /
/dev/nvme0n1p1      xfs       960M  409M  552M  43% /boot
tmpfs               tmpfs     363M  100K  363M   1% /run/user/1000
```

파일 /etc/fstab은 리눅스에서 사용하는 파일시스템 정보를 정적으로 저장하고 있는 파일이다. 리눅스 파일시스템 정보와 부팅 시 마운트 정보를 가지고 있다.

```
[linux@localhost etc]$ cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sat Mar  9 20:56:49 2024
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/rl-root     /                       xfs     defaults        0 0
UUID=5e9655ad-8432-4623-9f74-0fb8906882a2 /boot                   xfs     defaults        0 0
/dev/mapper/rl-swap     none                    swap    defaults        0 0
```

디바이스 | 마운트지점 | 파일시스템 유형 | 옵션 | 덤프여부 | 파일시스템 검사여부

필드	내용
디바이스	장치의 이름으로 디바이스 파일(예를 들어 **/dev/sda1**) 또는 **UUID(universally unique identifier)** 가 사용됨 SCSI 디스크의 첫번 째 파티션
마운트 지점	파일 시스템 트리에서 디바이스가 부착되는 위치(디렉터리)
파일 시스템 유형	리눅스에서 허용하는 파일 시스템의 유형
옵션	마운트 옵션으로 **mount** 명령에서 **-o** 옵션을 사용하는 것과 의미가 같음
덤프 여부	백업을 위한 것으로 **0**은 덤프를 하지 말라는 것이며, **1**은 하라는 것
파일 시스템 검사 여부	파일 시스템 검사를 위한 것으로 **0**은 검사를 하지 말라는 것이며, 루트(**/**) 파일 시스템의 경우는 **1**일 때, 나머지 파일 시스템의 경우에 **2**일 때 검사를 함

수동으로 마운트

* UUID : 16진수 32개로 파티션을 식별하는 숫자 (운영체제에서 자동으로 부여)

```
UUID=5e9655ad-8432-4623-9f74-0fb8906882a2 /boot                   xfs     defaults        0 0
```

▪ 파일 /etc/fstab 설정 옵션

참고만 시험 안 나옴

속성	의미
defaults	일반적인 파일 시스템에 지정하는 속성이다. rw, nouser, auto, exec, suid 속성을 모두 포함한다.
auto	부팅 시 자동으로 마운트한다.
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.