File System & Memory Swap

2020 Summer Wheel Seminar

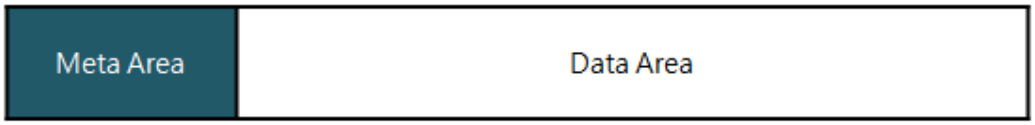
jessie

1. File System

File

- 사용자 입장:
  - 이름이 있는 byte sequence
  - 내가 이름으로 찾을 수 있는 정보
- File System 입장:
  - 하드웨어 속의 disk block 모음
  - disk block: 정보를 저장하는 단위

File System



- 데이터를 저장, 불러오기 위해 파일을 관리하는 체계
- 파일 시스템이 없다면. 모든 정보는 저장 장소 안에서 각각 시작과 끝을 알 수 없는 데이터 더미 뿐
- 사용자 입장: Manages files and data stored in files
- File system 입장: Map name & offset to disk block
- OS의 일부 (OS 마다 다름)
  - 같은 데이터이지만, 그 데이터를 어디에 저장하며, 어떻게 읽고 쓸 것인가
- Storage (persistent memory)를 두 부분으로 분류
  - Meta Area: 파일의 이름, Data Area 내에서의 위치, 크기, 시간 정보 등 저장 (Ex. Finder)
  - Data Area: 파일의 실제 내용 저장
  - ex) 내가 a.txt를 만든다면?

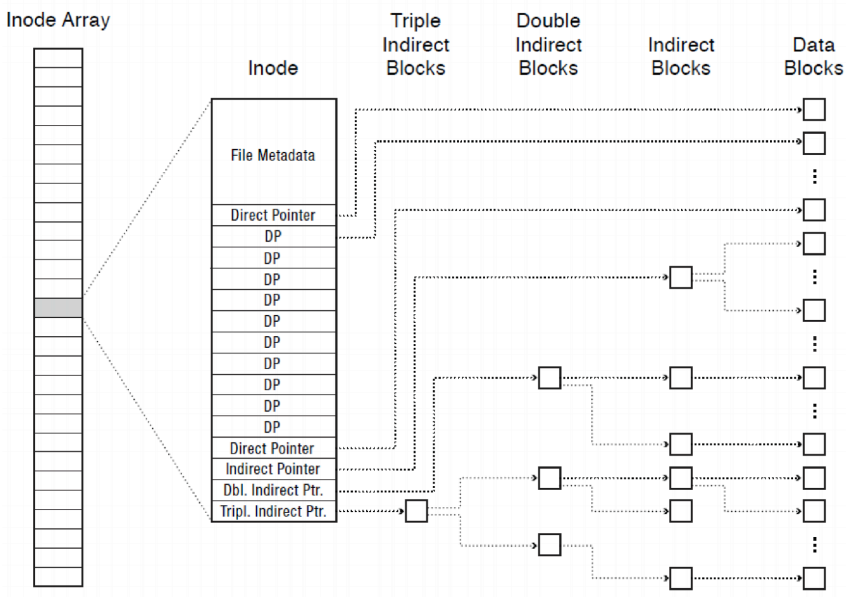
Block & Sector

- Sector:
  - 하드 디스크에서 데이터를 저장하는 최소 단위
  - 일반적으로 512 byte, 최근에는 4096 byte으로 확장
- Block:
  - 파일 시스템에서 파일을 저장하는 최소 단위
  - Sector의 정수 배 크기
    - 일반적으로 4KB (8 * 512-byte sectors)
  - 파일 시스템에 따라서 Block 크기 조절 가능

파일 시스템 원리 1

- 모든 파일은 inode block 1개와 data block 0개 이상으로 이루어져 있음
- Inode block
  - 파일에 대한 metadata
  - 파일 이름, 크기, permission, owner 등
  - data block의 위치
  - storage 한쪽에 모든 inode 들이 array로 저장
- Data block
  - 파일이 실제 담고 있는 데이터
  - inode를 통해서 접근

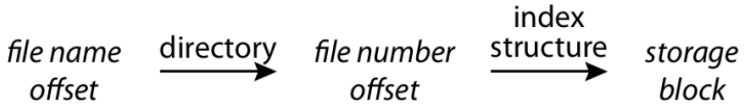
ex) FFS의 구조



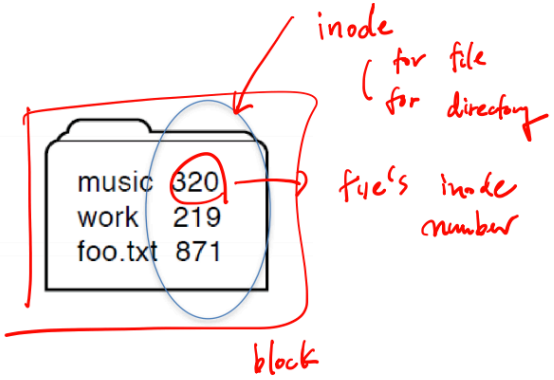
권용진 교수님, 2020 Spring CS330

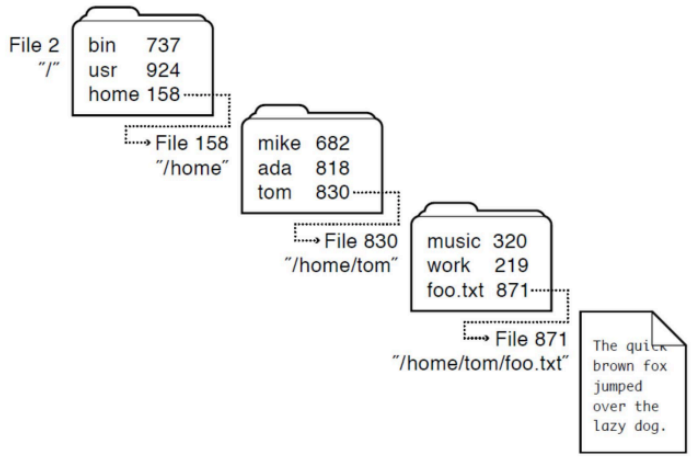
파일 시스템 원리 2

- 디렉토리는, (파일 이름, inode number) mapping을 저장하는 파일
  - 디렉토리도 파일로 취급

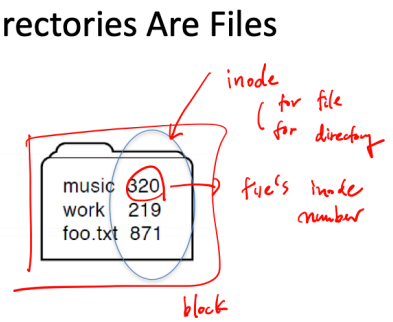


Directories Are Files

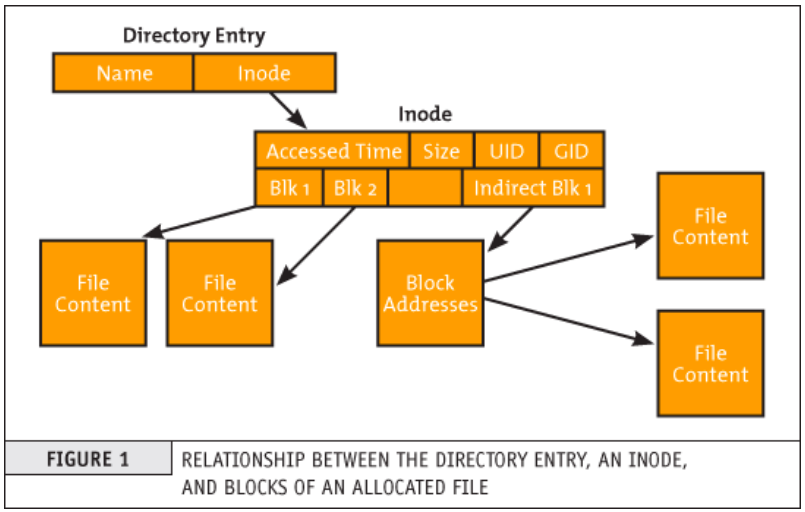


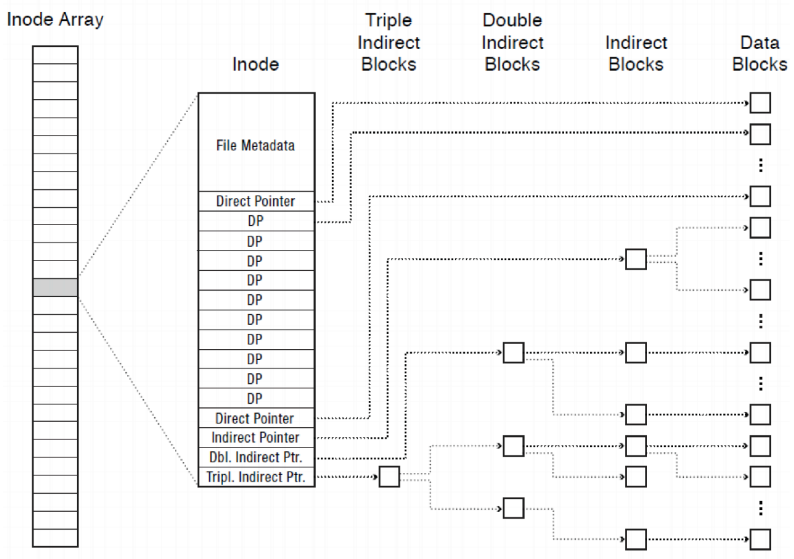


파일 시스템 원리 2



- 디렉토리는, (파일 이름, inode number) mapping을 저장하는 파일
  - 디렉토리도 파일로 취급





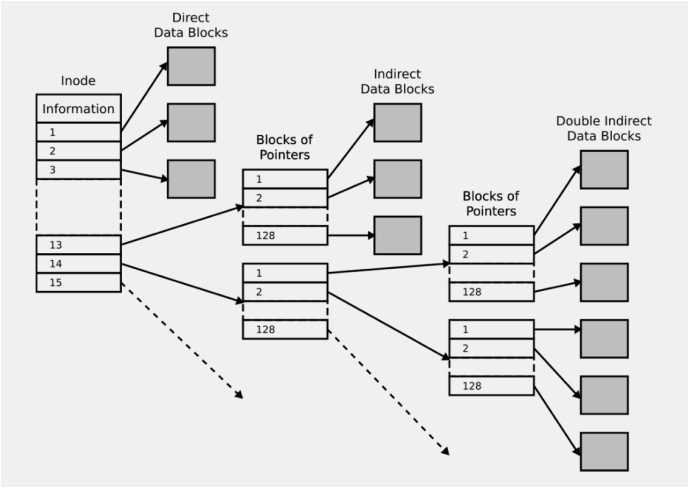
https://linuxbyimraan.co.za/2020/01/22/inodes-on-linux-explained/

Types of file systems

- Window: FAT16, FAT32, NTFS
- Linux: Btrfs, EXT2, EXT3, EXT4, ReiserFS, XFS
- MacOS: HFS/HFS+, APFS
- 차이점: inode에 data block을 저장하는 방식 (linked list vs indexing), inode에 저장되어 있는 메타데이터, 기타 기능 지원 (journaling, extent) 등..
- 종류가 엄청 많음
- 참고:
  - https://en.wikipedia.org/wiki/Comparison_of_file_systems

Linux file systems

- EXT (Extended File System)
  - Linux에서 지원하는 파일 시스템: 많은 배포판들에서 쓰임
- EXT2 – 과거에 널리 사용
- EXT3 - Journaling 지원 (하위 호환)
- EXT4 - Extents -> 현재 제일 흔함
- XFS - Journaling, Large on memory cache (Good performance) -> 특정 경우에 사용



Journaling의 필요성

- 파일은 하드 디스크(storage)에 있음
- 파일을 읽고 수정할 때는, 파일의 내용이 메모리로 복사된 후에 메모리를 읽고 수정하는 것임
- 따라서 메모리에서 수정된 파일을, 다시 디스크로 복사해야 함
- 메모리에서 수정된 파일을 디스크에 쓰던 중에 갑자기 컴퓨터가 꺼지면…?
  - 메모리(RAM)는 전원이 꺼지면 날라감
  - 디스크에 저장된 그 파일의 일부는 최근 버전, 일부는 옛날 버전이 됨
  - 이를 inconsistency라고 함
  - 있어서는 안되는 사태!

Journaling

- Inconsistency를 방지하기 위한 장치
- 파일 시스템에 가해진 변경 사항들을 반영하기 위해서:
- 하드디스크에 데이터를 쓰기 전에, 이 변경 사항들을 미리 기록해 두고 나서 씀으로써 파일 시스템의 복구를 쉽게 만들어 줌
- 데이터를 저장하기 전에, 하드디스크의 journal 영역에 변경 이력을 먼저 쓰고, 그 다음에 실제로 저장하기.
- EXT2, FAT16/32는 제외 (jounaling 파일 시스템이 아님)

EXT3에서 지원하는 Journaling

- Modes of Journaling (각각 장단점이 있음) :
  - 차이점: 저널에 metadata와 data 둘다 혹은 하나만 쓰이는지, journal과 storage에 저장되는 순서
- Journal
  - 데이터와 메타데이터가 저널에 먼저 쓰인 다음에 주 파일 시스템에 쓰인다. 가장 신뢰할만한 방식이지만 데이터가 두 번 쓰 이기 때문에 속도가 느릴 수 있다.
- Ordered
  - 메타데이터만 저널에 쓰인다. 파일은 쓴 후에 저널에 반영 하기 때문에 파일 쓰는 도중에 오류가 나면 확인이 가능하다.
- Write-back
  - 메타데이터만 저널에 쓰인다. 저널에 쓰기 전 또는 후에 파일 데이터가 쓰이기 때문에 신뢰성이 가장 떨어질 수 있다
- 파일 시스템을 mount할 때 journaling mode를 선택할 수 있음
  - mount option "data=[mode]”

Extents

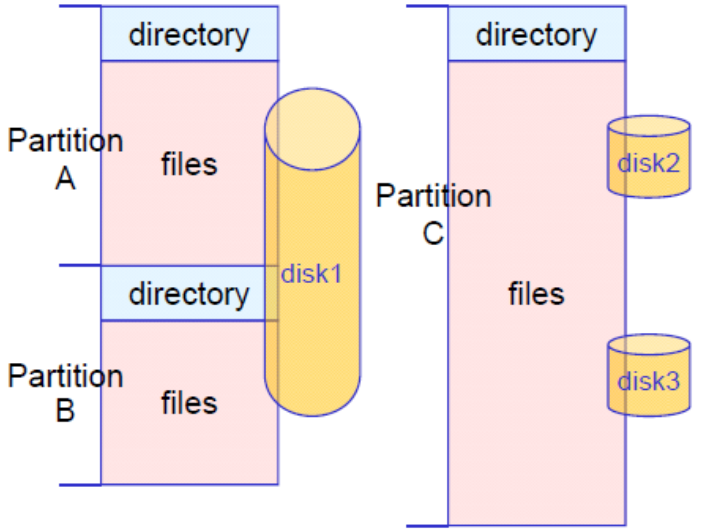
- 파일의 데이터 블록을 모두 저장하는 대신, 파일 데이터를 구간별로 연속적으로 블록들에 저장하여, 그 구간의 첫/마지막 블록만 저장
- 각 파일은 0개 이상의 구간 (extent)로 이루어짐
- fragmentation 감소
- inode에서 저장해야 할 블록 수 감소

EXT4

- 2008년 정식 출시
- EXT3 의 64비트 버전
- 대형 파일 시스템: 최대 1EB공간, 최대 16TB파일
- Extent 사용 (새로운 공간 할당 방법)
- 하위 호환성: EXT2, 3 도 지원
- 지연된 할당: allocate-on-flush
- 하위 디렉토리 제한 없음(기존 32000)

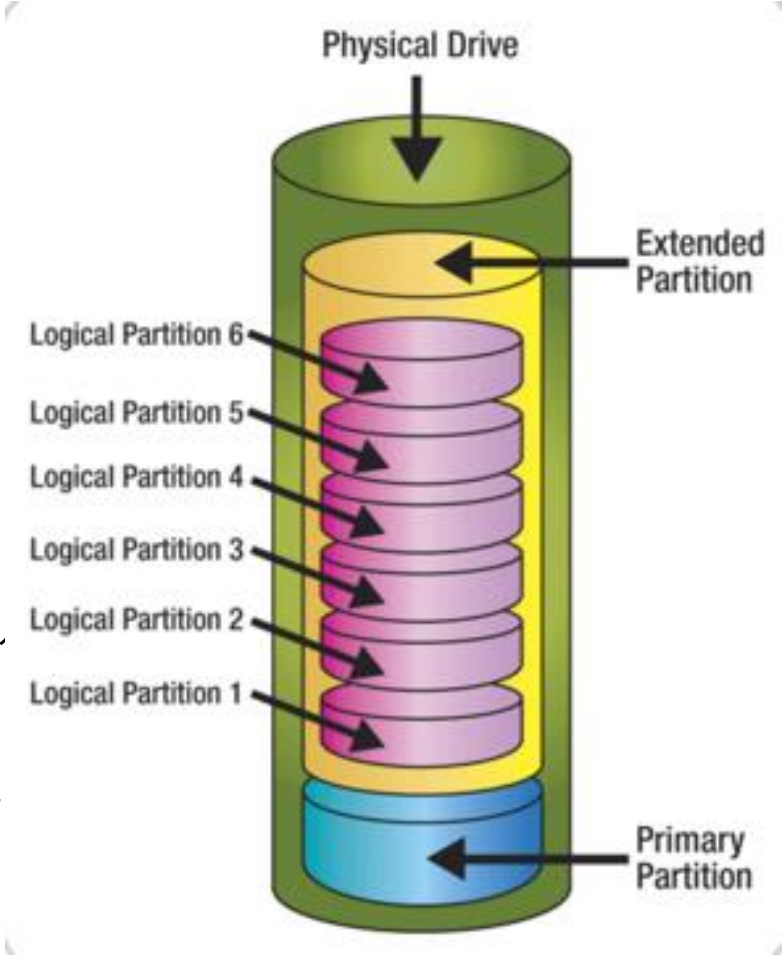
Partition

- 하나의 물리적인 디스크를 논리적으로 여러 개의 저장소인 것처럼 취급
- 각 파티션을 따로 관리할 수 있다
  - 한 파티션의 파일 시스템이 고장나더라도 다른 파티션에 영향을 주지 않는다
  - 각 파티션마다 따로 포맷 가능
- OS가 파티션을 요구할 수 있음
  - Window & Linux: partition 여러 개 필요
  - Mac: 1 partition 만 필요
- ex) Window:
  - System partition & data partition
  - 디스크 용량이 꽉 차서 system operation (booting 등)이 안되는 것을 방지



Partition 종류

- 물리적 파티션 (physical partition) [primary]
  - 실제 장치를 직접 나누는 파티션
  - 하나의 디스크에 최대 4개
  - 그 이상은 Logical Partition 이용
- 논리적 파티션 (logical partition) [extended]
  - Extended partition: 주 파티션과 다른 새로운 파티션을 생성할 수 있다고 영역만 그어놓은 파티션
  - Logical partition: 확장 파티션 안에 생성할 수 있는 파티션.

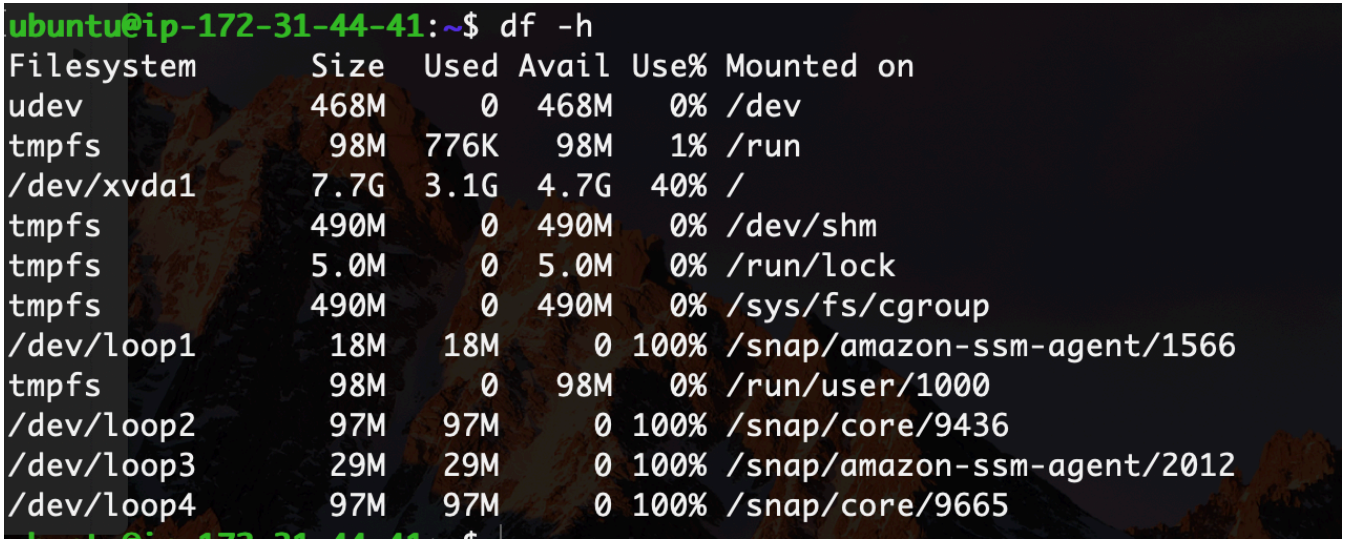


Types of partition table layouts

- partition 정보를 드라이브에 저장하는 방식
  - 각 partition의 시작과 끝, 어느 partition이 booting 가능한지 등
- MBR (Master Boot Record)
  - 디스크의 가장 첫 섹터에 위치한 512B 공간을 boot sector로사용
  - 여기에 OS를 위한 boot loader 및 이 드라이브의 logical partition 관련 정보 저장
  - 부팅 코드, 디스크 서명, 파티션
  - 최대 4개 파티션 제한, 최대 파티션 크기 2TB 제한
  - 옛날부터 사용되어 왔고, 아직까지 호환성 제일 좋음
- GPT (GUID Partition Table)
  - 모든 파티션은 Globally Unique Identifier (GUID)를 가짐
  - 부팅 정보가 여러곳에 중복 저장되어 있어 더 robust & 안전
  - 최대 8ZB, 최대 128개 파티션
  - 모든 파티션이 primary 파티션임
  - 장점이 많아서 요즘 많이 쓰이지만, 옛날 컴퓨터은 호환 안됨
- 점점 MBR에서 GPT로 갈아타는 추세이지만, 어쩔 수 없이 MBR를 써야할 때도 있음

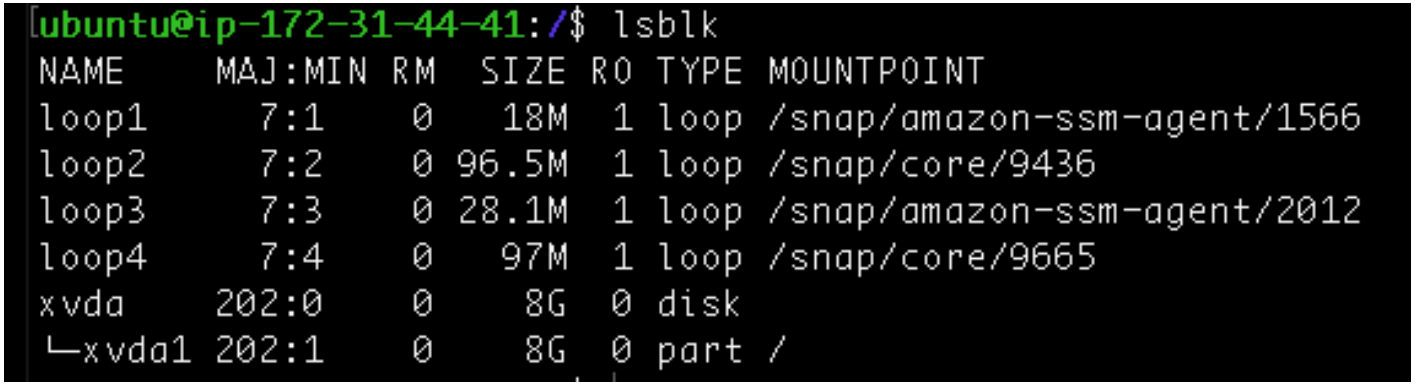
Linux 파일 시스템 관리

- 파티션 설정: fdisk
- 파일 시스템 생성: mkfs
- 장치 마운트/언마운트: mount/unmount
- 파일 시스템 점검/복구: fsck
- $df (-h): 리눅스가 이용하고 있는 파일 시스템들의 공간, 사용량, 마운트 포인트를 알 수 있다.



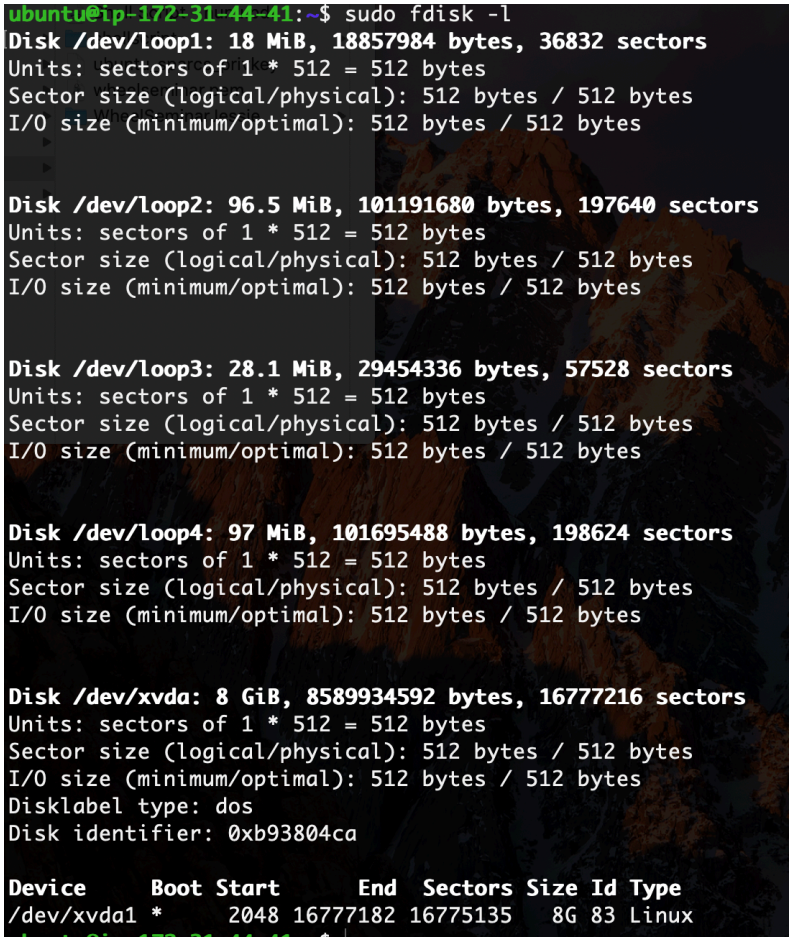
1. 파티션 설정

- Linux 시스템의 장치 파일들은 /dev 에 존재
  - IDE 형식의 하드디스크 연결 /dev/hda, /dev/hdb, /dev/hdc, ..
  - S-ATA 형식의 하드디스크 연결 /dev/sda, /dev/sdb, /dev/sdc,, ..
  - 물리적인 디스크가 /dev/had 라면, 파티션을 여러 개 만들 경우
    - /dev/hda1, /dev/hda2, ..
- $ lsblk: list block devices



1. 파티션 설정

- fdisk 사용하기 위해서는 앞에 sudo 붙여야 함
- $ fdisk [디스크장치] => 디스크에 대한 파티션 설정을 하는 command mode 진입 (ex) sudo fdisk /dev/sda
  - 대화형 인터페이스로 구성
  - p: 파티션 테이블 출력
  - n: 파티션 추가
  - d: 파티션 삭제
  - w: 파티션 테이블 저장하고 종료
  - q: 파티션 테이블 저장하지 않고 종료
  - m: 도움말
- • $ fdisk –l [디스크장치]
  - 디스크의 파티션 테이블을 보여줌.
  - [디스크 장치]를 생략하면 전체 파티션을 보여줌
- $ fdisk –s [파티션장치] => 파티션의 크기를 블록 단위로 보여줌.
- 비슷한 명령어: parted (주로 2TB 미만은 fdisk, 2TB 이상은 parted 사용)



2. 파일 시스템 생성

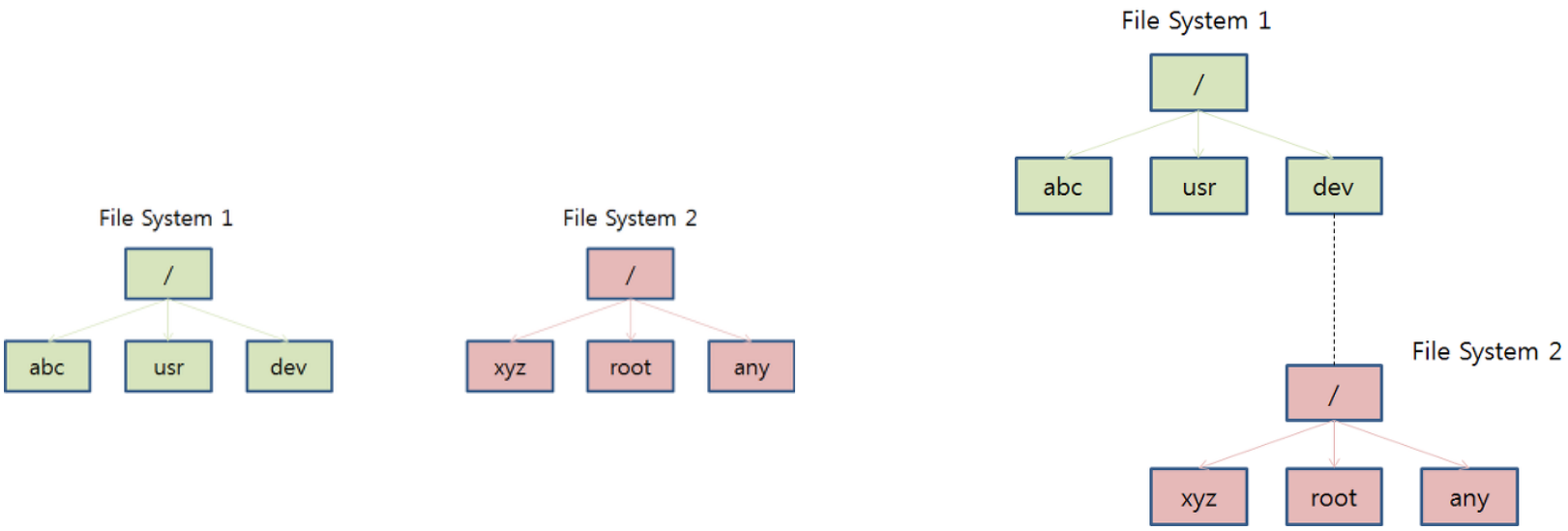
- 파일 시스템마다 고유의 mkfs 명령이 존재
  - ex) EXT3: mkfs.ext3
  - mkfs는 그저 명령어를 각각 실행시켜줌

- $ mkfs –t [파일 시스템] [파티션 장치]
  - default: ext2
  - mkfs –t ext3 [파티션 장치]는 mkfs.ext3 [파티션 장치] 와 같음

- $ mkfs –c [파티션 장치]
  - 배드섹터 검사를 진행
- 조합해서 mkfs [–c] [–t file_system_type] <partition_device>

3. 장치 마운트 / 언마운트

- 마운트 (mount) : 하나의 파일 시스템을 다른 파일 시스템의 어떤 location (directory) 에 붙이는 것.
  - ex) USB 인식
- 아래 예시에서는, 마운트 포인트가 /dev/



3. 장치 마운트 / 언마운트 - 수동

- $ mount => 현재 마운트 정보를 출력
  - $ mount | column –t 하면 더 예쁘게 출력
- $ mount -t [파일 시스템] [파티션 장치] [디렉토리]
  - 앞에서 만든 파티션의 파일 시스템을 써야겠죠?

```
$ mkfs –t [파일 시스템] [파티션 장치]
$ mkfs.ext3 [파티션 장치]
```

- $ umount [디렉토리명(mount point) or 장치명]
- $ mount –a
  - /etc/fstab 에서 auto 로 표시된 파일 시스템들을 마운트

3. 장치 마운트 / 언마운트 - 자동

- /etc/fstab
- fstab (file systems table): 파일 시스템 정보
  - <device> <mount point> <file system type> <options> <backup operation> <file system check order>
- 옵션
  - -auto (default): 부팅할때 자동으로 마운트
- mount 명령은 이 파일을 읽음

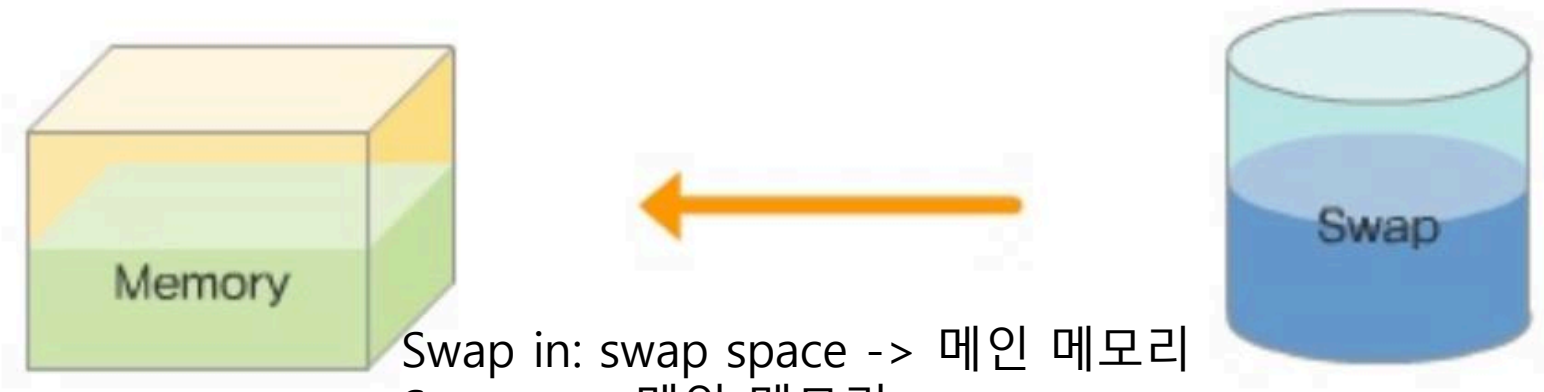
4. 파일 시스템 점검/복구

- $ fsck
  - file system consistency check
  - inconsistency 등이 의심될 때 부팅 때 자동으로 실행 됨
- $ fsck [–t 파일시스템 종류] [파티션 장치]
  - 파일 시스템이 정상인지 확인하고 오류가 있다면 복구한다.
  - ex) $ fsck –y /dev/sdb
  - -y: 자동으로 error correction 하기
- 반드시 unmount 한 후에 사용
  - mount된 filesystem에 사용하면 disk / data corruption 일어날 수 있음
- mkfs 처럼 파일 시스템마다 fsck 존재

Memory Swap

Swap Space

- 보조 저장 장치(disk)를 이용해 RAM의 부족을 해결한다.
- 주기억 장치(RAM)의 free 메모리의 수준이 어떤 임계값 이하로 떨어지게 되면,
  => 곧 엄청 많이 사용하려고 할 때
- 주기억 장치에 존재하는 일부 프로세스를 디스크의 스왑 영역으로 옮김.
  - 일반적으로 RAM 용량의 2배 정도로 설정한다.



Swap in: swap space -> 메인 메모리
Swap out: 메인 메모리 -> swap space

Swap Space 할당 방법 2가지

- Swap Partition
  - Swapping 만을 위해 사용되는 파티션
  - 다른 파일은 존재할 수 없음
  - contiguous disk blocks -> 속도 ☺
- Swap File
  - Swapping 을 위한 파일을 파일 시스템 내에 만들기
  - 파티션 안에 다른 파일들과 함께 존재
- 차이점: Swap Partition (4GB) vs 빈 파티션에 Swap File (4GB)
  - swap file 이 조각으로 저장될 때 느리다.
  - 일반적으로 swap partition 을 사용하며, 임시로 추가할 때 swap file 사용

Swap Space 관리

1. 메모리상태확인
    1. free
2. 스왑 파일/파티션 만들기
    1. dd/fdisk
3. 스왑공간만들기
    1. mkswap
4. 스왑 공간의 활성화/비활성화
    1. swapon/swapoff

여기부터 실습 시작!

1. EC2에서 Swap file 만들기
2. EBS로 Swap partition만들기

1. 메모리 상태 확인

- free: 시스템의 메모리 사용 현황을 출력 (KB)
- total/used/free는이름그대로.
- shared: 주로 tmpfs 에 의해 사용되는 메모리 (임시 파일 저장 기능)
- buffers: 자주 사용되는 데이터의 정보(metadata)를 메모리에 캐쉬
- cached: 디스크의 페이지를 메모리에 캐쉬 해놓은 것(file data)
- 두번째 줄]: buffers + cached 까지 free 로 취급해서 계산한 것.

2. Swap File / Partition 만들기

- Swap Partition
  - 앞에서 배운 fdisk 로 파티션 생성!
- Swap File
  - $ dd if=/dev/zero of=[스왑파일 위치] bs=[블록크기] count=[파일 크기]
    - if: input file / of: output file / bs: block size / count: data size(KB)
    - /dev/zero 는 00000, bs 는 보통 1024(1KB)
  - $ chmod 0600 [스왑파일명]
    - 접근 권한 변경(root user만 읽고 쓰도록)
- [참고] 시스템 다운을 대비해 sync 해주면 좋다. 저장되지 않은 메모리의 데이터를 디스크에 저장.

Swap File 만들기

```
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      282916      568172         776      151216      572152
Swap:             0           0           0
ubuntu@ip-172-31-44-41:/$ sudo swapon --show
ubuntu@ip-172-31-44-41:/$ ls
bin   dev  home        initrd.img.old  lib64       media  opt   root  sbin  srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
ubuntu@ip-172-31-44-41:/$ sync
ubuntu@ip-172-31-44-41:/$
```

```
ubuntu@ip-172-31-44-41:/$ sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
1048576+0 records in
1048576+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 14.2437 s, 75.4 MB/s
ubuntu@ip-172-31-44-41:/$ sudo chmod 600 /swapfile
```

- https://linuxize.com/post/how-to-add-swap-space-on-ubuntu-18-04/

3. Swap Space 만들기

- mkswap: 주어진 partition 이나 file 을 swap space 로 포맷
  - $ mkswap –c [스왑파일명] [스왑영역크기] => -c 는 배드 블록 검사.

```
ubuntu@ip-172-31-44-41:/$ sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
1048576+0 records in
1048576+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 14.2437 s, 75.4 MB/s
ubuntu@ip-172-31-44-41:/$ sudo chmod 600 /swapfile
ubuntu@ip-172-31-44-41:/$ sudo mkswap -c /swapfile
mkswap: warning: checking bad blocks from swap file is not supported: /swapfile
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=d4806751-c0ea-4124-ad43-e5c6f55bb550
```

4. Swap Space 활성화/비활성화

- $ swapon [파티션장치/스왑파일명]
- $ swapoff [파티션장치/스왑파일명]
- $ swapon –a
  - =>/etc/fstab파일시스템테이블에서sw옵션값을갖는항목모두활성화(부팅시 자동으로 실행)
- $ swapon –s
  - 스왑파티션의 상태를 보여준다

```
ubuntu@ip-172-31-44-41:/$ sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
1048576+0 records in
1048576+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 14.2437 s, 75.4 MB/s
ubuntu@ip-172-31-44-41:/$ sudo chmod 600 /swapfile
ubuntu@ip-172-31-44-41:/$ sudo mkswap -c /swapfile
mkswap: warning: checking bad blocks from swap file is not supported: /swapfile
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=d4806751-c0ea-4124-ad43-e5c6f55bb550
ubuntu@ip-172-31-44-41:/$ sudo swapon /swapfile
ubuntu@ip-172-31-44-41:/$ swapon -s
Filename                                Type            Size    Used    Priority
/swapfile                               file            1048572 0       -2
ubuntu@ip-172-31-44-41:/$ free
               total        used        free      shared  buff/cache   available
Mem:         1002304      282432       85796         776      634076      559692
Swap:        1048572           0     1048572
ubuntu@ip-172-31-44-41:/$
```

제출하기 위해서 스크린 샷!

- swap file을 만든 후의 $ sudo swapon /swapfile 결과를 스크린 샷으로 제출
- 예시)

```
ubuntu@ip-172-31-44-41:/$ sudo swapon /swapfile
ubuntu@ip-172-31-44-41:/$ swapon -s
Filename                                Type            Size    Used    Priority
/swapfile                               file            1048572 0       -2
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      282432       85796         776      634076      559692
Swap:       1048572           0     1048572
```

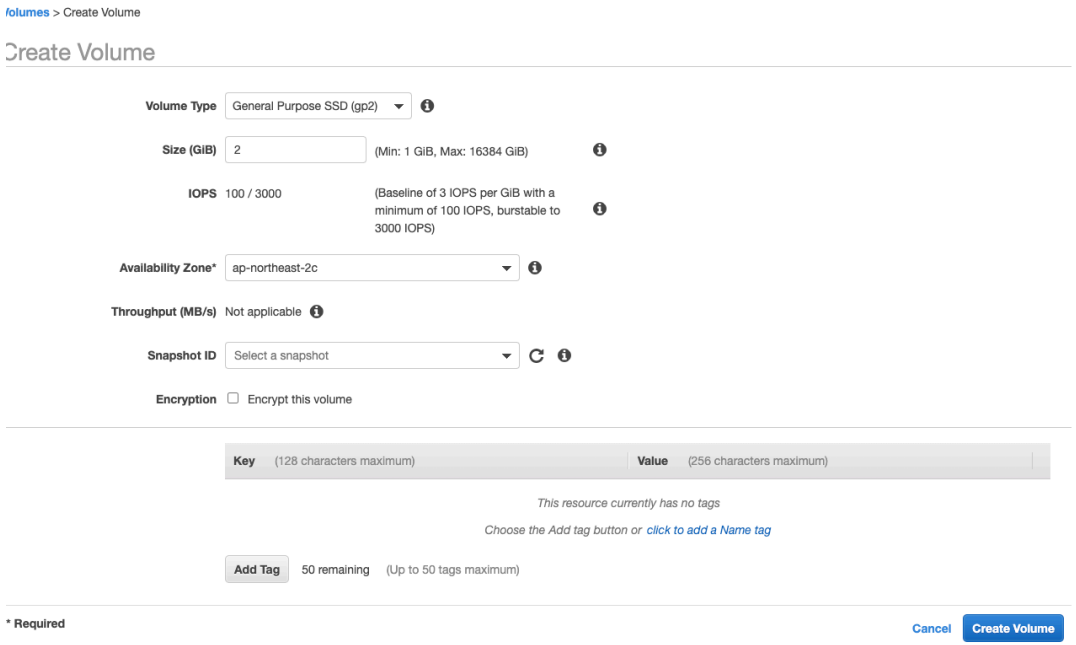
Swap file 지우기

- $ sudo swapoff -v /swapfile
- $ sudo rm /swapfile
- $ free (지우진 것 확인용)

```
ubuntu@ip-172-31-44-41:/$ sudo swapon /swapfile
ubuntu@ip-172-31-44-41:/$ swapon -s
Filename                                Type            Size    Used    Priority
/swapfile                               file            1048572 0       -2
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      282432       85796         776      634076      559692
Swap:       1048572           0     1048572
ubuntu@ip-172-31-44-41:/$ sudo swapoff -v /swapfile
swapoff /swapfile
ubuntu@ip-172-31-44-41:/$ sudo rm /swapfile
ubuntu@ip-172-31-44-41:/$ swapon -s
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      282476      589684         776      130144      583144
Swap:             0           0           0
ubuntu@ip-172-31-44-41:/$
```

실습: EBS로 EC2 instance의 swap partition 만들기

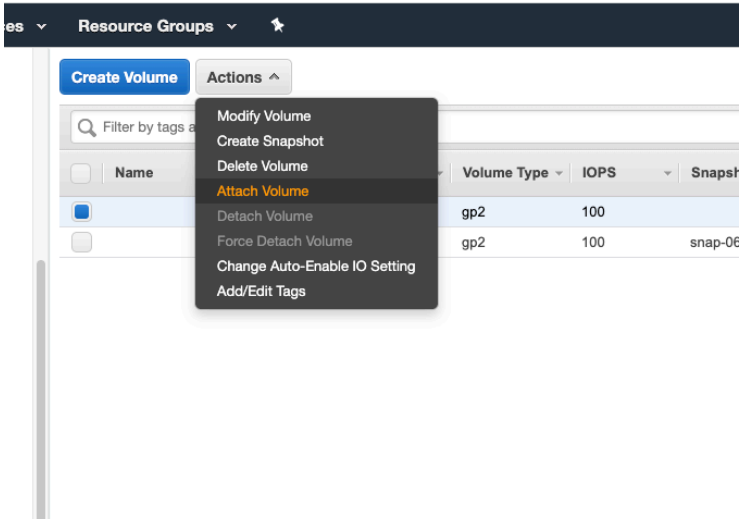
1. aws 접속 -> EC2로 들어가기
2. 왼쪽 메뉴에서 Elastic Block Store -> Volumes 선택
3. Create Volume
    1. General purpose
    2. 2 GiB
    3. Availability zone: EC2 instance 확인해서 같은 zone으로!!!

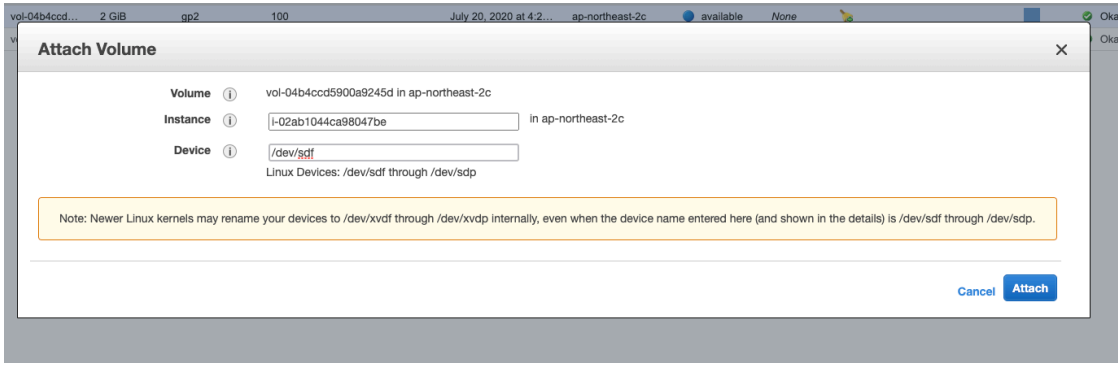


4. Attach EBS to EC2 instance

새로 만든 EBS 클릭하고 Actions -> Attach Volume

내 EC2 instance 선택





5. EBS 연결 확인: 터미널에서 $lsblk, 혹은 $sudo fdisk -l 했을때 xvdf 생성된 것 확인

```
ubuntu@ip-172-31-44-41:/$ lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop1     7:1    0   18M  1 loop /snap/amazon-ssm-agent/1566
loop2     7:2    0 96.5M  1 loop /snap/core/9436
loop3     7:3    0 28.1M  1 loop /snap/amazon-ssm-agent/2012
loop4     7:4    0   97M  1 loop /snap/core/9665
xvda    202:0    0    8G  0 disk
└─xvda1 202:1    0    8G  0 part /
xvdf    202:80   0    2G  0 disk
```

```
ubuntu@ip-172-31-44-41:/$ sudo fdisk -l
Disk /dev/loop1: 18 MiB, 18857984 bytes, 36832 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 /dev/loop2: 96.5 MiB, 101191680 bytes, 197640 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 /dev/loop3: 28.1 MiB, 29454336 bytes, 57528 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 /dev/loop4: 97 MiB, 101695488 bytes, 198624 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 /dev/xvda: 8 GiB, 8589934592 bytes, 16777216 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
Disklabel type: dos
Disk identifier: 0xb93804ca

Device     Boot Start      End  Sectors Size Id Type
/dev/xvda1 *     2048 16777182 16775135   8G 83 Linux


Disk /dev/xvdf: 2 GiB, 2147483648 bytes, 4194304 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
```

6. 새로 만든 disk 안에서 partition 하기

1. $ swapoff –a
   1. 전에 지정된 모든 swap 끄기
2. $sudo fdisk -l : disk 주소 확인
3. $sudo fdsk /dev/xvdf
   1. /dev/xvdf를 내 디스크 주소로 대체
   2. 이제 fdisk로 진입
4. p : 현재 디스크 확인
5. n: 새 디스크 만들기
   1. 모두 default로 두고 엔터 치면 됨
6. p : 새로 생성된 파티션 확인

```
ubuntu@ip-172-31-44-41:/$ sudo fdisk /dev/xvdf

Welcome to fdisk (util-linux 2.31.1).
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 disklabel with disk identifier 0xa289daf9.

Command (m for help): p
Disk /dev/xvdf: 2 GiB, 2147483648 bytes, 4194304 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
Disklabel type: dos
Disk identifier: 0xa289daf9

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

Using default response p.
Partition number (1-4, default 1):
First sector (2048-4194303, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-4194303, default 4194303):

Created a new partition 1 of type 'Linux' and of size 2 GiB.

Command (m for help): p
Disk /dev/xvdf: 2 GiB, 2147483648 bytes, 4194304 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
Disklabel type: dos
Disk identifier: 0xa289daf9

Device     Boot Start     End Sectors Size Id Type
/dev/xvdf1       2048 4194303 4192256   2G 83 Linux
```

6. 새로 만든 disk 안에서 partition 하기

1. 타입 id를 83 (linux)에서 82 (linux swap)으로 바꿔주기
    1. t : 방금 만든 파티션 선택
    2. 82 : 파티션의 타입을 82로 변경
    3. 모든 파티션 타입을 보기 위해서: l
2. p : id 바뀐 것 확인
3. w : 바뀐 partition table을 저장
4. $ sudo partprobe /dev/xvdf
    1. 커널이 partition table을 다시 읽어서 바로 적용되도록

```
Command (m for help): p
Disk /dev/xvdf: 2 GiB, 2147483648 bytes, 4194304 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
Disklabel type: dos
Disk identifier: 0xa289daf9

Device     Boot Start     End Sectors Size Id Type
/dev/xvdf1       2048 4194303 4192256   2G 83 Linux

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'.

Command (m for help): p
Disk /dev/xvdf: 2 GiB, 2147483648 bytes, 4194304 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
Disklabel type: dos
Disk identifier: 0xa289daf9

Device     Boot Start     End Sectors Size Id Type
/dev/xvdf1       2048 4194303 4192256   2G 82 Linux swap / Solaris

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

Swap Space로 format하기

- $ free로 현재 swap space 확인
- $ sudo mkswap –c /dev/xvdf1
  - -c: bad block 검사
- $ sudo swapon /dev/xvdf1
  - swap space 실행
- $ free로 달라진 swap space 확인

```
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      283088      394232         784      324984      559948
Swap:             0           0           0
ubuntu@ip-172-31-44-41:/$ mkswap /dev/xvdf1
mkswap: cannot open /dev/xvdf1: Permission denied
ubuntu@ip-172-31-44-41:/$ sudo mkswap /dev/xvdf1
Setting up swapspace version 1, size = 2 GiB (2146430976 bytes)
no label, UUID=2652e5b7-e77c-46ad-b275-4bfc337207fd
ubuntu@ip-172-31-44-41:/$ swapon /dev/xvdf1
swapon: cannot open /dev/xvdf1: Permission denied
ubuntu@ip-172-31-44-41:/$ sudo swapon /dev/xvdf1
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      284136      393128         784      325040      558900
Swap:       2096124           0     2096124
```

제출: 스크린 샷 2장

- swap file을 만든 후의 $ sudo swapon /swapfile 결과를 스크린 샷으로 제출

```
ubuntu@ip-172-31-44-41:/$ sudo swapon /swapfile
ubuntu@ip-172-31-44-41:/$ swapon -s
Filename                                Type            Size    Used    Priority
/swapfile                               file            1048572 0       -2
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      282432       85796         776      634076      559692
Swap:       1048572           0     1048572
```

- swap file 지우기 – 그냥 지우면 안됨! 아래 나온 순서대로 지우기

```
ubuntu@ip-172-31-44-41:/$ sudo swapoff -v /swapfile
swapoff /swapfile
ubuntu@ip-172-31-44-41:/$ sudo rm /swapfile
ubuntu@ip-172-31-44-41:/$ swapon -s
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      282476      589684         776      130144      583144
Swap:             0           0           0
```

- EBS swap partition을 만들고 난 후의 $ swapon –s, $ free, $lsblk 결과를 스크린 샷으로 제출

```
ubuntu@ip-172-31-44-41:/$ swapon -s
Filename                                Type            Size    Used    Priority
/dev/xvdf1                              partition       2096124 0       -2
ubuntu@ip-172-31-44-41:/$ free
              total        used        free      shared  buff/cache   available
Mem:        1002304      283360      393616         784      325328      559676
Swap:       2096124           0     2096124
ubuntu@ip-172-31-44-41:/$ lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop1     7:1    0   18M  1 loop /snap/amazon-ssm-agent/1566
loop2     7:2    0 96.5M  1 loop /snap/core/9436
loop3     7:3    0 28.1M  1 loop /snap/amazon-ssm-agent/2012
loop4     7:4    0   97M  1 loop /snap/core/9665
xvda    202:0    0    8G  0 disk
└─xvda1 202:1    0    8G  0 part /
xvdf    202:80   0    2G  0 disk
└─xvdf1 202:81   0    2G  0 part [SWAP]
```

-

감사합니다~!

레퍼런스

- 2019 Summer Wheel Seminar 파일 시스템과 스왑 영역 (hubo)
- 2019 Winter Wheel Seminar, File system & Swap area (tink)
- 권영진 교수님 CS330 자료