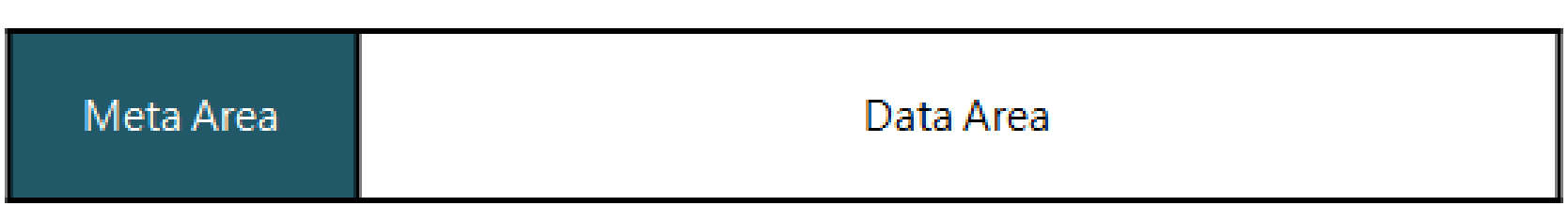
파일 시스템과 스왑 영역

2019 Wheel Seminar

HUBO

파일 시스템이란?

- 파일 시스템이란 데이터를 저장, 불러오기 위해 파일을 관리하는 체계
- 파일 시스템이 없다면, 저장 장소 안에 시작과 끝을 알 수 없는 데이터 더미 뿐.
- OS 별로 서로 다른 종류의 파일 시스템 => *중요* 같은 데이터이지만, 그 데이터를 어디에 저장하며, 어떻게 읽고 쓸 것인가



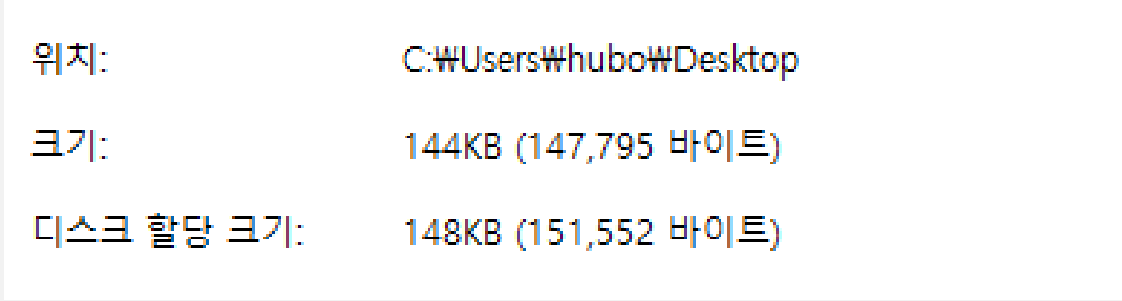
- Meta Area 에 파일의 이름, 위치, 크기, 시간정보 등이 저장. Ex) Finder

블록 & 섹터

- 섹터 Sector : 하드 디스크에서 데이터를 저장하는 최소 단위
  - 일반적으로 512 byte, 최근에는 4096 byte 으로 확장

- 블록 Block : 파일 시스템에서 파일을 저장하는 최소 단위
  - Sector 의 정수 배의 크기 (일반적으로 4KB)
  - 파일 시스템에 따라 Block 크기 조절 가능

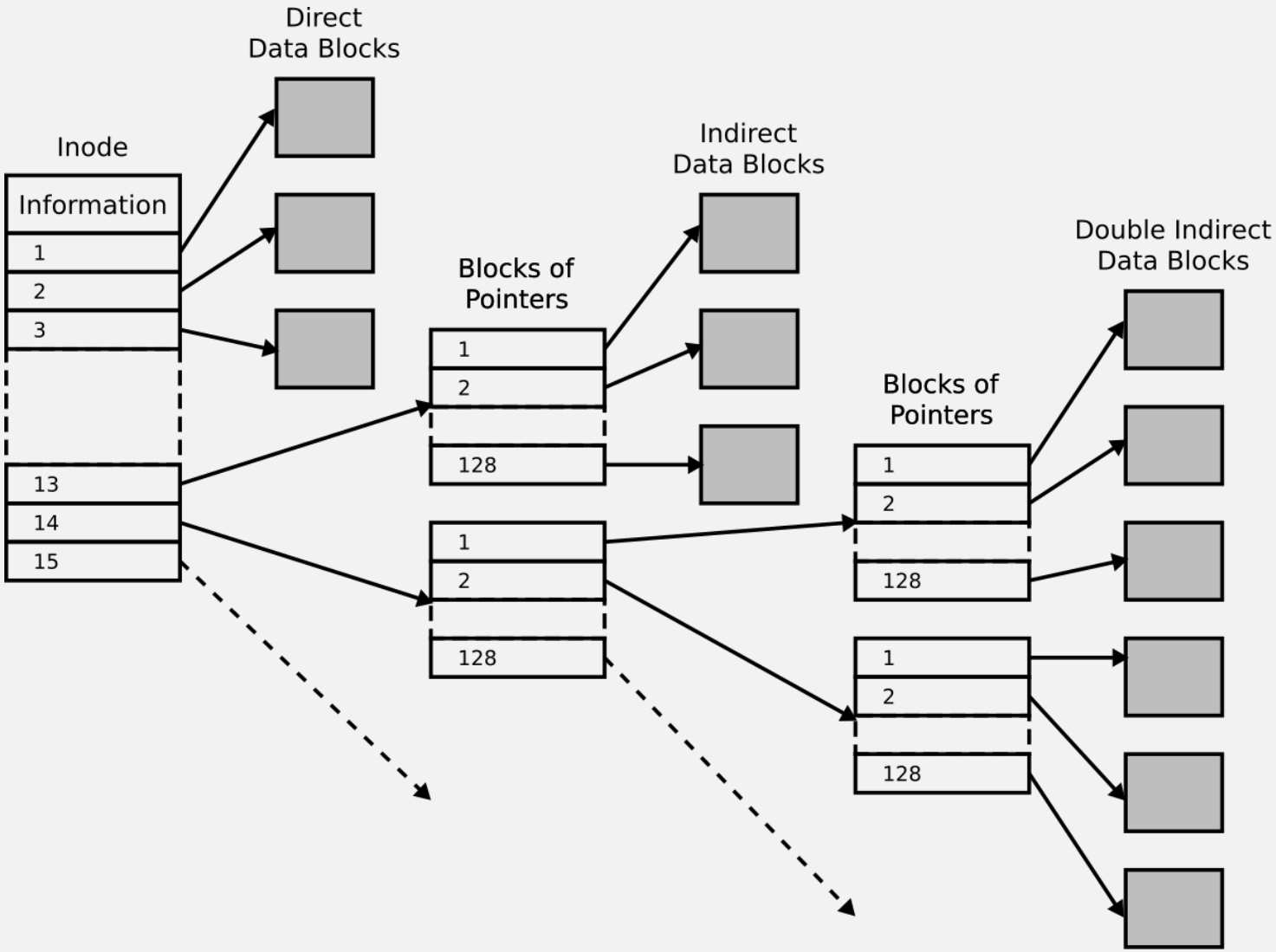
- 4KB 차이 나죠? =>



파일 시스템의 동작 원리

- 파일 시스템에는 i node block 과 data block 이 있다.
- i node(index node) 에는 파일의 metadata 와 data block 의 정보, data block 에는 실제 파일 내용이 들어 있다.
- I node 는 포인터처럼 data block 을 가리키는데, 저장 공간을 넓히기 위해...

파일 시스템의 동작 원리


Direct
Data Blocks
Inode
Information
1
2
3
13
14
15
Indirect
Data Blocks
Blocks of
Pointers
1
2
128
1
2
128
Double Indirect
Data Blocks
Blocks of
Pointers
1
2
128
1
2
128

파일 시스템의 종류

- Window: FAT16, FAT32, NTFS, ReFS
- Linux: EXT2, EXT3, EXT4, ReiserFS, Btrfs, XFS, F2FS
- Mac: HFS/HFS+, APFS

- https://en.wikipedia.org/wiki/Comparison_of_file_systems
- 종류 엄청 많음.

EXT

- Ext (extended file system)
- - Linux 에서 지원하는 파일 시스템 : 많은 배포판들에서 쓰임

- Ext2 과거에 널리 사용 (앞의 그림이 EXT2)
- Ext3 **저널링**을 지원하는 ext2 와 완전한 호환이 가능한 버전
  - 저널링이 뭐냐?

저널링

- 파일 시스템에 가해진 변경 사항들을 기록해 두는 기술로서, 파일 시스템의 복구를 쉽게 만들어준다.
  - (데이터를 저장하기 전에, Journal 영역에 변경 이력을 먼저 쓰고, 저장.)

- EXT2, FAT16/32 는 제외. 저널링 파일 시스템이 아니다.

  - Ex) UNIX 시스템에서 파일을 지우는 단계
    - 1. 입력된 데이터를 지우기
    - 2. 해당 디스크 부분을 'free'라고 마킹하기
  - 만일 실행 도중 crash 가 일어나 한가지만 수행된다면, 오류가 발생할 수 있다. Journal 을 참고하면 복구가 쉽겠죠? (완전한 복구는 아님)

EXT3 에서 지원하는 저널링

- 지원하는 저널링의 종류:
- Journal: 데이터와 메타데이터가 저널에 먼저 쓰인 다음에 주 파일 시스템에 쓰인다. 가장 신뢰할만한 방식이지만 데이터가 두 번 쓰이기 때문에 속도가 느릴 수 있다.
- Ordered: 메타데이터만 저널에 쓰인다. 파일은 쓴 후에 저널에 반영하기 때문에 파일 쓰는 도중에 오류가 나면 확인이 가능하다.
- Writeback: 메타데이터만 저널에 쓰인다. 저널에 쓰기 전 또는 후에 파일 데이터가 쓰이기 때문에 신뢰성이 가장 떨어질 수 있다.

EXT4

- EXT3 의 64비트 버전
- - 대형 파일 시스템: 최대 1EB 공간, 최대 16TB 파일
- - Extent: 새로운 공간 할당 방법
- - 하위 호환성: ext2, 3 도 지원
- - 지연된 할당: allocate-on-flush
- - 하위 디렉토리 제한 없음(기존 32000)

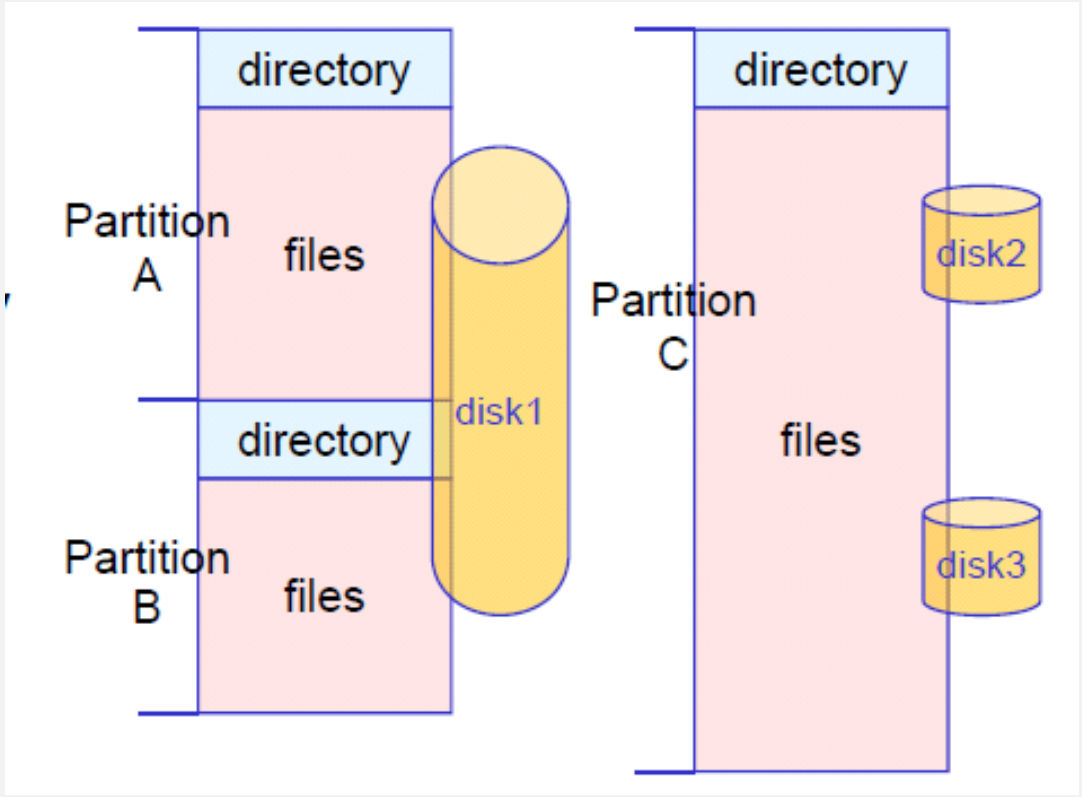
- https://blog.soobinpark.com/142

EXT4

- Extent: 연속적인 다수의 블록으로 저장.
  - 기존에는 I/O 시스템의 한계로, 한 블록을 단위로만 저장.
- 할당 지연 기능: 실제 파일크기에 따라 블록할당을 하여 하나의 파일에 대한 블록이 여러 곳으로 분산되는 것을 막음.
- 차이점: 위는 여러 개의 저장을 가능하게 한 거고, 아래는 여러 개의 저장 장소를 찾아보는 동안 기다리는 기능.
- (참고) XFS: 저널링, 할당 지연 기능, 메모리에 캐시 시스템을 만든 성능이 비슷한 시스템

파티션이란?

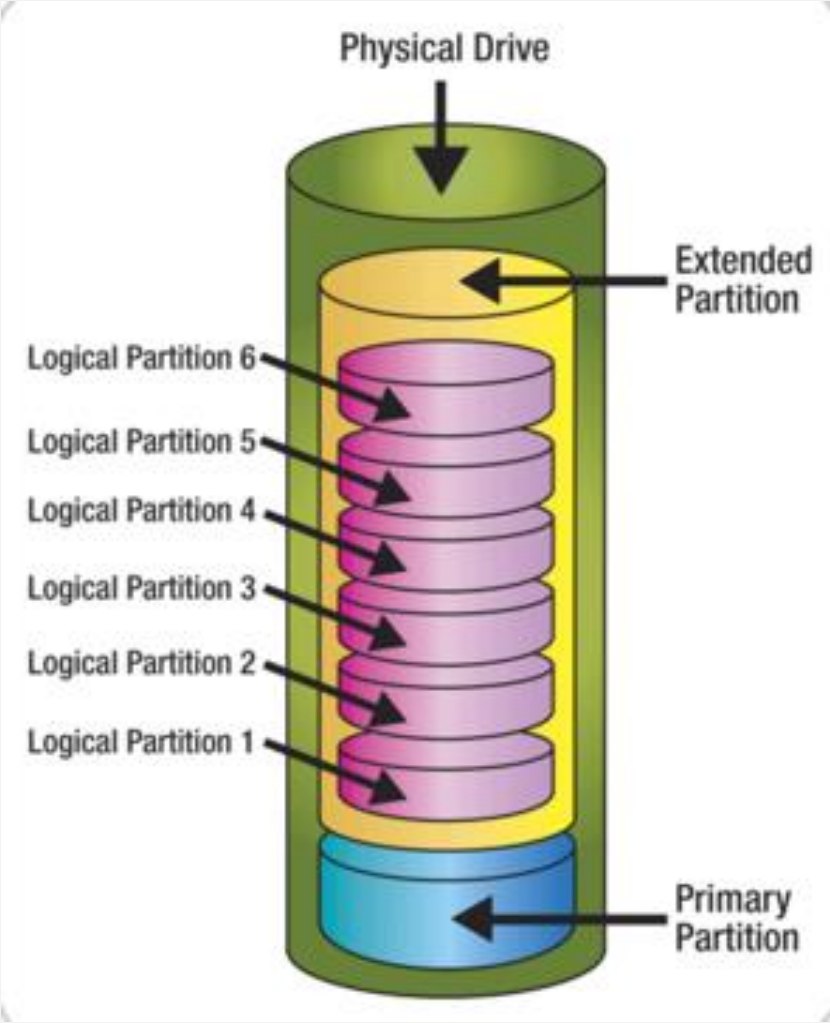
- 하나의 물리적인 디스크를 논리적으로 여러 개의 저장소인 것처럼 취급
- 한 파티션의 파일 시스템이 고장나더라도 다른 파티션에 영향을 주지 않는다.
- 각 파티션마다 따로 포맷 가능.



파티션이란? (MBR)

- 물리적 파티션 (Physical Partition) [Primary]
  - 실제 장치를 직접 나눈 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.



MBR & GPT

- MBR, GPT 는 파티션의 스타일!

- MBR (Master Boot Record)
  - 디스크의 가장 첫번째 섹터에 위치해 있는 512byte 공간
  - 부팅 코드, 디스크 서명, 파티션
  - 파티션 개수 4개 제한, 크기 2TB 제한

- GPT (GUID Partition Table)
  - 최대 8ZB, 최대 128개 파티션
  - 옛날 버전 호환 안됨.

Linux 파일 시스템 관리

1. 파티션 설정
   - fdisk
2. 파일 시스템 생성
   - mkfs
3. 장치 마운트/언마운트
   - mount/umount
4. 파일 시스템 점검/복구
   - fsck

```
hubo@ssal:~$ df -h
Filesystem                  Size  Used Avail Use% Mounted on
udev                        7.9G     0  7.9G   0% /dev
tmpfs                       1.6G  131M  1.5G   9% /run
/dev/mapper/ssal--vg-root   443G  217G  203G  52% /
tmpfs                       7.9G   23M  7.9G   1% /dev/shm
tmpfs                       5.0M     0  5.0M   0% /run/lock
tmpfs                       7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/sda1                   472M  169M  280M  38% /boot
tmpfs                       1.6G     0  1.6G   0% /run/user/4702
tmpfs                       1.6G     0  1.6G   0% /run/user/4918
hubo@ssal:~$
```

$ 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, ..

```
hubo@ssal:~$ lsblk
NAME                 MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                    8:0    0 465.8G  0 disk
├─sda1                 8:1    0   487M  0 part /boot
├─sda2                 8:2    0     1K  0 part
└─sda5                 8:5    0 465.3G  0 part
  ├─ssal--vg-root    252:0    0 449.3G  0 lvm  /
  └─ssal--vg-swap_1  252:1    0    16G  0 lvm  [SWAP]
hubo@ssal:~$
```

1. 파티션 설정

- $ fdisk [디스크장치] => 디스크에 대한 파티션 설정을 함.
  - 대화형 인터페이스로 구성
    - P 파티션 테이블 출력
    - n 파티션 추가
    - d 파티션 삭제
    - w 파티션 테이블 저장하고 종료
    - q 파티션 테이블 저장하지 않고 종료 *도움말 m

- $ fdisk –l [디스크장치] => 디스크의 파티션 테이블을 보여줌.
- $ fdisk –s [파티션장치] => 파티션의 크기를 블록 단위로 보여줌.

2. 파일 시스템 생성

- 파일 시스템은 고유의 mkfs 명령이 존재
  - 예로, EXT3 파일 시스템의 경우 mkfs.ext3
  - mkfs 는 그저 명령어를 각각 실행시켜줌

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

=> 두 명령어는 동치

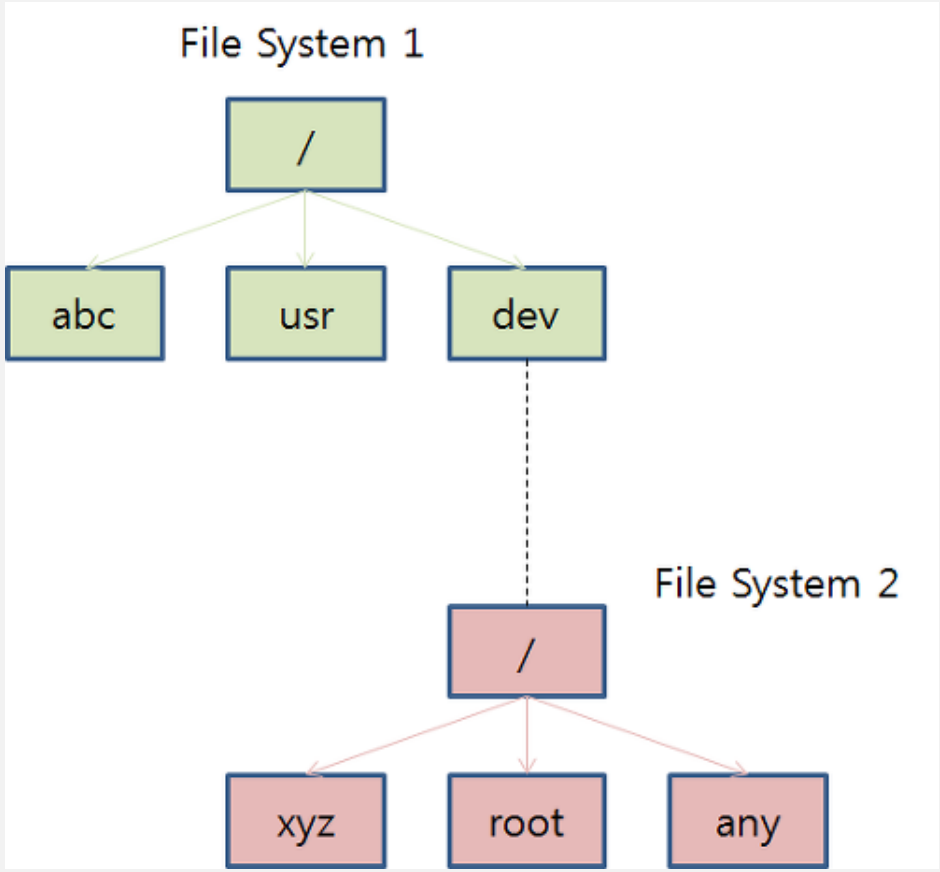
- $ mkfs –c [파티션 장치]
  - 배드섹터 검사를 진행

3. 장치 마운트/언마운트

- 마운트 (mount) : 하나의 파일 시스템을 다른 파일 시스템의 어떤 location 에 붙이는 것.

  ex) USB 인식

- 저 /dev/가 마운트 포인트



3. 장치 마운트/언마운트

- $ 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 or 장치명]

- $ mount –a
  - /etc/fstab 에서 auto 로 표시된 파일 시스템들을 마운트

3. 장치 마운트/언마운트

```
hubo@ssal:/dev$ 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>
/dev/mapper/ssal--vg-root /               ext4    errors=remount-ro 0       1
# /boot was on /dev/sda1 during installation
UUID=bf0ef9ae-ec5e-4ed5-874f-b1fc93c43c10 /boot           ext2    defaults        0       2
/dev/mapper/ssal--vg-swap_1 none            swap    sw              0       0
143.248.234.104:/home /home nfs rw,hard,intr 0 0

hubo@ssal:/dev$
```

4. 파일 시스템 점검/복구

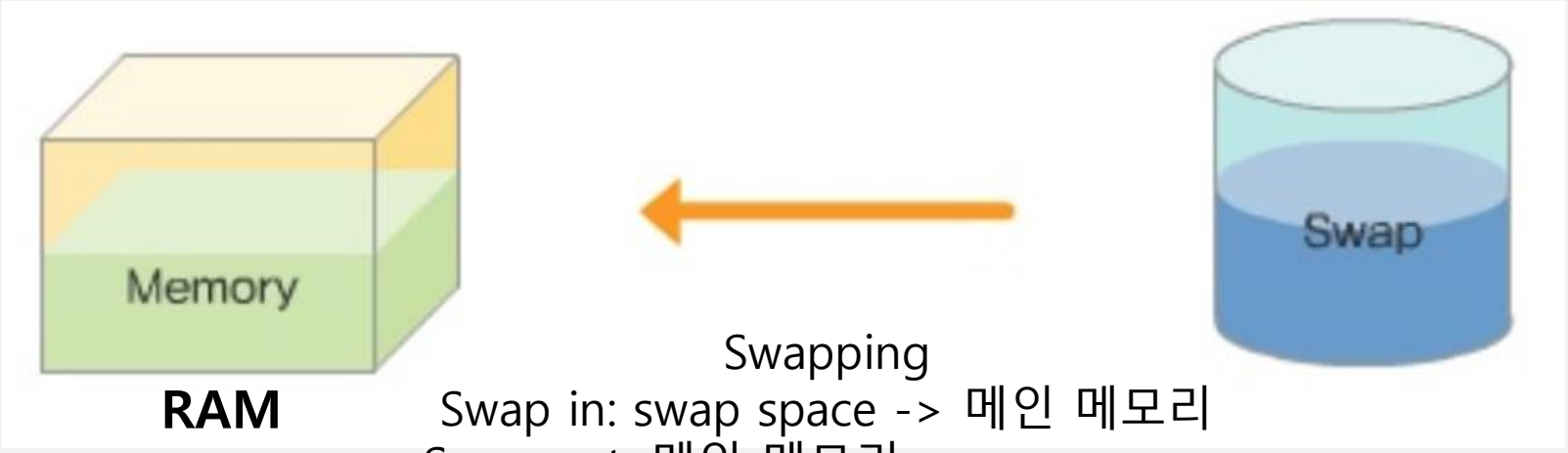
- $ fsck –t [파일시스템 종류] [파티션 장치]
  - 파일 시스템이 정상인지 확인하고 오류가 있다면 복구한다.

- **반드시!! unmount 한 후에 사용**

- mkfs 처럼 파일 시스템마다 fsck 존재
- 재부팅을 해야 적용된다.

스왑 영역이란?

- 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.



스왑 공간의 할당 방법

- Swap Partition
  - Swapping 만을 위해 사용되는 파티션
  - 다른 파일은 존재할 수 없음

```
hubo@hubo-VirtualBox:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    10G  0 disk
├─sda1   8:1    0     9G  0 part /
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0  1022M  0 part [SWAP]
sr0     11:0    1  82.6M  0 rom  /media/hubo/VBox_GAs_6.0.8
```

- Swap File
  - Swapping 을 위한 파일
  - 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

스왑 공간의 관리

1. 메모리 상태 확인
   - free
2. 스왑 파일/파티션 만들기
   - dd/fdisk
3. 스왑 공간 만들기
   - mkswap
4. 스왑 공간의 활성화/비활성화
   - swapon/swapoff

1. 메모리 상태 확인

- free: 시스템의 메모리 사용 현황을 출력 (KB)

```
hubo@hubo-VirtualBox:~$ free
             total       used       free     shared    buffers     cached
Mem:       1024148     754088     270060       5396      46656     406952
-/+ buffers/cache:     300480     723668
Swap:      1046524          0    1046524
hubo@hubo-VirtualBox:~$
```

- total / used / free 는 이름 그대로.
- 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. 스왑 파일/파티션 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.

2. 스왑 파일/파티션 만들기

```
hubo@hubo-VirtualBox:~$ ls
bochs-2.6.2  Documents  examples.desktop  Pictures  Public     Videos
Desktop      Downloads  Music             pintos    Templates
hubo@hubo-VirtualBox:~$ dd if=/dev/zero of=swapfile bs=1024 count=100
100+0 records in
100+0 records out
102400 bytes (102 kB) copied, 0.0042767 s, 23.9 MB/s
hubo@hubo-VirtualBox:~$ ls
bochs-2.6.2  Documents  examples.desktop  Pictures  Public    Templates
Desktop      Downloads  Music             pintos    swapfile  Videos
hubo@hubo-VirtualBox:~$ sync
```

```
-rw-------  1 hubo hubo 102400  6월 30 19:01 swapfile
-rw-rw-r--  1 hubo hubo 102400  6월 30 19:10 swapfile-withoutchmod
```

3. 스왑 공간 만들기

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

```
hubo@hubo-VirtualBox:~$ mkswap swapfile
Setting up swapspace version 1, size = 96 KiB
no label, UUID=667325a9-3eb4-4e00-a0af-4ed9f5749d22
hubo@hubo-VirtualBox:~$ mkswap -c swapfile
Setting up swapspace version 1, size = 96 KiB
no label, UUID=8e5d582f-2f4c-4700-b8a0-786306a5d8d3
hubo@hubo-VirtualBox:~$ mkswap -c swapfile 80
Setting up swapspace version 1, size = 76 KiB
no label, UUID=9abcffa1-01bb-4edf-acb6-4af16384489e
hubo@hubo-VirtualBox:~$
```

4. 스왑 공간의 활성화/비활성화

- $ swapon [파티션장치/스왑파일명]
- $ swapoff [파티션장치/스왑파일명]

- $ swapon –a
  - => /etc/fstab 파일 시스템 테이블에서 sw 옵션 값을 갖는 항목 모두 활성화 (부팅 시 자동으로 실행)
- $ swapon –s
  - 스왑파티션의 상태를 보여준다

4. 스왑 공간의 활성화/비활성화

전

```
hubo@hubo-VirtualBox:~$ free
             total       used       free     shared    buffers     cached
Mem:       1024148     754088     270060       5396      46656     406952
-/+ buffers/cache:     300480     723668
Swap:      1046524          0    1046524
hubo@hubo-VirtualBox:~$
```

후

```
hubo@hubo-VirtualBox:~$ free
             total       used       free     shared    buffers     cached
Mem:       1024148     804792     219356       5444      50884     439452
-/+ buffers/cache:     314456     709692
Swap:      1046600          0    1046600
hubo@hubo-VirtualBox:~$
```

감사합니다 :)

Q&A

참고

wiki.sparcs.org

nick-20160709-0.pptx
moonde-20170703-0.pptx
coearth-20150712_1-0.pptx
andromeda-20140729-0.pdf