gangok@sparcs

5. 파일시스템, 스왑 영역, 장치 파일 관리

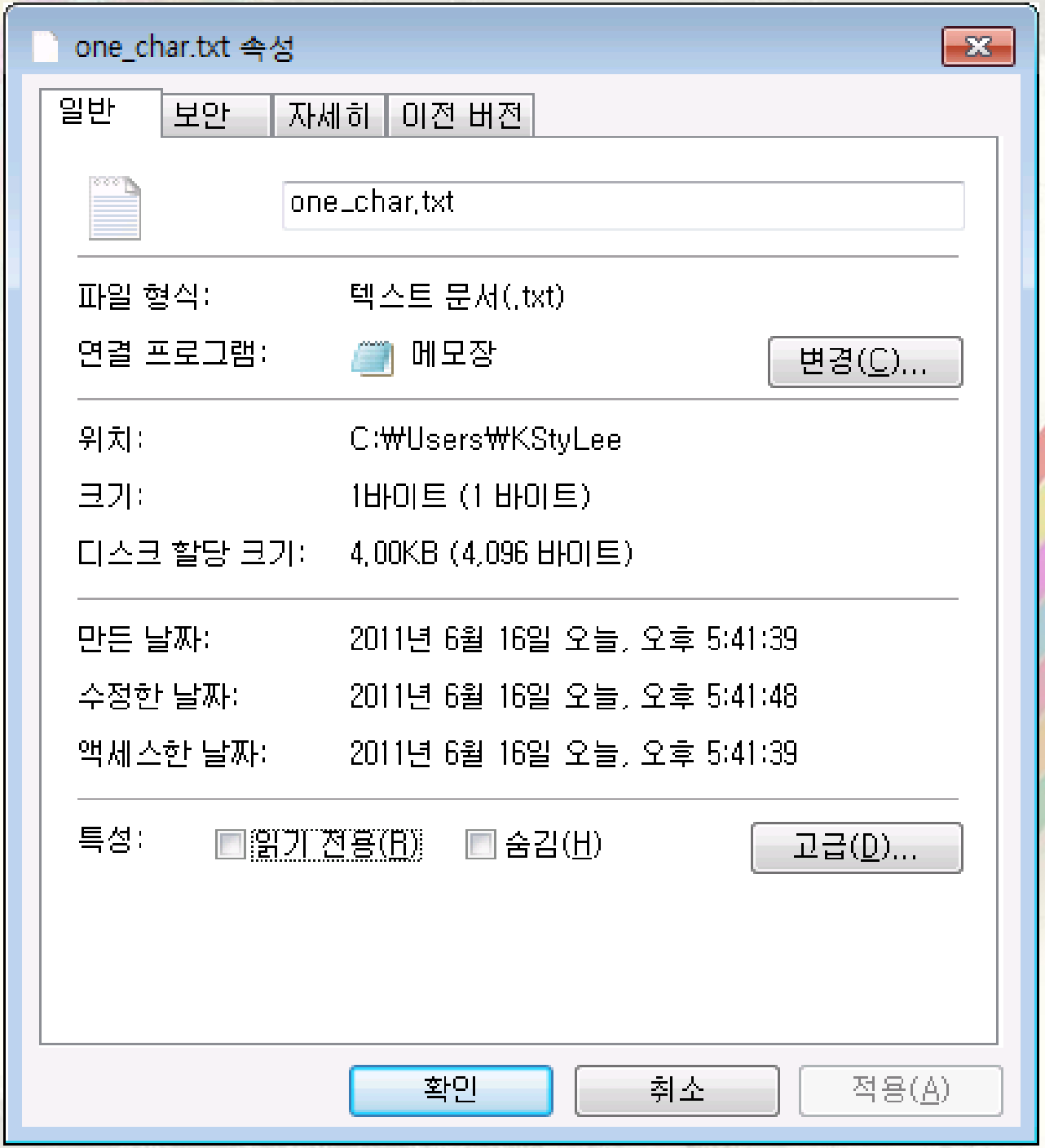
파일시스템(file system) 관리

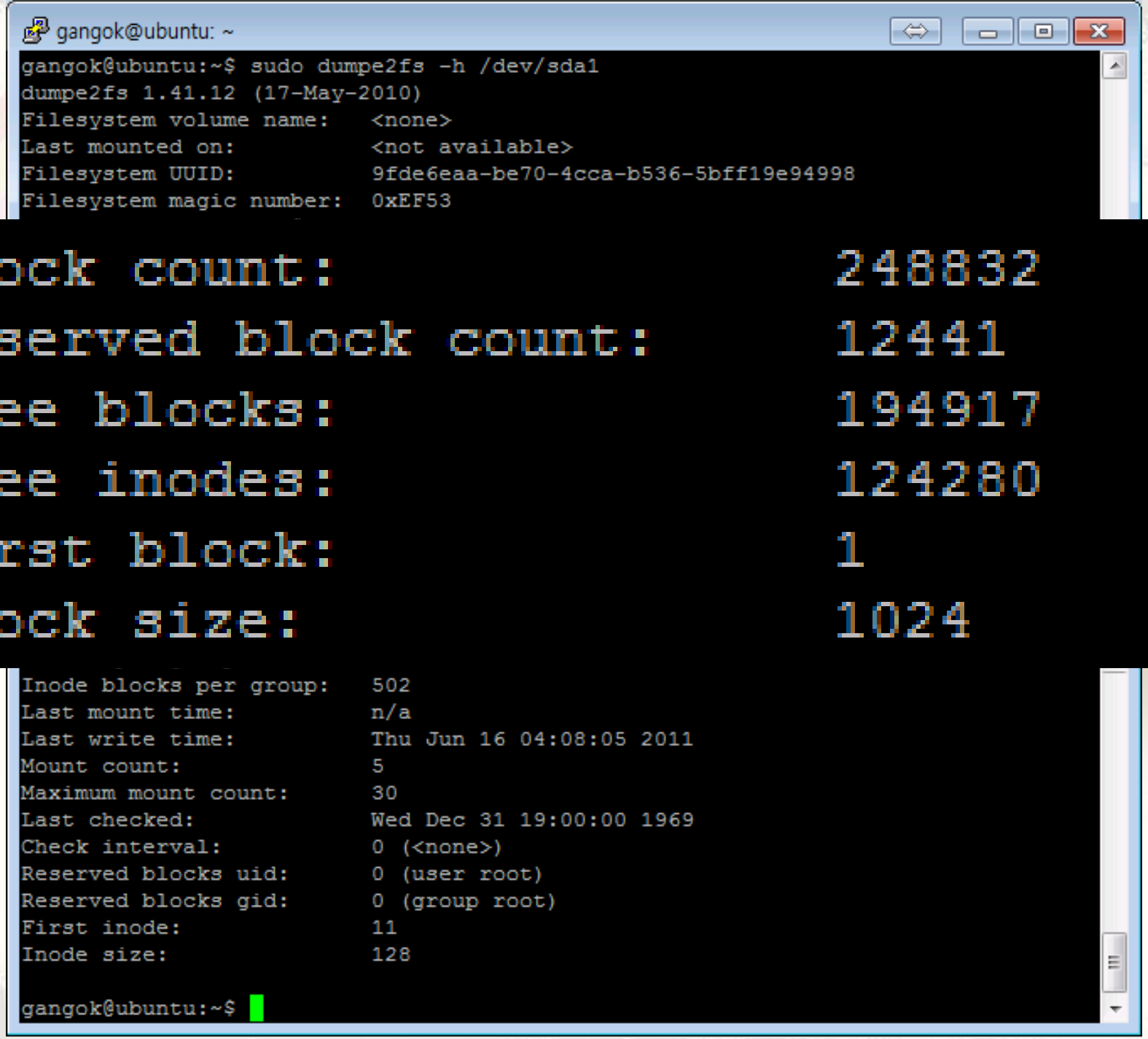
file system?

- 컴퓨터에서 파일이나 자료를 쉽게 발견 및 접근할 수 있도록 보관 또는 조직하는 체제
- 운영체제별로 지원하는 파일 시스템의 종류가 다름
- Magnetic Tape과 같이 sequential access만 허용하는 저장 매체에도 파일시스템이 필요하나 일반적으로 우리가 생각하는 파일시스템과는 형태가 다름

블록(block) vs 섹터(sector)

- sector : 하드디스크에서 데이터를 저장하는 최소 단위
  - 일반적으로 512byte, 최근 4096byte로 표준이 확장
- block : 파일시스템에서 파일을 저장하는 최소 단위
  - Sector의 정수 배의 크기를 가짐
  - file system을 설정할 때 block 크기를 결정할 수 있음


one_char.txt 속성
일반
보안
자세히
이전 버전
one_char.txt
파일 형식: 텍스트 문서(.txt)
연결 프로그램: 메모장
변경(C)...
위치: C:₩Users₩KStyLee
크기: 1바이트 (1 바이트)
디스크 할당 크기: 4.00KB (4,096 바이트)
만든 날짜: 2011년 6월 16일 오늘, 오후 5:41:39
수정한 날짜: 2011년 6월 16일 오늘, 오후 5:41:48
액세스한 날짜: 2011년 6월 16일 오늘, 오후 5:41:39
특성: 읽기 전용(R) 숨김(H)
고급(D)...
확인
취소
적용(A)


gangok@ubuntu: ~
gangok@ubuntu:~$ sudo dumpe2fs -h /dev/sda1
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 9fde6eaa-be70-4cca-b536-5bff19e94998
Filesystem magic number: 0xEF53
Block count: 248832
Reserved block count: 12441
Free blocks: 194917
Free inodes: 124280
First block: 1
Block size: 1024
Inode blocks per group: 502
Last mount time: n/a
Last write time: Thu Jun 16 04:08:05 2011
Mount count: 5
Maximum mount count: 30
Last checked: Wed Dec 31 19:00:00 1969
Check interval: 0 (<none>)
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: 11
Inode size: 128
gangok@ubuntu:~$

ext2 파일 시스템의 동작 원리

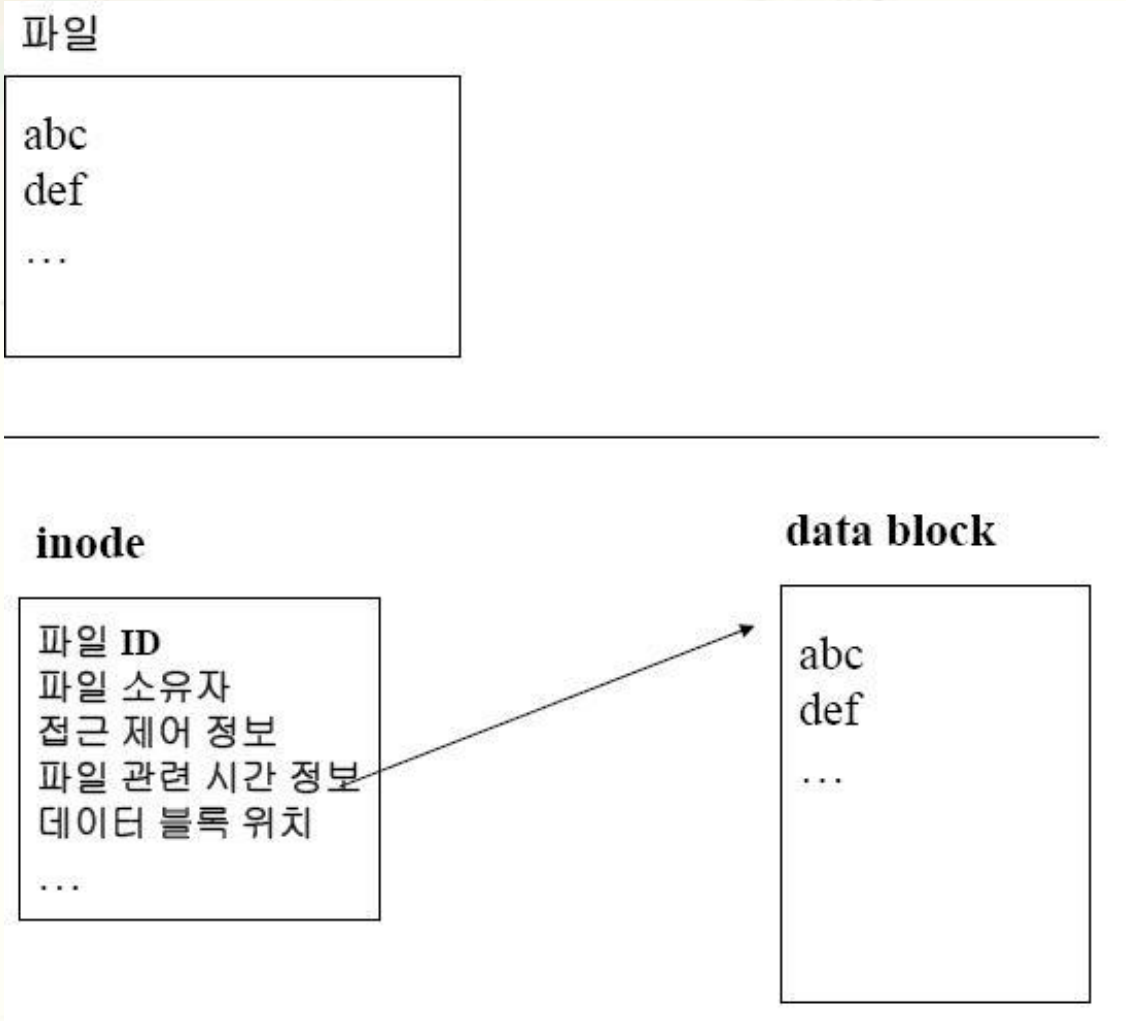
- 파일 시스템에 inode block과 data block이 있는데, inode block에는 파일들이나 디렉토리들의 메타 데이터가, data block에는 내용이 들어있다

- 파일은 inode block에 해당 파일의 data block 위치, 접근 권한 정보, 소유자 정보 등을 저장하고 data block에는 실제 파일 내용을 저장한다

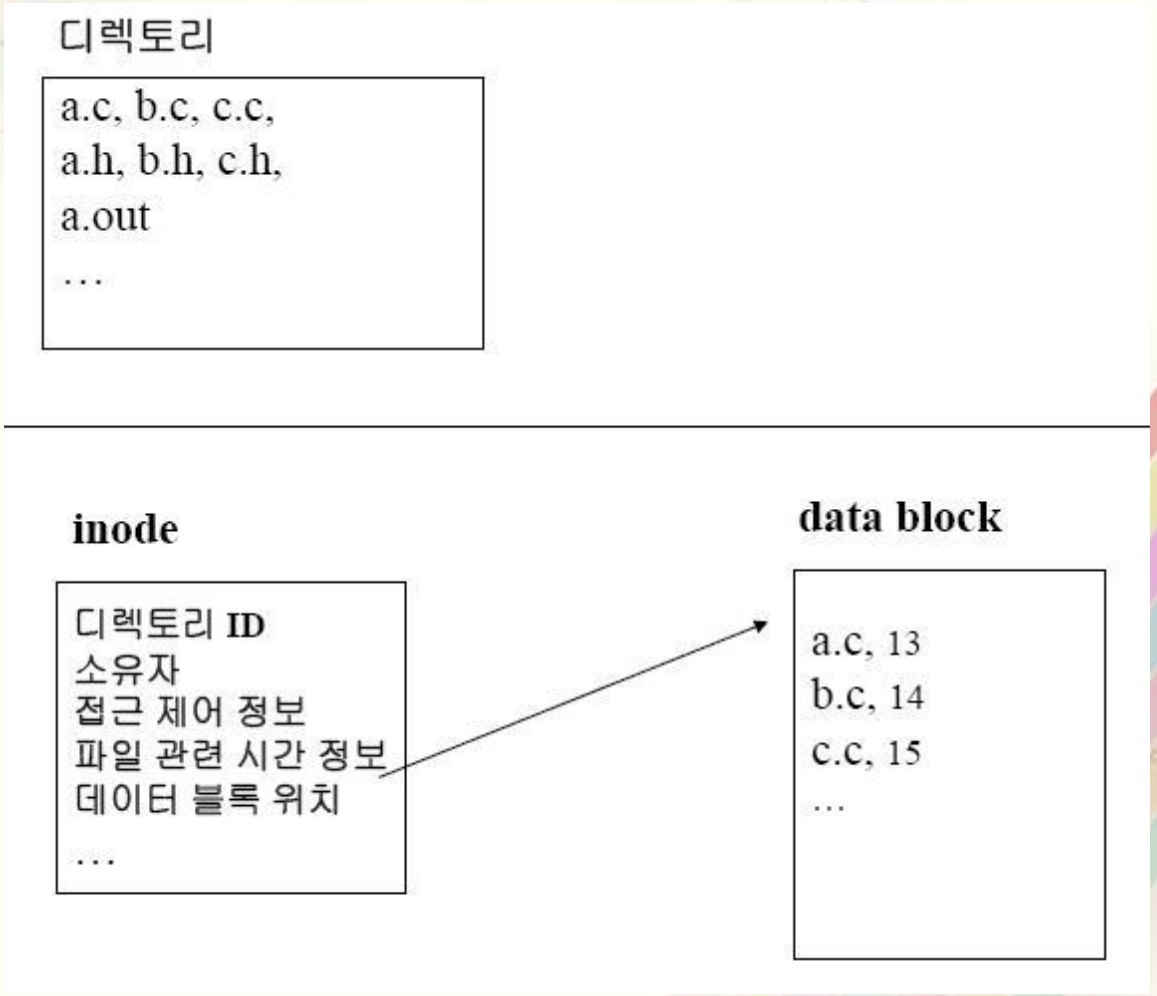
ext2 파일 시스템의 동작 원리

- 디렉토리 역시 inode block에는 해당 디렉토리의 data block 위치, 접근 권한 정보, 소유자 정보 등을 저장하며 data block에는 해당 디렉토리에 속하는 파일의 이름과 inode block의 위치 목록을 저장한다

ext2 파일 시스템의 동작 원리



ext2 파일 시스템의 동작 원리



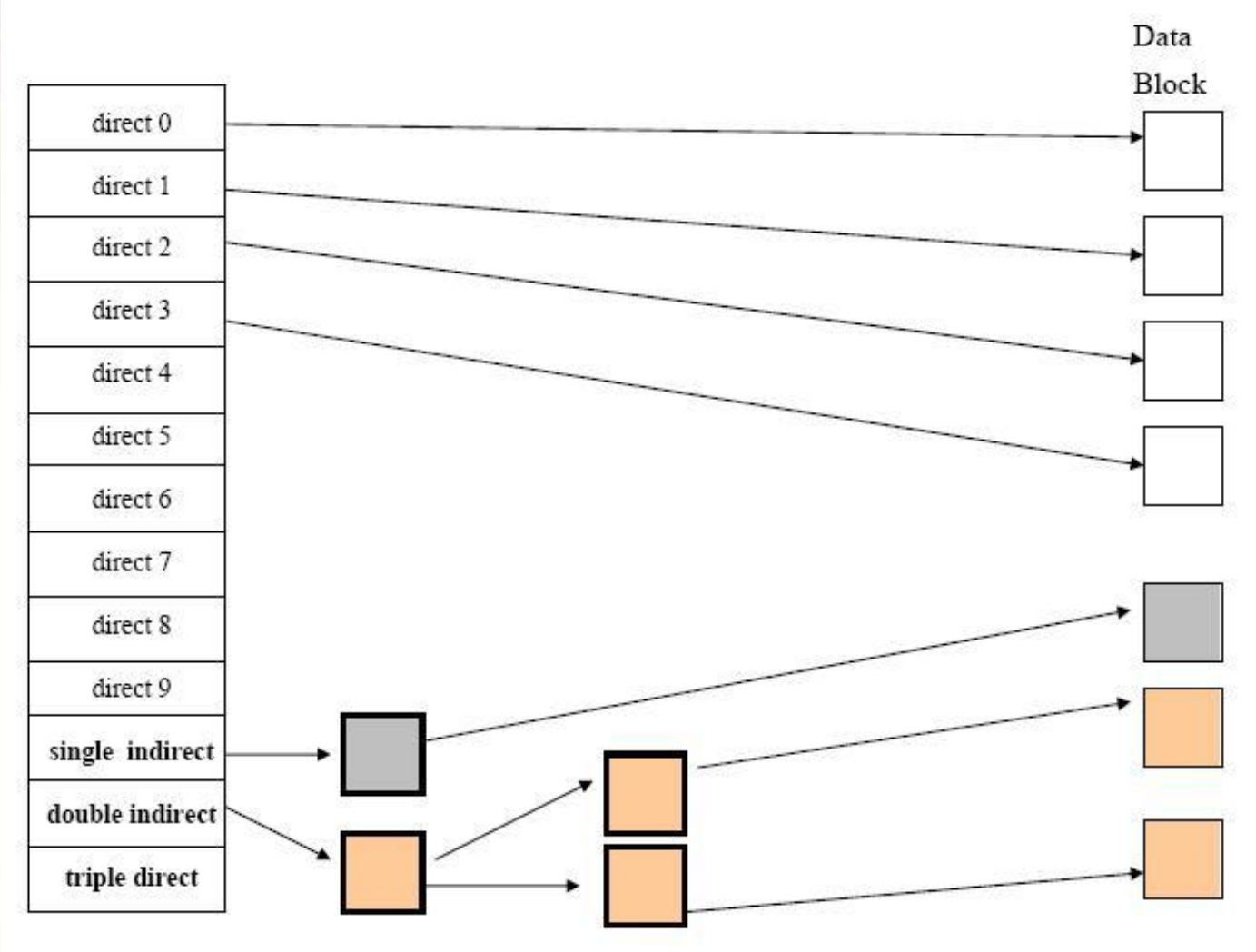
ext2 파일 시스템의 동작 원리

- 일반적으로 inode block에서 data block을 가리키는 포인터의 숫자는 고정되어 있는데, 그렇다면 data block이 많이 필요한, 크기가 매우 큰 파일에 대해서는 어떻게 될까??

ext2 파일 시스템의 동작 원리

- 그 경우 inode block의 포인터들은 빈 data block을 가리키고 그 data block 안에 포인터들을 배치시켜서 파일의 data block들을 가리키게 할 수 있는데, 이를 single indirect라 한다
- data block의 포인터들이 다른 data block들을 가리키고, 거기서 가리켜진 data block들의 포인터들이 파일의 실제 data block들을 가리키면 이는 double indirect이다

ext2 파일 시스템의 동작 원리



윈도우에서 쓰이는 file system의 종류(1/2)

- FAT16/FAT32
  - FAT는 File Allocation Table, 즉 파일의 위치 정보 등을 기록하기 위한 테이블을 말하는데 이것이 파일 시스템 자체를 가리키게 되었다
  - FAT16은 단일 파티션의 최대 용량이 2GB로 저용량 장치에만 쓰일 수 있어 요즘에는 이동식 드라이브 외에는 거의 사용되지 않는다
  - FAT32는 단일 파티션의 최대 용량이 8TB에 달하지만, 4GB 이상의 파일을 가질 수 없고 보안상의 결점이 많아 대신 NTFS를 쓰는 것이 일반화되었다

윈도우에서 쓰이는 file system의 종류(2/2)

  - 극히 단순한 자료구조가 장점이자 단점으로, 대부분의 OS에서 지원하나 성능은 좋지 않다.
- NTFS
  - 기존의 FAT 시리즈에 비해 메타 데이터를 지원하고 성능을 개선하는 등 장점이 많다
  - 윈도우즈 NT 계열에서부터 사용되었다

리눅스에서 주로 쓰이는 file system의 종류(1/5)

- ext2
- ext3
- ext4
- XFS
- ReiserFS
- NFS

- 요즘 가장 많이 쓰이는 것은 ext3와 ext4이다

리눅스에서 주로 쓰이는 file system의 종류 (2/5)

- ext2
  - 과거에 리눅스 시스템에 널리 사용되었으나 ext3로 대체되어 가고 있음
- ext3
  - ext2와 비슷하나 저널링을 지원

저널링(journalling)

- 파일시스템에 가해진 변경 사항을 저널(journal)에 기록해 두는 기술
- 망가진 파일 시스템을 복구할 때 도움이 됨

ext3에서 지원하는 저널링 (1/3)

- Journal(리스크 최소)
  - 메타 데이터와 파일 내용 모두 저널에 쓴 뒤에 메인 파일 시스템에 업데이트 한다.
  - 데이터가 2번 기록되기 때문에 성능이 저하될 수 있다.

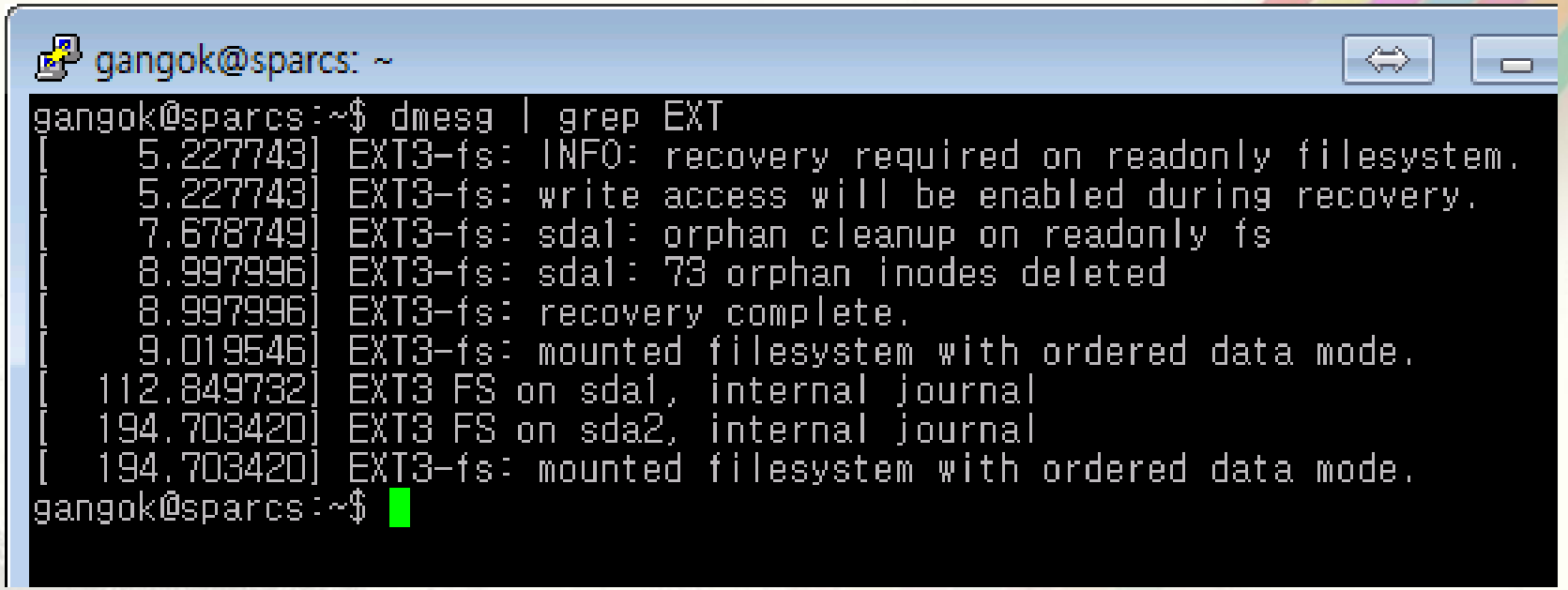
ext3에서 지원하는 저널링 (2/3)

- Ordered(리스크 중간)
  - 메타 데이터만 저널에 기록된다.
  - 저널의 메타 데이터에 commited라고 표시된 경우 파일 내용도 제대로 쓰여 졌다고 확신할 수 있다.
  - 저널의 메타 데이터에 not commited라고 표시된 경우, 이 메타 데이터가 가리키는 파일이 새로운 파일이거나 추가된 파일의 경우, 자동으로 삭제된다.
  - 덮어씌우는 파일을 가리키는 경우, 이는 과거 버전, 현재 버전 어느것으로도 복구할 수 없다

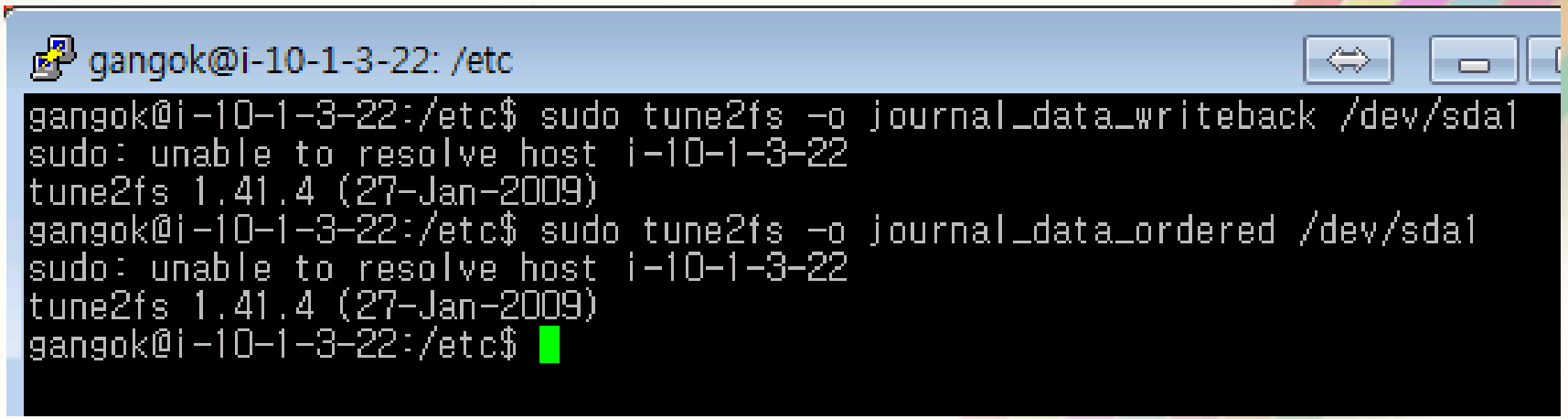
ext3에서 지원하는 저널링 (3/3)

- Writeback(리스크 높음)
  - 메타 데이터만 저널에 기록된다.
  - 실제 데이터가 기록되기 전이나 후에나 저널이 업데이트 될 수 있다.
  - 문제가 생기기 직전 기록하고 있던 파일들은 제대로 기록된 것인지 아닌지 확인하기 힘들다.

- 저널링 방식 확인
  - # sudo dmesg | grep EXT
  - 기본값: EXT3-fs (sdX): mounted filesystem with ordered data mode.



- 저널링 모드를 변경
  - # tune2fs -o journal_data_writeback /dev/sdX



리눅스에서 주로 쓰이는 file system의 종류 (3/5)

- ext4
  - ext3의 64비트 버전
  - 하위 디렉토리 제한(ext3에서 32000개) 없음
  - 할당 지연 기능 지원
    - 데이터를 기록할 때 해당 데이터를 위해 빈 공간의 크기는 그만큼 줄어든 것으로 생각하지만, 어떠한 블록에 기록할지는 결정하지 않는다
    - 조금씩 크기가 커지는 파일이 여러 조각으로 나뉘어 저장되는 것을 방지한다.
    - 금방 사라지는 임시 파일들에 대해서는 디스크 블록이 할당되지 않기 때문에 할당 해제 작업도 필요하지 않다

리눅스에서 주로 쓰이는 file system의 종류 (4/5)

- XFS
  - 저널링을 사용하는 64비트 파일 시스템
  - 디스크를 쓰기보다는 메모리상에 많은 데이터를 캐시하여 전반적으로 좋은 퍼포먼스를 보이며, 특히 큰 파일을 주로 다루는 시스템에서 좋다
  - 할당 지연 기능 지원

리눅스에서 주로 쓰이는 file system의 종류 (5/5)

- ReiserFS
  - 저널링을 지원한 최초의 파일 시스템
  - 초기에는 metadata에 대한 저널링만 지원하였음
- NFS
  - Network File System으로 뒤쪽의 휠 세미나에서 다뤄질 예정이다

기타 file system

- ISO 9660 / UDF
  - CDROM / DVD
- HFS/HFS_Plus
  - Mac OS

file system의 선택에 따라 달라지는 것들

- block size
- maximum 파일명 길이
- maximum 파일 크기
- maximum 파일 시스템 크기
- 저장되는 metadata의 종류
- 안정성
- 등등

파티션(partition)?

- 하드 디스크를 나누어 놓은 구획
- 하나의 물리적인 하드디스크를 논리적으로 여러 개의 저장소인 것 처럼 취급

- primary partition
- extended partition
- logical partition

MBR(Master Boot Record)

- 물리적 하드디스크의 첫 512byte
- 이 중 16byte는 primary partition table를 표현하는데 사용
- 이 table은 최대 4개의 entry를 가질 수 있음

primary partition

- 실제 디스크를 직접적으로 나눈 파티션
- MBR의 구조 때문에 하나의 디스크에 최대 4개의 primary partition을 가질 수 있음
- 4개 이상의 partition을 만드려면 logical partition을 이용해야 함

extended / logical partition

- 하나의 디스크에는 최대 하나의 extended partition을 가질 수 있음
- 하나의 extended partition은 여러 개의 logical partition을 담는 그릇
- 한 디스크에 만들 수 있는 partition의 최대 개수는 제한되어 있음

logical vs primary partition

- logical partition과 primary partition의 속도 차이는 없음
- primary partition에만 설치 가능한 OS도 있음
- 보통 많은 수의 partition이 필요한 경우
  - 3개의 primary partition + 1개의 extended partition(which includes many logical partitions) 와 같이 구성함

file system on partition

- 하나의 파일 시스템은 하나의 파티션 위에 얹어져, 해당 파티션 내의 파일과 디렉토리를 관리하는 역할을 하게 된다

- 윈도에선 '포맷' 을 통해 나뉘어진 파티션 위에 파일 시스템을 설정

여러 파일 시스템을 이용할 때의 장점(1/2)

- 안정적
  - 한 파티션의 파일 시스템에 문제가 생겨도 다른 파티션의 파일 시스템은 보통 문제가 없음
- OS와 프로그램 설치 파일, 유저 파일 등을 분리하여 저장가능
  - /home 용 파티션을 따로 만들어 둔 경우, 이 파티션을 제외한 다른 파티션을 모두 지운 후 완전히 새롭게 시스템을 설치할 수 있다.

여러 파일 시스템을 이용할 때의 장점(2/2)

- 저장 매체가 여러 개인 경우
  - 1GB 크기의 하드 디스크가 2개인 경우 하나는 루트 파일 시스템, 하나는 /usr 파일 시스템을 만드는 식으로 사용 가능
- 한 디스크에 여러 OS 설치 가능

여러 파일 시스템을 이용할 때의 단점

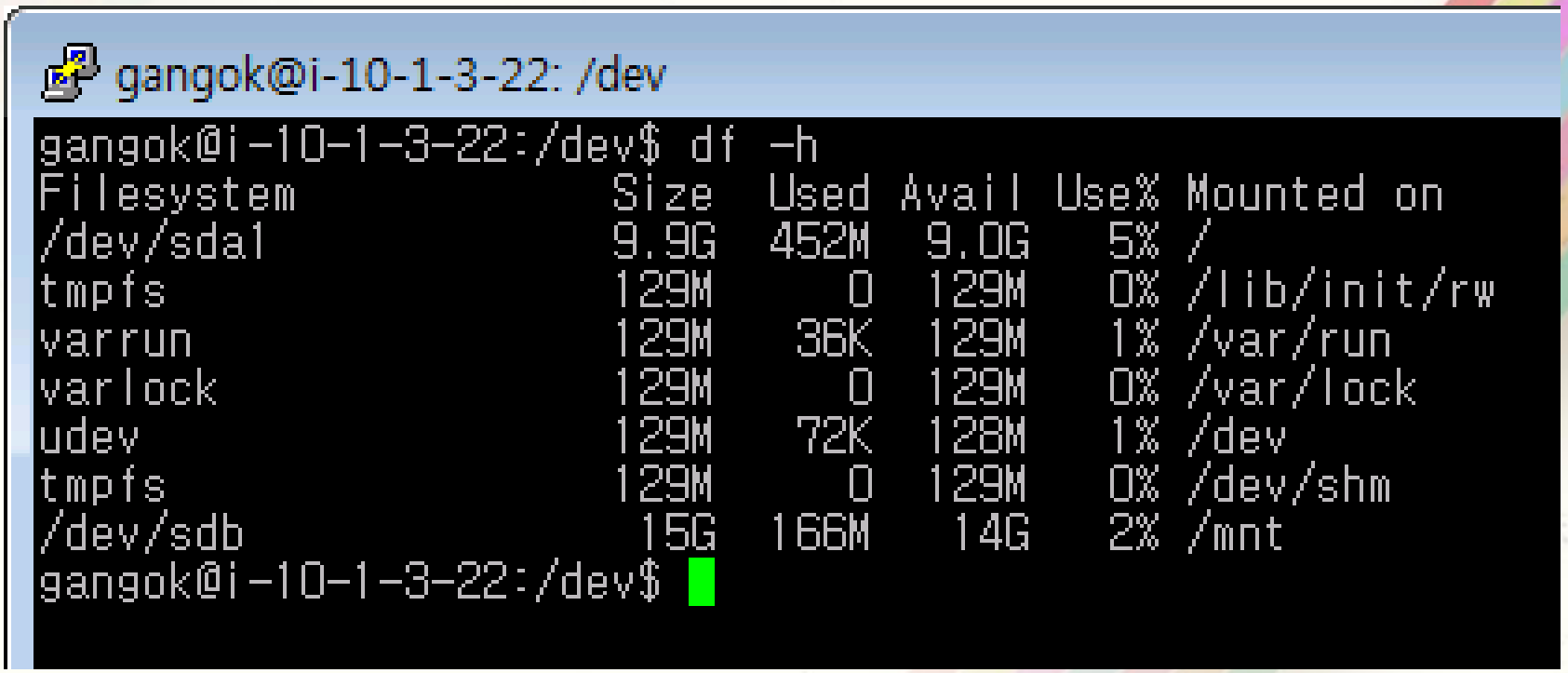
- 디스크 자원 전체를 사용하지 못하는 경우가 생김
  - ex) C:에 1GB, D:에 1GB 빈 공간이 남았는데 1.5GB 크기의 영화 파일을 저장하고 싶은 경우
- 같은 디스크라도 다른 파티션간에는 파일 이동이 파일 복사와 같은 작업을 하게됨
  - 같은 파티션 내에서는 파일 내용은 그대로 두고 위치 정보만 바꾸면 됨

partition 및 file system 관리

- 파일 시스템 상태 보기
  - df
- 파티션 설정
  - fdisk
- 파일시스템 설정
  - mkfs
- 장치 마운트 / 언마운트
  - mount/umount
- 파일 시스템 점검/복구
  - fsck

파일 시스템 상태 보기

- $ df [-h]



파티션 설정 (1/7)

- 리눅스 시스템의 장치 파일들은 /dev에서 볼 수 있음
- IDE 형식의 하드디스크
  - /dev/hda, /dev/hdb, /dev/hdc, …와 같이 지정
- S-ATA 형식의 하드디스크
  - /dev/sda, /dev/sdb, /dev/sdc, …와 같이 지정

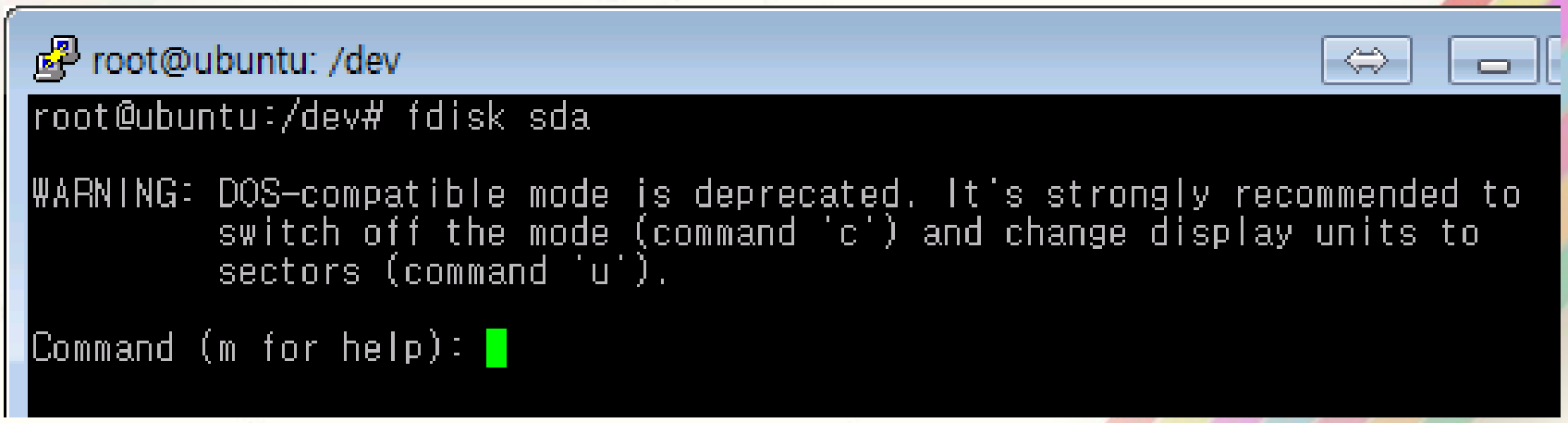
파티션 설정 (2/7)

- 이러한 물리적인 디스크에 파티션이 생성되면 그 장치 파일 뒤에 1번부터 번호가 붙은 이름으로 파티션에 해당하는, 가상의 장치의 파일이 생성된다

- 예를 들어 /dev/hda에 대해 파티션 두개를 만들었다면 /dev/hda1, /dev/hda2로 지정된다

파티션 설정 (3/7)

- # fdisk [디스크 장치]



파티션 설정 (4/7)

- 주요 명령어 : p 현재 파티션 상태 보기

```
Command (m for help): p

Disk sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001ad3b

Device Boot      Start         End      Blocks   Id  System
  sda1   *           1          32      248832   83  Linux
Partition 1 does not end on cylinder boundary.
  sda2              32        2611    20719617    5  Extended
  sda5              32        2611    20719616   8e  Linux LVM

Command (m for help):
```

파티션 설정 (5/7)

- 주요 명령어 : n 파티션 추가하기

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
p
Partition number (1-4): 3
No free sectors available

Command (m for help):
```

- ……

파티션 설정 (6/7)

- Partition number (1-4) : 2
- First cylinder (204 – 683) : 204
- Last cylinder or +size or +sizeM or +sizeK (204-683) : +80M

파티션 설정 (7/7)

- 그 밖의 주요 명령어들
- d : 파티션 삭제
- w : 이제까지 변경 사항을 저장하고 종료
- q : 이제까지 변경 사항을 무시하고 그냥 종료

- **m : help**

파일시스템 설정 (1/3)

- #mkfs [파티션 장치]
  - -c : 배드섹터 검사
  - -t 파일 시스템 타입 : 해당 파일 시스템 타입으로 포맷
  - ex) mkfs –c –t ext3 /dev/sda2

파일시스템 설정(2/3)

- 각 파일시스템은 자신과 관련된 고유의 mkfs명령을 가지고 있다.
  - ex) mkfs.msdos, mkfs.ext3
  - mkfs는 이러한 것들의 frontend일 뿐

- mkfs.ext3 –c /dev/sda2

```
asanabri@asanabri-desktop:~$ sudo mkfs -t ext3 /dev/sdb
sdb   sdb1
asanabri@asanabri-desktop:~$ sudo mkfs -t ext3 /dev/sdb1
mke2fs 1.40.8 (13-Mar-2008)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
62976 inodes, 250976 blocks
12548 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
7872 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
asanabri@asanabri-desktop:~$
```

장치 마운트 / 언마운트 (1/4)

- 리눅스에서 어떤 파일시스템에 접근하려면, 먼저 어떤 디렉토리에 해당 파일 시스템을 연결시켜야 한다.

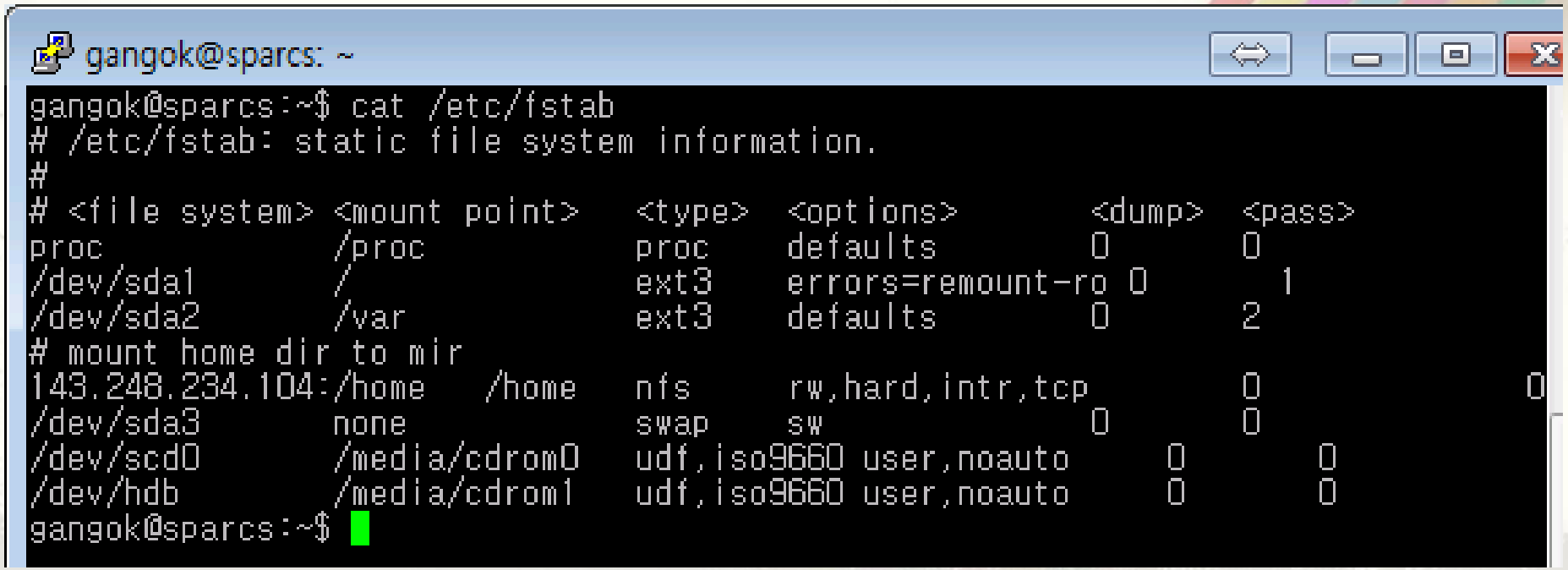
- 이러한 연결 작업을 마운트라 하며 파일시스템의 파일들이 마치 그 디렉토리에 있는 것처럼 보인다.

장치 마운트 / 언마운트 (2/4)

- # mount –t type device mount-point
  - ex) mount –t ext3 /dev/sda2 /mnt
  - ext3로 포맷된 /dev/sda2를 /mnt라는 디렉토리에 연결시키는 명령
  - 해당 디렉토리는 먼저 만들어두어야 한다.

장치 마운트 / 언마운트 (3/4)

- # mount –a
  - 부팅시 자동으로 실행
  - /etc/fstab 파일에 “auto”옵션을 가진 모든 파일시스템들을 마운트 시킨다



장치 마운트 / 언마운트 (4/4)

- # umount device or
- # umount mount-point

- 해당 파일 시스템을 언마운트 시킴
- CD-ROM, 플로피디스크의 경우 언마운트를 시켜줘야 안전하게 제거가 가능함

파일시스템 점검/복구 (1/2)

- #fsck –t type device
- 리눅스 파일시스템의 자료에 문제가 없는지 점검하고, 만약 자료 에러나 손실이 있을 경우 복구해주는 프로그램
- mkfs와 마찬가지로 프론트엔드일 뿐

파일시스템 점검/복구 (2/2)

- **일반적으로 fsck를 이용하여 파일시스템을 점검하려면 해당 파일시스템을 언마운트 시킨 상태에서 수행하는 것이 좋다.**

- 루트 파일시스템의 경우 언마운트가 불가능하므로 부팅 디스크를 이용하거나 GRUB부트로더에서 읽기전용, 단일 사용자 모드로 부팅하여 수행할 수 있다.

파일시스템 관리 실습

- dd 명령을 이용하여 적당한 크기(100M)의 빈 파일을 만든다.
- mkfs 명령을 이용하여 ext 파일시스템으로 해당 파일을 포맷한다.
- Mount 명령을 이용하여 해당 파일을 마운트 시킨 뒤 파일 작성 등 테스트를 진행한다.
- 언마운트 된 상태에서 해당 파일을 삭제한다.

스왑 영역(swap space) 관리

swap space?

- 시스템에서 사용 가능한 메모리량을 늘리기 위해 디스크 저장 장치에서 사용되는 공간
- 디스크의 접근 속도는 RAM에 비해 훨씬 늦기 때문에 자주 사용되지 않는 데이터를 저장하는데 사용됨
- Not mandatory, but highly recommended in linux

swap space 할당 방법

- 스왑 공간을 할당하는 방법에는 두 가지가 있다.
  - swap file
  - swap partition

- 단일 파티션과 단일 파일 모두 최대 2GB의 크기를 가질 수 있고, 이를 최대 8개 만드는 것이 가능하며 최대 총 16GB의 크기를 가질 수 있다.(인텔 시스템 기준)

swap file vs swap partition

- 스왑 파티션이 더 빠르다
  - 디스크 블록이 연속적이기 때문
  - 반대로 스왑 파일을 사용하면 디스크 블록이 파일시스템 여러 곳에 흩어져 있을 수 있음
- 그래서 일반적으로 스왑 파티션을 사용하며 임시로 스왑 영역을 추가할 때 스왑 파일을 사용

swap space 관리

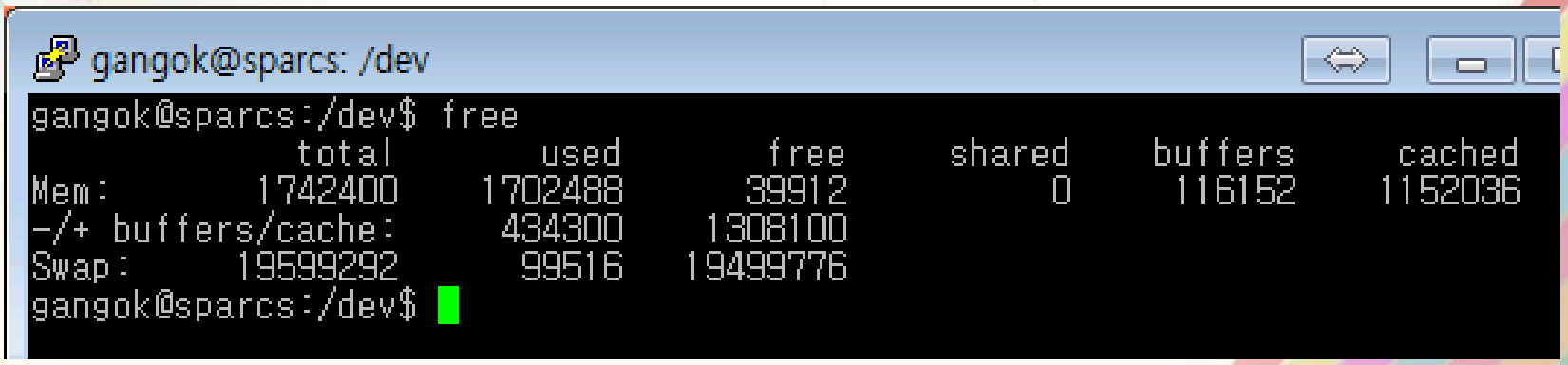
- 현재 메모리 상태 확인
  - free
- 스왑 파티션 만들기
  - fdisk, mkswap
- 스왑 파일 만들기
  - dd, mkswap
- 스왑 영역 활성화, 비활성화
  - swapon/swapoff

현재 메모리 상태 확인

- free 명령을 이용하여 메모리의 상태를 확인할 수 있다.

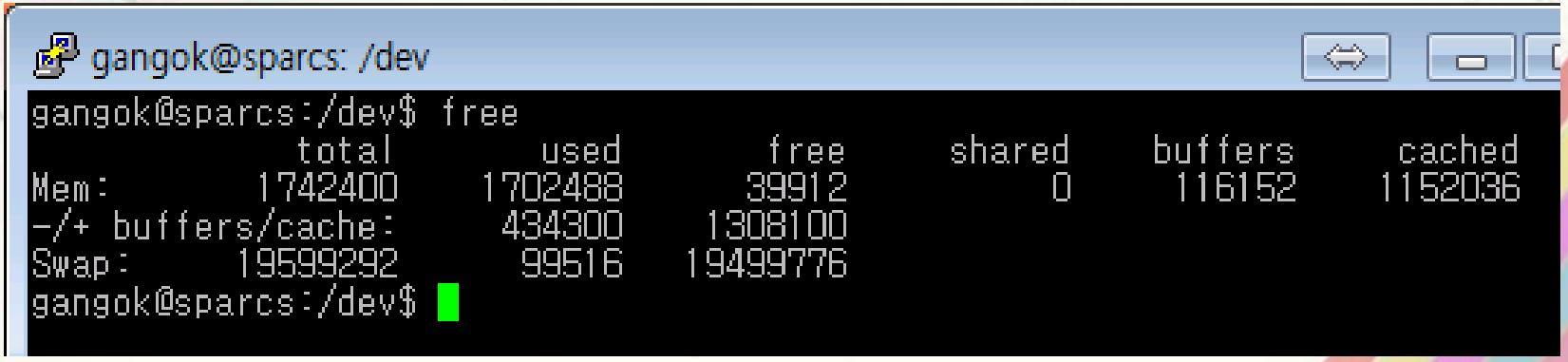
free(명령어)

- 시스템의 메모리 사용 상황을 보여줌



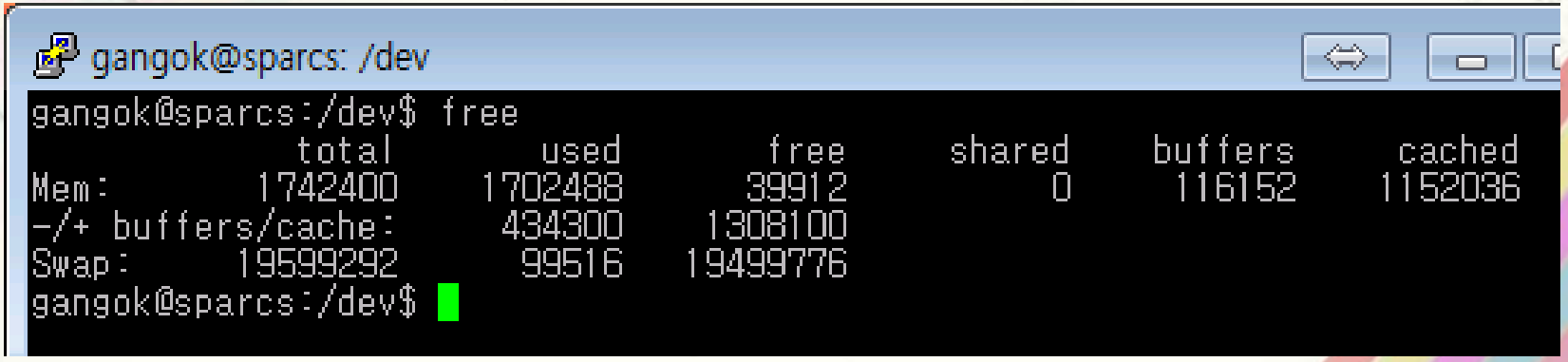
- 표시되는 숫자는 1024byte 블록 단위
  - 1742400 => 1701Mbyte
- -m 옵션으로 MB 단위로 볼 수 있음

free, 첫번째 행(Mem) (1/2)



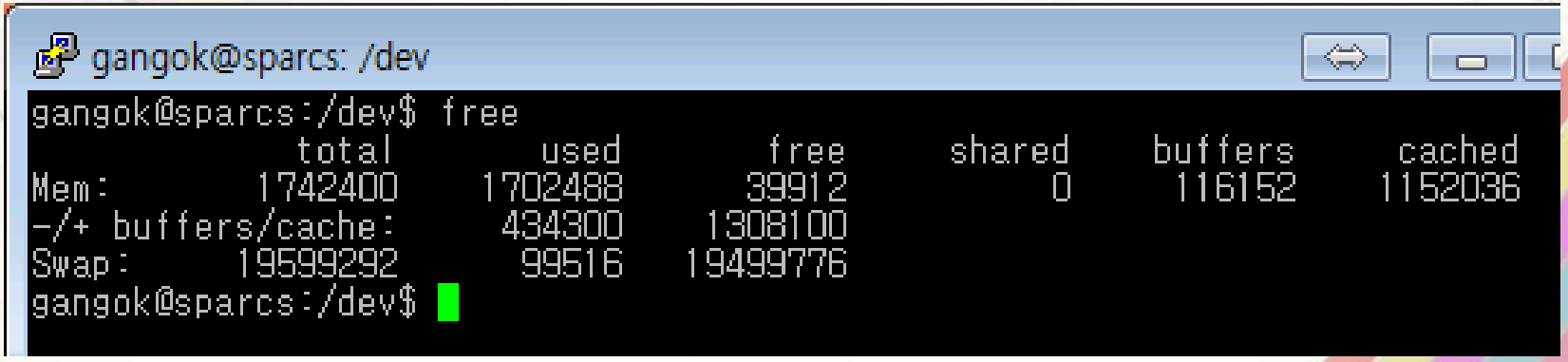
- total : 전체 RAM 공간
  - 실제 물리적인 RAM의 크기에서 커널이 자체적으로 사용하고 있는 메모리를 뺀 양
- used : 사용되고 있는 메모리양
- free : 사용되고 있지 않은 메모리양
- shared : 여러 프로세스가 공유하고 있는 물리적인 메모리양

free, 첫번째 행(Mem) (2/2)



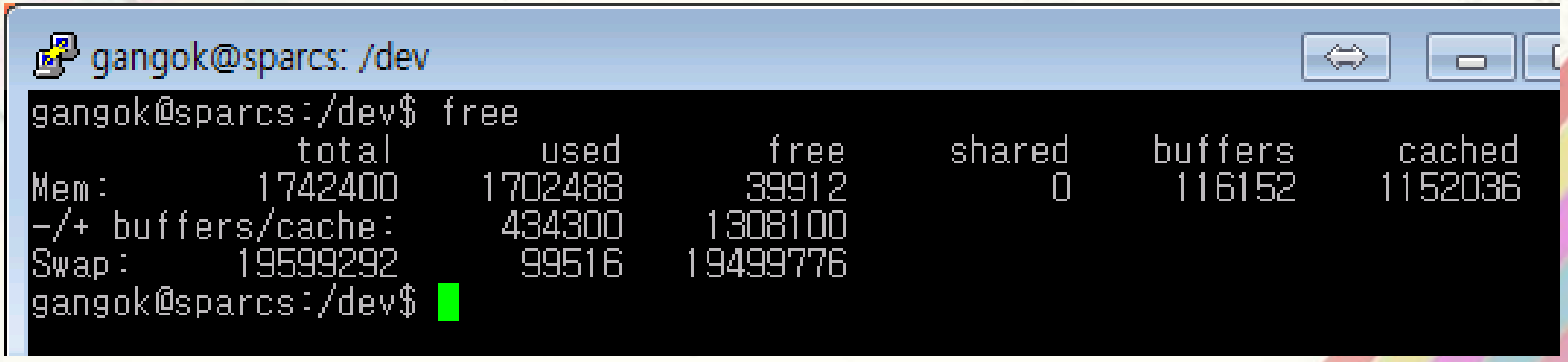
- buffers : 커널 버퍼 캐시에서 사용하고 있는 메모리양
- cached : 커널이 메모리 페이지를 캐시해 둔 양

free, 두번째 행(-/+ buf/cache)



- 버퍼, 캐시로 사용 중인 메모리는 필요하면 application에서 요청하여 사용할 수 있음
- used : application이 사용하고 있는 메모리 양
  - Mem의 used – (buffers + cached)
- free : application이 사용할 수 있는 메모리 양
  - Mem의 free + (buffers + cached)

free, 세번째 행(swap)



- total : 전체 스왑 영역의 크기
- used : 사용중인 스왑 영역의 크기
- free : 사용중이지 않은 스왑 영역의 크기

스왑 파티션 만들기

- fdisk 명령을 통해 swap 파티션을 설정하고
- mkswap 명령을 통해 해당 영역을 포맷한다

스왑 파일 만들기

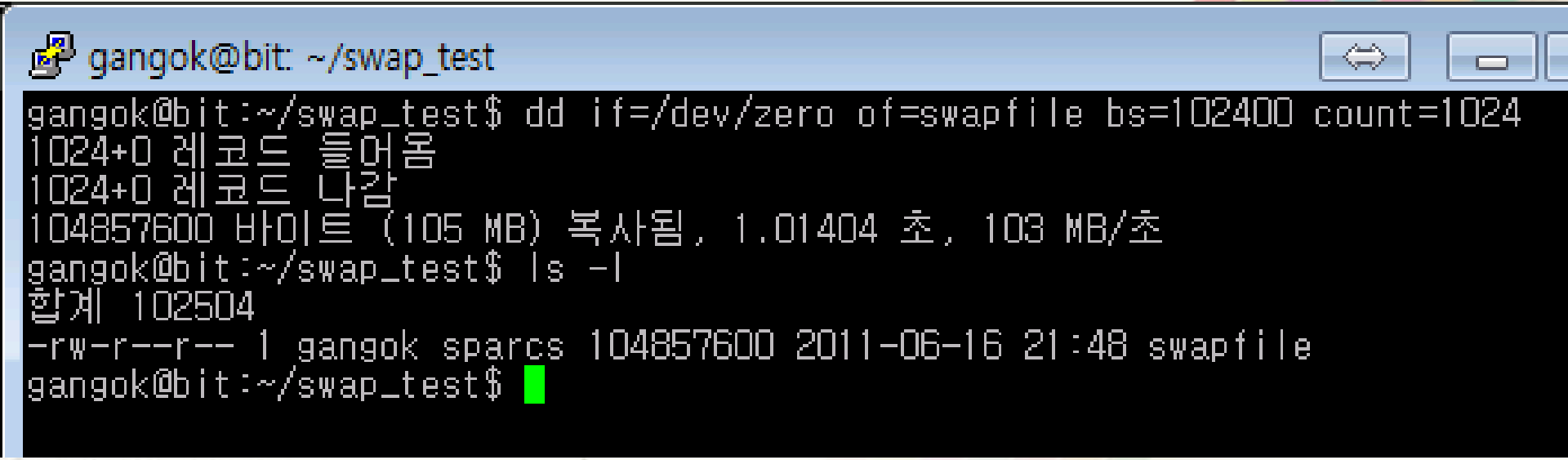
- dd 명령을 통해 빈 파일을 만들고
- mkswap 명령을 통해 해당 영역을 포맷한다

dd(명령어) (1/2)

- $ dd if=/dev/zero of=swapfile bs=1024 count=102400
- 이 명령어는 /dev/zero에서 100MB를 swapfile 파일에 써넣는 것이다
- /dev/zero는 항상 빈 널(null) 바이트를 돌려주는 장치이다
- 이를 하고 나서는 시스템 다운을 대비해 sync를 해주면 좋다

dd(명령어) (2/2)

- 100MB 크기의 파일이 생겼습니다!



mkswap(명령어) (1/2)

- $ mkswap –c [파티션 장치 또는 스왑 파일명] [스왑 영역 크기]
  - 여기서 스왑 영역 크기는 1KB를 한 블럭으로 하는, 블럭 단위의 크기이다
  - -c 옵션은 배드 블럭이 있는지 검사를 하는 옵션이다
  - 이를 한 뒤에는 역시 sync를 하여 포매팅이 확실히 되도록 하는 것이 좋다

mkswap(명령어) (2/2)

root@ubuntu: ~/swap_test

```
root@ubuntu:~/swap_test# mkswap -c swapfile 102400
mkswap: swapfile: warning: don't erase bootbits sectors
        on whole disk. Use -f to force.
Setting up swapspace version 1, size = 102396 KiB
no label, UUID=1cf89064-591e-44a0-89a6-af2085938790
root@ubuntu:~/swap_test# chmod 0600 swapfile
root@ubuntu:~/swap_test# ls -l
total 102400
-rw------- 1 root root 104857600 2011-06-16 09:54 swapfile
root@ubuntu:~/swap_test#
```

- chmod 명령어를 이용하여 퍼미션을 변경해 줘야 한다.

스왑 영역 활성화/비활성화

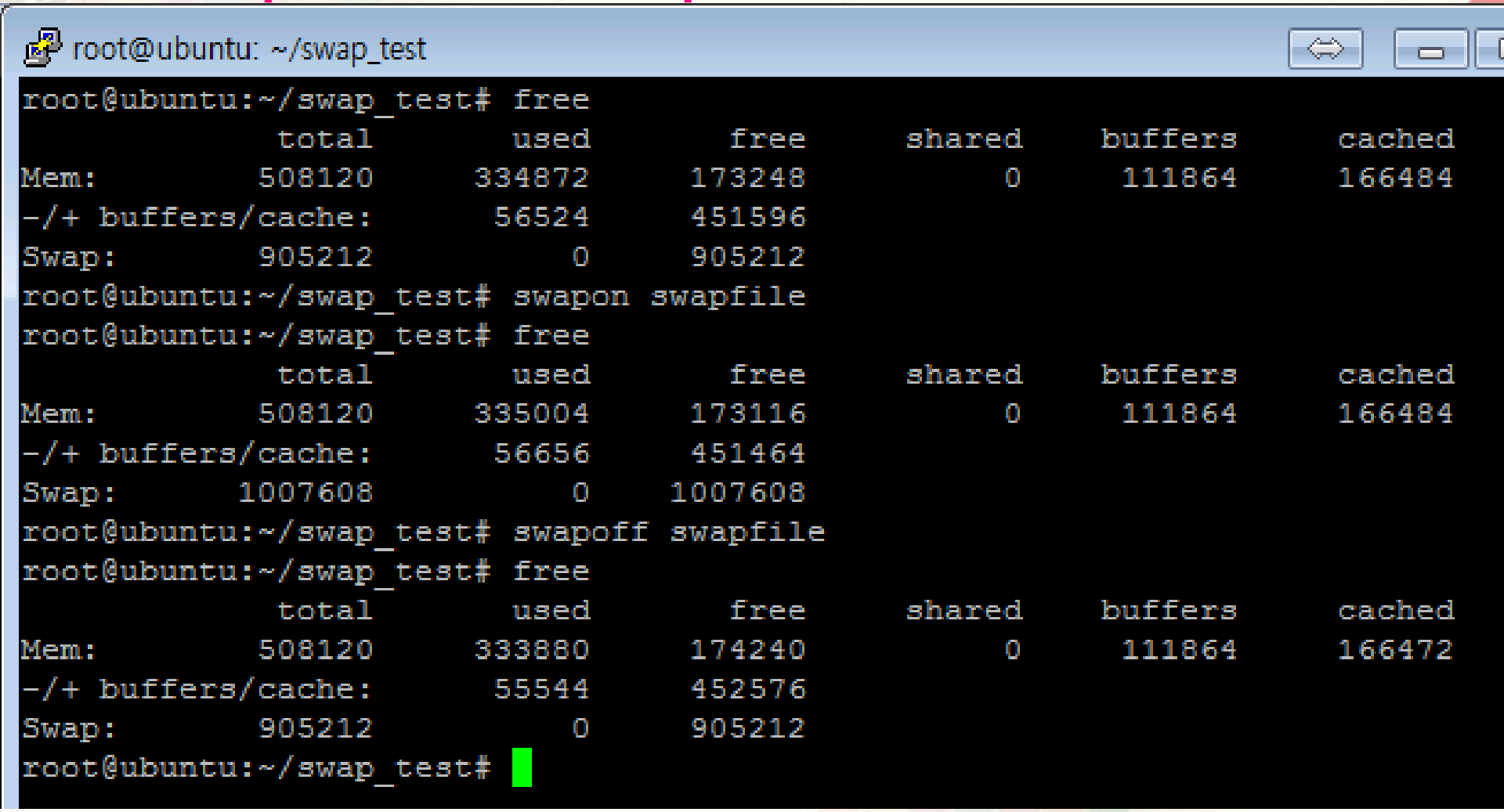
- 새롭게 만들어진 스왑 영역(파일 or 파티션)을 사용하기 위해서는 이를 활성화 시켜야 한다.

- swapon / swapoff 명령어를 이용하여 스왑 영역을 활성화, 비활성화 시킬 수 있다.

swapon / swapoff(명령어) (1/3)

- # swapon [파티션 장치 또는 스왑 파일명]
- # swapon –a
  - /etc/fstab에서 options 필드에 sw라고 적힌 항목은 모두 활성화
  - 부팅시 자동으로 실행
- # swapoff [파티션 장치 또는 스왑 파일명]
  - 파티션 또는 파일을 삭제하기 전 꼭 swapoff 를 해주자!

swapon / swapoff(명령어) (2/3)



swapon / swapoff(명령어) (3/3)

- sparcs.org 서버의 /etc/fstab

gangok@sparcs: ~

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/sda1       /               ext3    errors=remount-ro 0       1
/dev/sda2       /var            ext3    defaults        0       2
# mount home dir to mir
143.248.234.104:/home   /home   nfs     rw,hard,intr,tcp        0
0
/dev/sda3       none            swap    sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/hdb        /media/cdrom1   udf,iso9660 user,noauto     0       0
~
```

- 파일을 삭제하기 전에 꼭 swapoff 할 것!

스왑 영역 실습

- dd 명령을 이용하여 적당한 크기(100M)의 빈 파일을 만든다.
- mkswap 명령을 이용하여 해당 파일을 swap 영역으로 포맷한다.
- swapon / swapoff 명령을 이용하여 swap 영역을 활성화/비활성화 시키면서 free명령을 이용하여 메모리 상태 변화를 관찰한다.
- swapoff로 해당 파일을 비활성화 시킨 후 삭제한다.

장치 파일 관리

device file? (1/2)

- 사용자 프로그램이 커널을 통하여 시스템 하드웨어에 접근할 때 사용

- 단지 프로그램의 관점에서만 '파일'처럼 보일 뿐 단순한 파일이 아니며 파일 시스템 안에 보이는 장치 드라이버의 인터페이스

device file? (2/2)

- 읽고, 쓰고, mmap() 등의 작업을 할 수 있다.

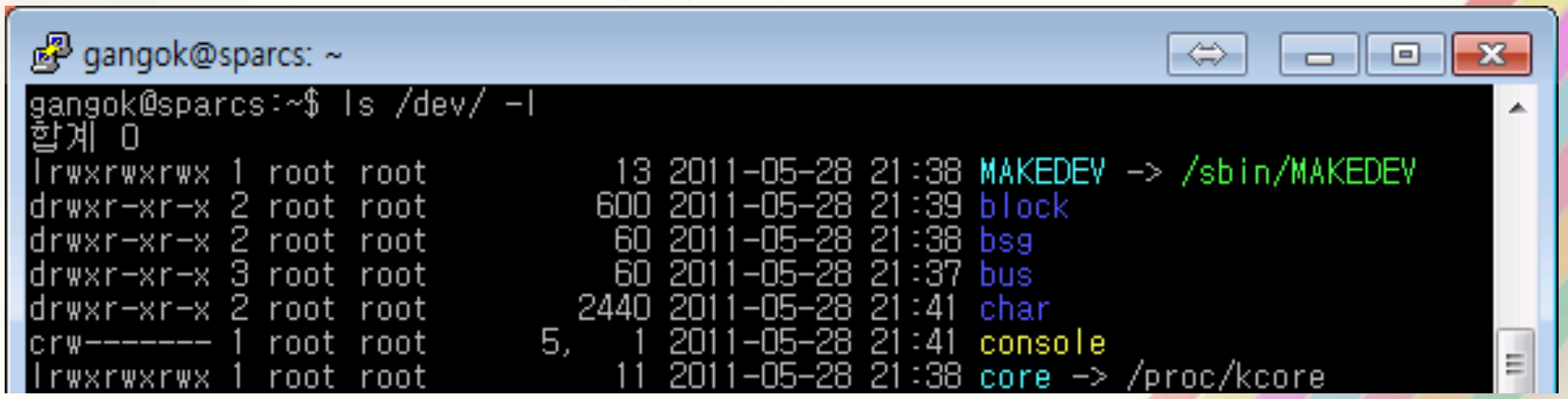
- 이렇게 장치 파일에 접근하면 커널은 I/O 요청을 확인하여 장치 드라이버에게 전달

device file 관리

- 장치파일 보기
  - ls /dev/
- 장치파일 만들기
  - mknod
- symbolic link 만들기
  - ln -s
- 장치파일 삭제하기
  - rm

장치파일 보기 (1/4)

- /dev 디렉토리에 위치



장치파일 보기 (2/4)

gangok@sparcs: ~

```
brw-rw---- 1 root disk      1,   3 2011-05-28 21:37 ram3
brw-rw---- 1 root disk      1,   4 2011-05-28 21:37 ram4
brw-rw---- 1 root disk      1,   5 2011-05-28 21:37 ram5
brw-rw---- 1 root disk      1,   6 2011-05-28 21:37 ram6
brw-rw---- 1 root disk      1,   7 2011-05-28 21:37 ram7
brw-rw---- 1 root disk      1,   8 2011-05-28 21:37 ram8
brw-rw---- 1 root disk      1,   9 2011-05-28 21:37 ram9
crw-rw-rw- 1 root root      1,   8 2011-05-28 21:37 random
lrwxrwxrwx 1 root root           4 2011-05-28 21:38 root -> sda1
lrwxrwxrwx 1 root root           4 2011-05-28 21:37 rtc -> rtc0
crw-rw---- 1 root audio   254,   0 2011-05-28 21:37 rtc0
brw-rw---- 1 root disk      8,   0 2011-05-28 21:38 sda
brw-rw---- 1 root disk      8,   1 2011-05-28 21:39 sda1
brw-rw---- 1 root disk      8,   2 2011-05-28 21:41 sda2
brw-rw---- 1 root disk      8,   3 2011-05-28 21:38 sda3
drwxrwxrwt 2 root root          40 2011-05-28 21:38 shm
crw-rw---- 1 root root     10, 231 2011-05-28 21:37 snapshot
drwxr-xr-x 2 root root          60 2011-05-28 21:38 snd
lrwxrwxrwx 1 root root          24 2011-05-28 21:38 sndstat -> /proc/asound/oss/
sndstat
lrwxrwxrwx 1 root root          15 2011-05-28 21:38 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root          15 2011-05-28 21:38 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root          15 2011-05-28 21:38 stdout -> /proc/self/fd/1
crw-rw-rw- 1 root root      5,   0 2011-06-16 21:17 tty
```

장치파일 보기 (3/4)

```
brw-rw---- 1 root disk      8,    0 2011-05-28 21:38 sda
```

- 제일 앞 열의 문자
  - 일반적인 파일은 -
  - 디렉토리는 d
  - b는 블록 장치
  - c는 문자 장치
- 블록 장치 : 자료의 I/O가 블록 단위로 이루어지며 임의 순서로 접근 가능
- 문자 장치 : 자료의 I/O가 한 바이트 단위로 이루어지며 순차적으로 접근 가능

장치파일 보기 (4/4)

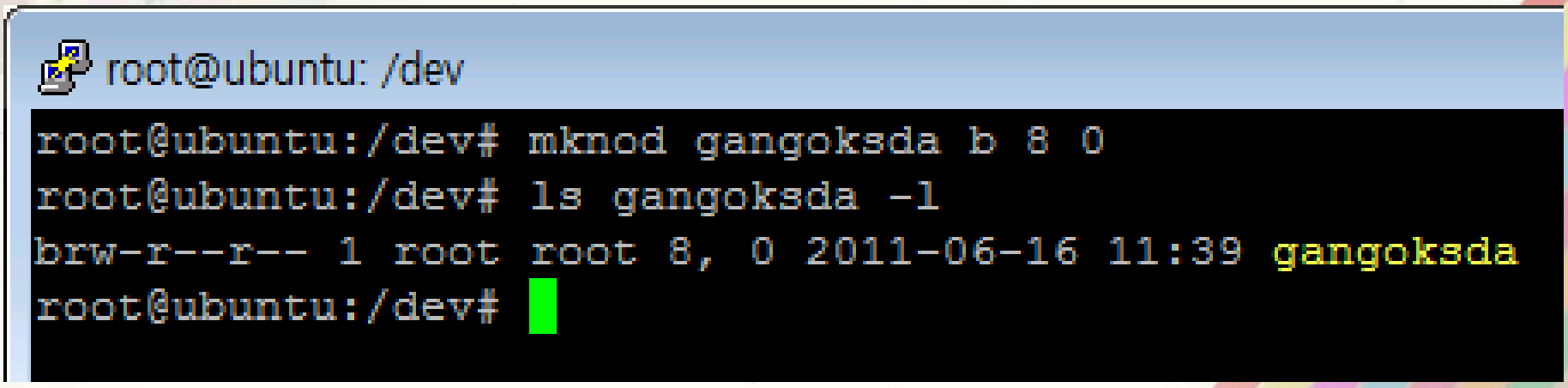
```
brw-rw---- 1 root disk      8,    0 2011-05-28 21:38 sda
```

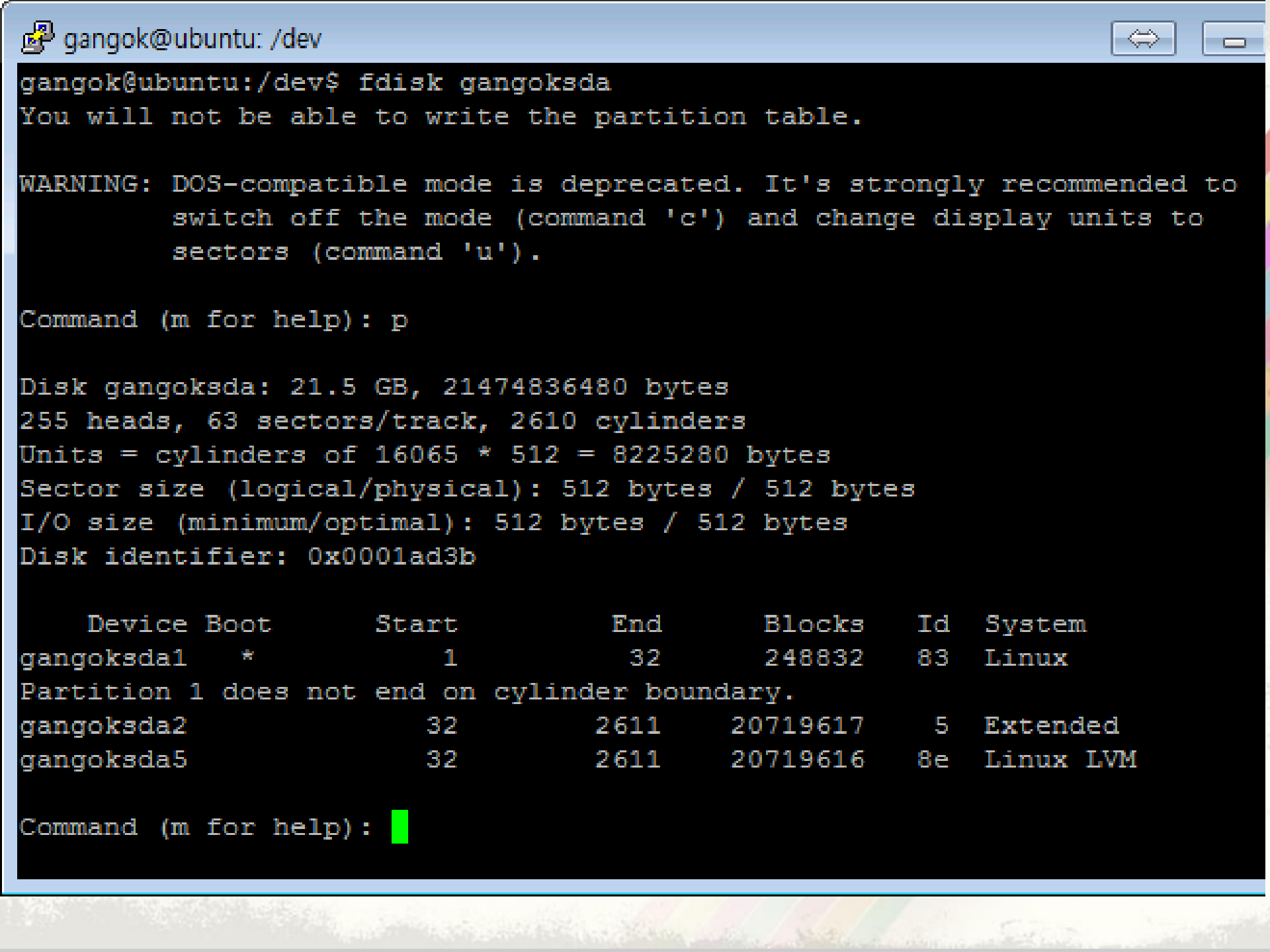
- 크기 필드 부분에 콤마로 구분된 두 개의 숫자가 표시됨
- 첫 번째 수가 메이저 장치 번호
  - 커널 내의 특정 드라이버를 가리킴
- 두 번째 수가 마이너 장치 번호
  - 드라이버 안에서 처리되는 개별 장치
- 프로그램에서 장치 파일에 접근하면 커널은 장치의 메이저 번호, 마이너 번호라는 관점에서 I/O 요청을 받아들임

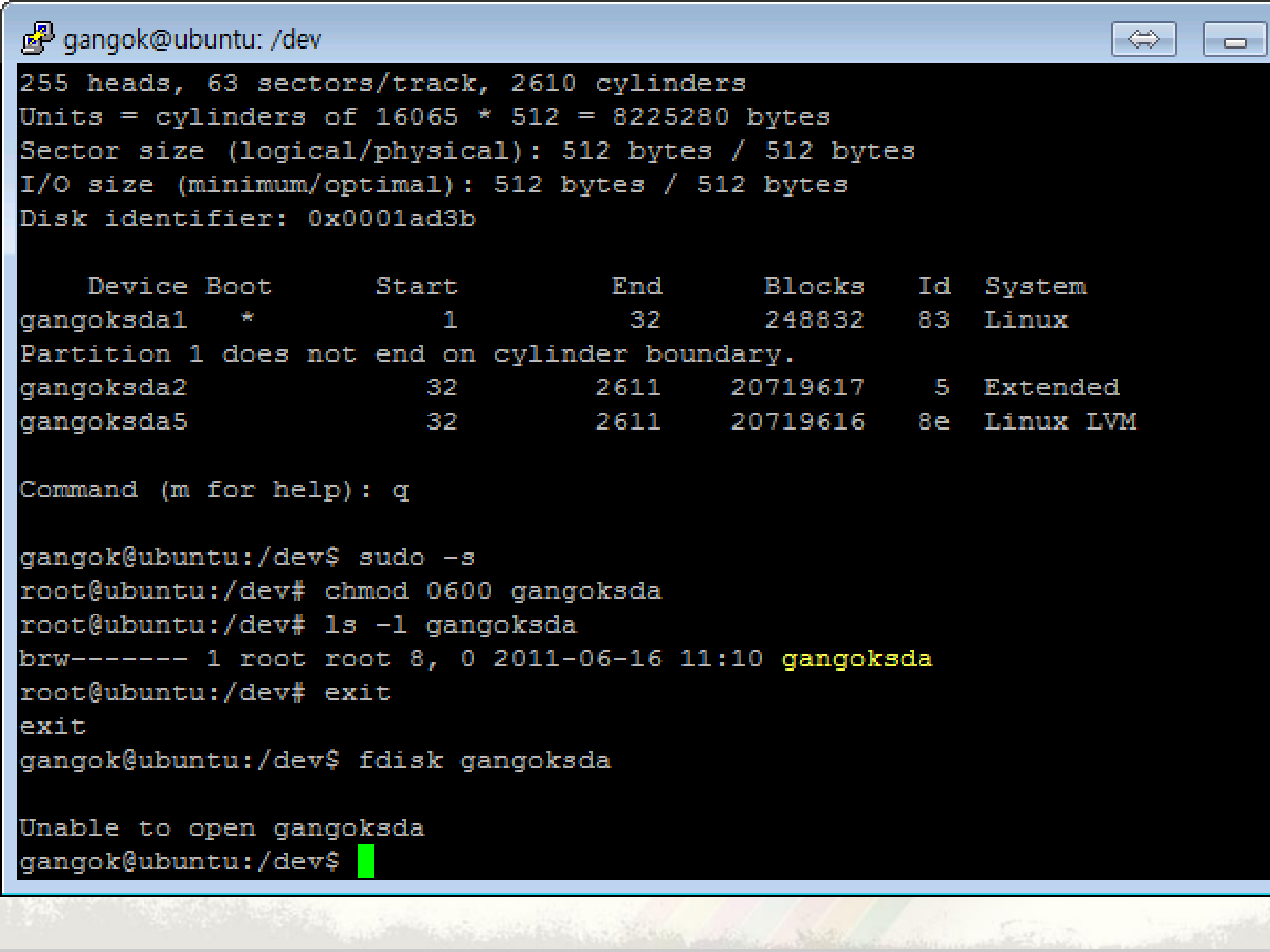
장치파일 만들기 (1/4)

- #mknod –m permissions name type major minor
- name : 만들 장치명 (ex: /dev/rft0)
- type : 문자 장치에는 c, 블록 장치에는 b
- major : 장치의 메이저 번호
- minor : 장치의 마이너 번호
- -m permissions : 선택적 옵션으로, 장치 파일의 퍼미션 설정(ex: -m 660)

장치파일 만들기 (2/4)


root@ubuntu: /dev
root@ubuntu:/dev# mknod gangoksda b 8 0
root@ubuntu:/dev# ls gangoksda -l
brw-r--r-- 1 root root 8, 0 2011-06-16 11:39 gangoksda
root@ubuntu:/dev#


gangok@ubuntu: /dev
gangok@ubuntu:/dev$ fdisk gangoksda
You will not be able to write the partition table.
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help): p
Disk gangoksda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001ad3b
Device Boot Start End Blocks Id System
gangoksda1 * 1 32 248832 83 Linux
Partition 1 does not end on cylinder boundary.
gangoksda2 32 2611 20719617 5 Extended
gangoksda5 32 2611 20719616 8e Linux LVM
Command (m for help):


gangok@ubuntu: /dev
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001ad3b
Device Boot Start End Blocks Id System
gangoksda1 * 1 32 248832 83 Linux
Partition 1 does not end on cylinder boundary.
gangoksda2 32 2611 20719617 5 Extended
gangoksda5 32 2611 20719616 8e Linux LVM
Command (m for help): q
gangok@ubuntu:/dev$ sudo -s
root@ubuntu:/dev# chmod 0600 gangoksda
root@ubuntu:/dev# ls -l gangoksda
brw------- 1 root root 8, 0 2011-06-16 11:10 gangoksda
root@ubuntu:/dev# exit
exit
gangok@ubuntu:/dev$ fdisk gangoksda
Unable to open gangoksda
gangok@ubuntu:/dev$

symbolic link 만들기

- $ ln –s [원본 파일명] [타겟 파일명]

root@ubuntu: /dev

```
root@ubuntu:/dev# ln -s sda harddisk
root@ubuntu:/dev# ls -l harddisk
lrwxrwxrwx 1 root root 3 2011-06-16 11:28 harddisk -> sda
root@ubuntu:/dev#
```

```
drwxr-xr-x  2 root     root          680 2011-06-16 04:08 block
drwxr-xr-x  2 root     root           80 2011-06-16 04:08 bsg
crw-------  1 root     root     10, 234 2011-06-16 04:08 btrfs-control
lrwxrwxrwx  1 root     root            3 2011-06-16 04:08 cdrom -> sr0
drwxr-xr-x  2 root     root         2560 2011-06-16 04:08 char
crw-------  1 root     root      5,    1 2011-06-16 04:08 console
lrwxrwxrwx  1 root     root           11 2011-06-16 04:08 core -> /proc/kcore
drwxr-xr-x  2 root     root           60 2011-06-16 04:08 cpu
crw-------  1 root     root     10,   58 2011-06-16 04:08 cpu_dma_latency
```

장치파일 삭제하기

- rm /dev/gangoksda

```
root@ubuntu:/dev# rm gangoksda
root@ubuntu:/dev# ls gangoksda
ls: cannot access gangoksda: No such file or directory
root@ubuntu:/dev#
```

장치파일을 다룰 때 주의할 점

- 파일 권한 확인!
- 장치 파일은 사용자가 시스템 장치와 상호 작용하는 창구일 뿐
- 장치 파일을 제거한다고 해서 메모리나 커널에서 해당 장치가 제거되는 것은 아님
- 장치 파일을 만든다고 해서 시스템에 장치 드라이버가 추가되는 것도 아님

장치파일 실습

- 장치 파일 만들기
  - 메이저 장치 번호 : 123
  - 마이너 장치 번호 : 456
  - type c
  - 권한 666
  - 파일 이름 dummy
- 실행해보고 결과 확인
  - cat dummy
  - ls > dummy
- dummy 파일 삭제하기