TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

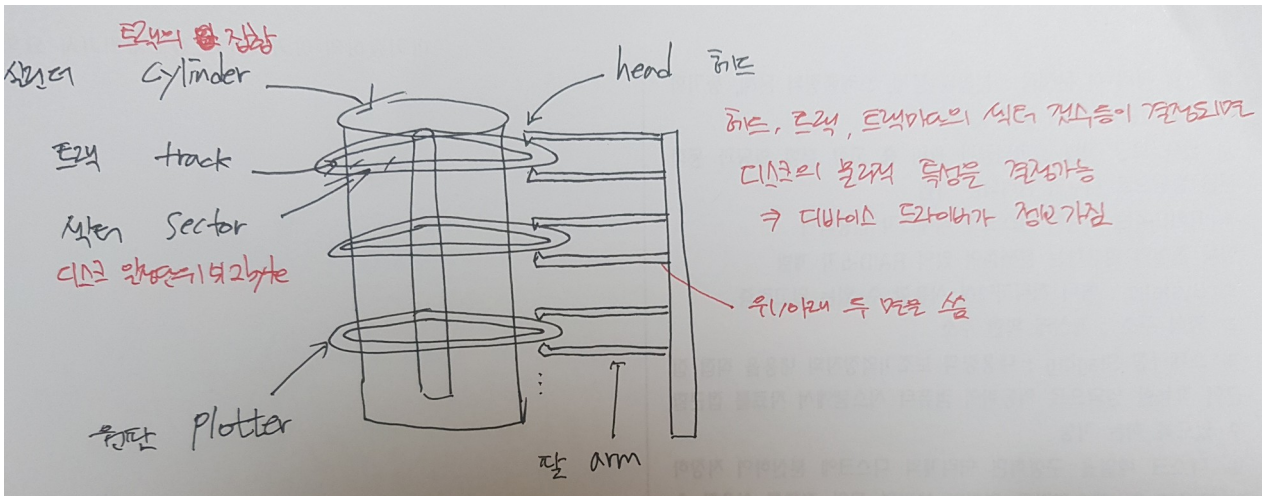
학생 – 문한나

mhn97@naver.com

Chapter 5. 파일시스템과 가상 파일시스템

파일시스템
파일시스템은 이름이라는 속성을 가지고 데이터를 리턴해준다.
이름이라는 속성으로 추상적 객체인 파일을 제공하기 위해서는 메타데이터와 사용자데이터가 필요하다.
메타데이터는 inode 나 superblock 과 같이 파일의 속성 정보나 데이터 블록 인덱스정보 등이 해당되며, 사용자데이터는 실제 기록하려했던 내용에 해당된다. 이러한 내용은 디스크 블록(4kb)을 할당받아서 기록된다.

디스크 구조와 블록관리 기법



디스크에서 데이터를 접근하는 데 걸리는 시간은 탐색시간 > 회전지연시간 > 데이터 전송시간 순으로 느리다.
탐색시간은 데이터를 탐색하고 원판이 돌아오는 시간까지 계산을 해야하고, 회전지연시간은 자기장이 발생해야 하기 때문에 어느 정도 회전속도가 있다. 그리고 전송은 전기적인 신호로 하기 때문에 제일 빠르다고 할 수 있다.

파일시스템은 디스크를 논리적인 디스크 블록들의 집합으로 본다. 실제로는 나눠져 있지 않지만 메모리에 올려야 하기 떄문이다.
일반적으로 페이지 프레임 크기와 같은 4kb 이다.

만약 파일 시스템이 14kb 크기의 파일 생성을 요청받았다면? 4 개의 디스크 블록을 할당해야 할 것이다.
이 디스크 블록을 할당하는 방법에는 속도를 중요시하는 연속 할당과 공간의 효율성을 중요시하는 불연속 할당이 있다.
불연속 할당에 경우에는 파일에 속한 디스크블록들이 어디에 위치하고 있는지에 대한 정보를 기록해 두어야 하는데, 이를 위한 방법으로는 블록체인 기법, 인덱스 블록 기법, FAT 기법 등이 존재한다.

블록체인 기법
블록체인 기법은 각 블록에 포인터를 두어 다음 블록의 위치를 기록하여 연결해 놓는 방법이다.
이 기법을 사용하여 파일의 끝 부분을 읽으려는 경우에는 처음부터 읽어야 하며, 중간에 한 블록이 유실된 경우 나머지 데이터까지 모두 잃게 된다.

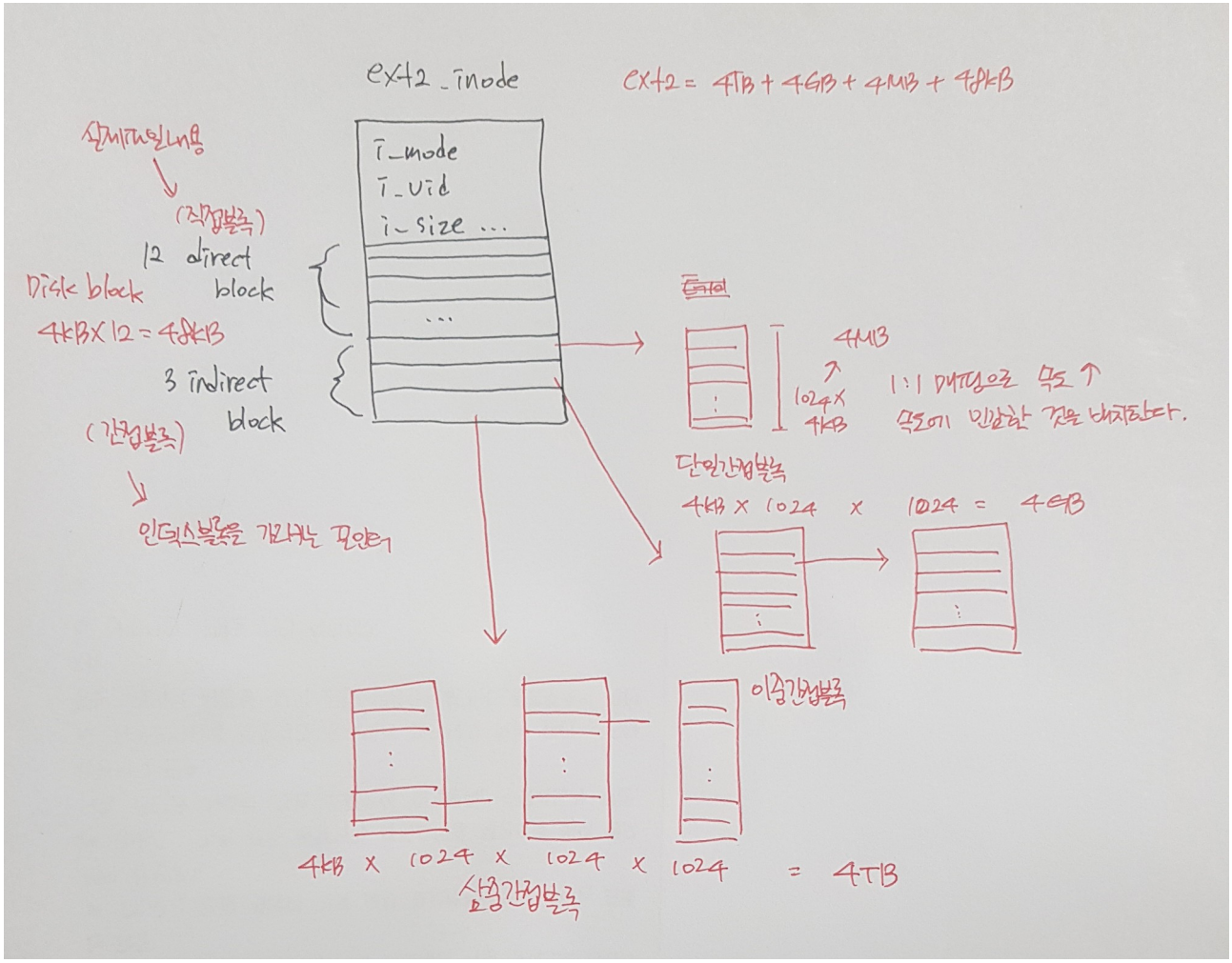
인덱스 블록 기법
인덱스 블록 기법은 블록들에 대한 위치 정보들을 기록한 인덱스 블록을 따로 사용하는 기법이다.
이 기법을 사용하면 원하는 위치의 정보만 읽을 수 있지만 만약 블록이 유실된다면 파일의 데이터 전체를 잃게된다.

FAT 기법
FAT 기법은 같은 파일에 속해 있는 블록들의 위치를 FAT 라는 자료구조에 기록해 놓는 기법이다.
이 기법은 데이터를 찾아갈 때 파일시스템이 관리하는 공간 내에 전역적으로 존재하는 FAT 구조를 사용하며, 중간에 데이터가 유실되어도 복구가 가능하다. 하지만 테이블 자체가 유실되면 복구가 불가능하며, 이를 위해 FAT 내용을 FAT 구조 파일시스템이 중복으로 관리한다.

FAT 파일시스템
파일시스템들은 저마다 자신만의 디렉토리 구조체를 선언해 놓은 뒤, 파일이 생성될 때, 이 구조체의 각 내용들을 채워서 디스크에 저장한다. 리눅스의 파일시스템은 수백~수천개인데 이를 관리하기 위해 함수포인터를 사용한다.

파일시스템은 사용자가 요청한 이름을 가지고 디렉토리 엔트리를 찾게 되는데, 어떻게 찾을 수 있을까?
우선 슈퍼블록으로 루트의 데이터블록을 찾는다. 그 후 블록에서 다음으로 찾아가야 할 번호를 찾아 그 블록으로 가서 파일속성을 가진 엔트리를 찾아 사용자에게 제공해준다.

inode 구조



하지만 총 4TB 로도 커버가 안되는 구글서버를 위해 ext4 가 개발되었다.

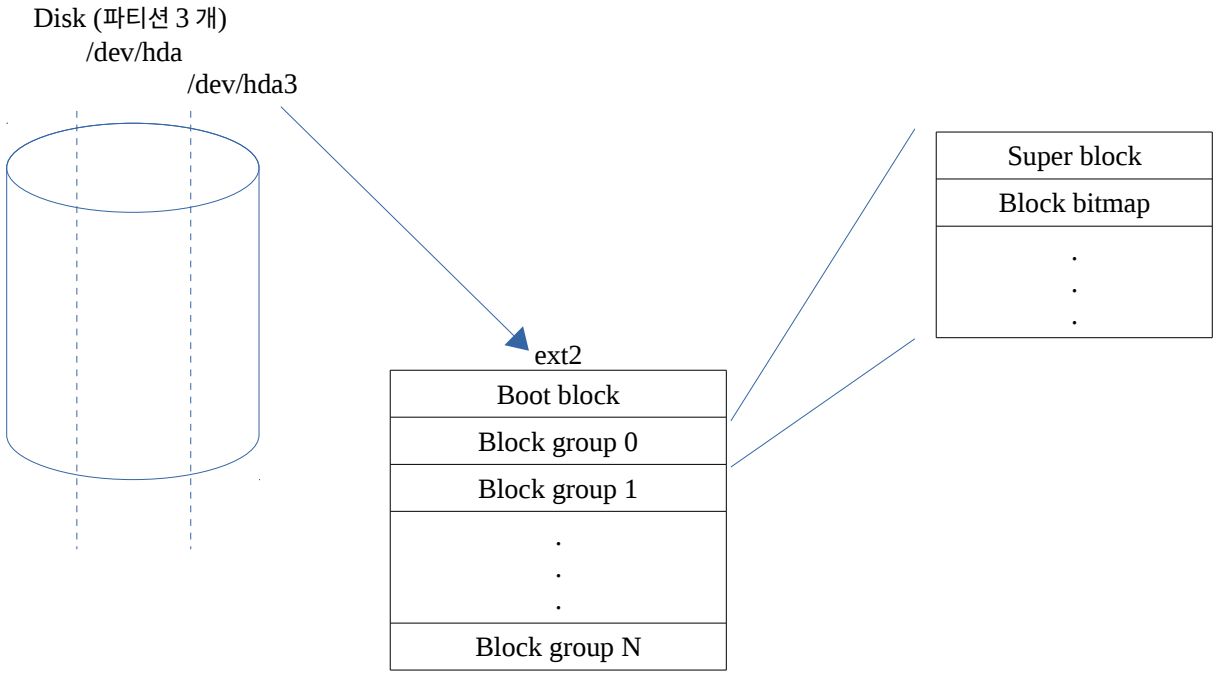
ext2 파일시스템

ext2 파일시스템을 기반으로, 다양한 형태의 결함에 강인한 저널링 기능을 추가한 것이 ext3 와 ext4 파일시스템이다.
따라서 ext 파일시스템끼리는 전부 호환이 가능하다.

파일시스템은 디스크를 관리한다. 디스크 방식에는 IDE 방식과 SCSI 방식 2 가지가 존재하며,
만약 IDE 방식이라면 /dev 디렉토리에 “hd”라는 이름의 블록 장치 파일로 접근하고, SCSI 방식이라면 “sd”라는 이름으로 접근한다. 대부분 SCSI 방식을 사용하며, 성능 또한 압도적이다.

```
mhn@mhn-900X3L:~/kernel/linux-4.4$ 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=1949864k,nr_inodes=487466,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=395240k,mode=755)
/dev/sda8 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/s
```

디스크가 장착되면 파티션을 나눌 수 있으며, 최대 64 개까지 분할할 수 있다.
파일 시스템은 각 파티션 당 하나씩 만들어지며, ext2 에서는 “mke2fs”라는 명령어를 사용한다.
파일시스템을 만들면 그 파티션은 복수개의 블록그룹으로 나뉜다.



ext2 파일 시스템은 디스크의 성능을 높이기 위해 같은 파일에 대한 inode 와 디스크 블록들을 인접한 실린더에 유지하여 디스크의 탐색시간을 줄인다.

가상파일시스템
사용자 태스크들은 시스템 호출을 통해 파일시스템에 접근한다.
하지만 리눅스의 파일 시스템은 한두가지가 아니다. 이를 고려하여 파일시스템과 사용자 태스크 사이에 여러 포맷들을 다 커버할 수 있는 가상적인 층을 만들었다. 이것이 VFS 이다.
VFS 는 사용자가 시스템 콜을 호출하면 파일의 인자를 확인하여 구조체를 만든다. 이 구조체를 인자로 하여 파일시스템에 요청을 하면 파일시스템은 inode 를 활용하여 넘겨받은 구조체에 값을 넣어준 후 리턴하는 방식으로 사용된다.

VFS 객체

수퍼블록	관리하고 있는 파티션의 고유 정보를 저장
아이노드	특정 파일과 관련된 정보를 저장
파일	태스크가 open()한 파일과 연관되어 있는 정보를 관리
디엔트리	태스크가 접근하려는 파일의 아이노드 객체와 태스크와 연관된 파일객체를 빠르게 연결시켜 주는 일종의 캐시 역할을 수행