I DSP,Xilinx zynq FPGA,MCU 및 Xilinx zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)
Gccompil3r@gmail.com
학생-윤지완
Yoonjw789 @naver.com

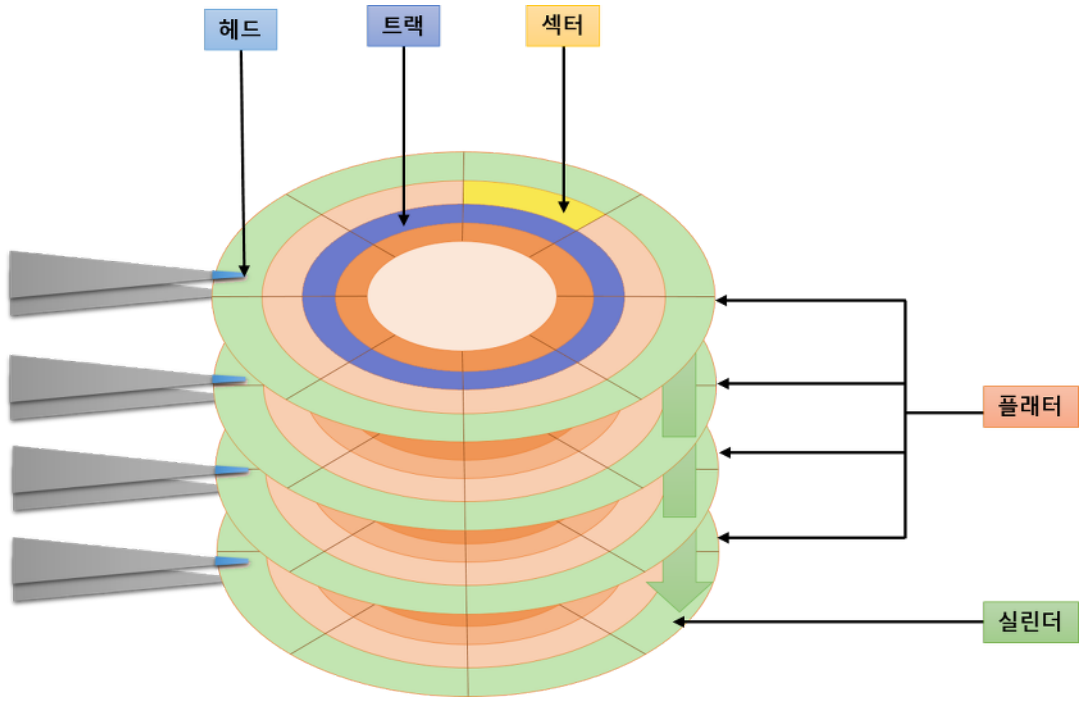
linux 5 장

1.파일시스템과 가상 파일 시스템

메모리 관리 기법과 파일 시스템은 모두 기억 장치를 관리한다. 기억장치는 ram 이던 디스크건 공간이 한정적이여서 최대한의 공간으로 사용해야 하는데 여기서 앞서 배운 버디나 slab 이 내부,외부 단편화를 줄이는것에 노력했는데 메모리 관리 기법도 단편화를 최소화 해야 한다.
메모리 관리 기법과 파일시스템 차이점
:이름이라는 특성을 가지고 있다.

디스크 블록 번호를 외우는것보단 파일 이름을 외우고 있는것이 사용자가 편하다.(ex 약속.txt 라는 파일 이름)

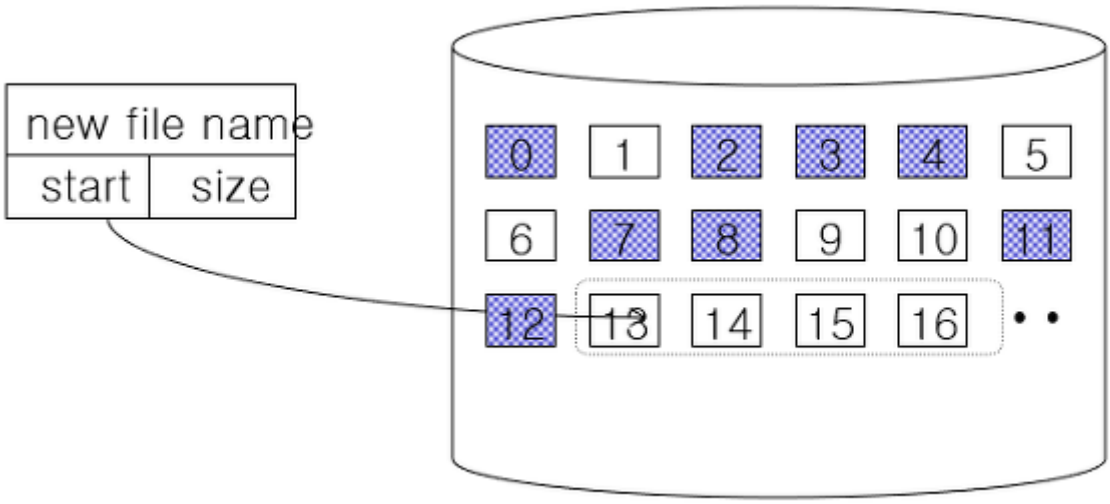
2.디스크 구조와 블록 관리 기법



원판 자체가 cylinder 가 아니라 track 이라는 집단이 일열로 나열되었는것을 cylinder 라고 한다.섹터는 데아터를 읽거나 쓸때 각 면의 기본 단위이고 단위는 512byte 이다.
디스크에서 데이터를 접근하는 시간 탐색 시간,회전지연 시간,데이터 전송시간이 있으며 이중에서 제일 속도가 느린것은 탐색 시간이다.

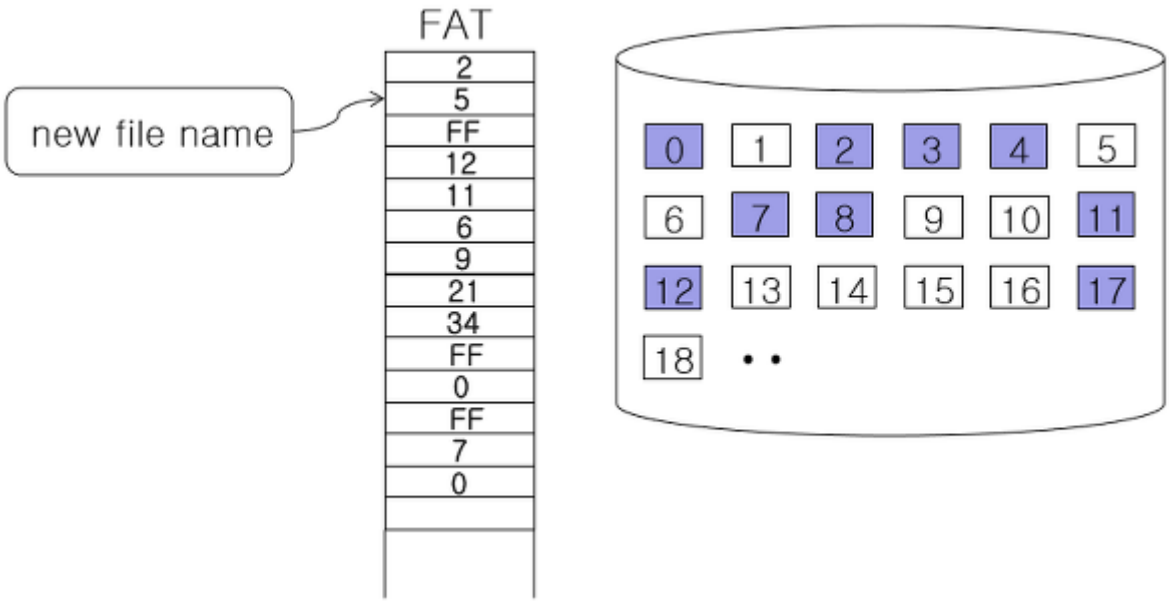
탐색 시간:헤드를 요청한 데이터가 존재하는 트랙 위치까지 가는데 이동 시간.

아래 그림을 보면 파일 시스템은 디스크를 물리적인 구조로 보지 않고 디스크 블록으 집합으로 이루어져 있다.디스크 블록의 크기는 페이지 frame 과 동일하다.파일 시스템 최대의 병목은 디스크 I/O 이다.
따라서 최근에는 디스크 블록의 크기가 상향이 되었다(4KB*2 의 N 승).
디스크 크기->증가=속도 증가 but 공간 효율성은 감소.
디스크 할당 방법:연속적 할당,불연속적 할당.
연속적 할당:아래 그림이 연속 할당의 그림이다.1,5,6,9,10 은 서로 떠렁져 있어지만 13,14,15,16 은 서로 바로 옆에 붙어있기 때문에 탐색 속도가 빨라진다는것을 알 수 있다.하지만 단점은 공간 효율성이 떨진다.
그래서 이부분을 보완하고자 불연속 할당을 쓴다,만약 17kb 이를 할당해야 되는데 연속적인 블록은 총 12kb 이면 나머지를 추가로 할당을 해야하지만 연속적인 할당은 불가능 하다.그래서 나머지 부분을 불연속 할당으로 채우면 된다.여기서 주의해야 알점은 파일 크기가 커질 때 기존에 있던 블록들을 다른 곳으로 복사하는 것은 성능 상 매우 큰 문제가 된다는것을 명심하자.결론 불연속 할당은 공갔거 효율성이 좋지만 속도는 떨어진다는 특징이 있다.



불연속 할당 방법은 파일에 속한 디스크 블록들이 어디에 위치하고 있는지에 대한 정보를 기록해 두어야 하는데 3 가지 방법이 있다.

블록체인 할당:파일에 속한 디스크 블록들을 체인으로 연결해주는것
->연결 리스트에서 link 로 다음 주소를 가리키는 방식과 동일하다.
포인터를 사용해 찾아가는 서비스.
단점:만약 lseek()와 같은 시스템 콜을 하여 읽을 라고 하면 처음부터 다시 읽어야 하며 중간에 데이터가 손실이 되었다면 모든 데이터들을 잃어 버린다.
인덱스 할당:각 블록들의 위치정보를 저장해놓는 방식으로 lseek()을 할 때도 처음부터 읽지않아도 된다.
장점:순차 접근 , 직접 접근 ,외부 단편화가 없다.
단점: 블록 주소를 따로 저장해야 되기때문에 별도로 다른 공간을 만들어야 하며 만약 index block 을 잃어버릴 경우 data 전체를 완전히 손실이 된다.
FAT 할당:FAT 라는 자료구조에 기록해 놓는 방식
index 할당과 다른점은 index 는 각 정보마다 index 블록이 필요한데 FAT 는 하나의 FAT 로 파일시스템을 관리하는 것이다.



FF:파일의 끝

0:free 상태

사진을 보면 처음 FAT 구조의 첫번째 블록(index[1])을 가리키며 첫 번째 블록이 5 번을 가리키고 있으며 5 번지에 가보면 다음 6 번지를 가리키고 있으며 6 번지는 9 번지, 9 번지는 불연속 할당의 끝이므로 FF 르 가리키는 형식으로 진행한다.

FAT 방식은 언 블록이 비어있는지 쉽게 할수 있으며 파일 엑세스가 쉽다.하지만 이것도 index 와 같이 별도의 정보를 담을 구조를 만들어야 하기 때문에 FAT 구조가 손실된다면 DATA 전체가 소실된다.

자주 사용되는 파일 시스템

Ext2:20 년전에 만든 시스템, 4TB 가 한계,현대에서는 이 방식을 쓰기에는 데이터량이 너무 커졌다 그래서 이것을 보완하는것이 있다.

Ext4:어떤 상황에서도 데이터 복원이 가능하고 가장 중요한것은 이 시스템의 사이즈는 infinite 이다.

```
struct msdos_dir_entry {
        __u8    name[MSDOS_NAME];/* name and extension */
        __u8    attr;           /* attribute bits */
        __u8    lcase;          /* Case for base and extension */
        __u8    ctime_cs;       /* Creation time, centiseconds (0-199) */
        __le16  ctime;          /* Creation time */
        __le16  cdate;          /* Creation date */
        __le16  adate;          /* Last access date */
        __le16  starthi;        /* High 16 bits of cluster in FAT32 */
        __le16  time,date,start;/* time, date and first cluster */
        __le32  size;           /* file size (in bytes) */
};
```

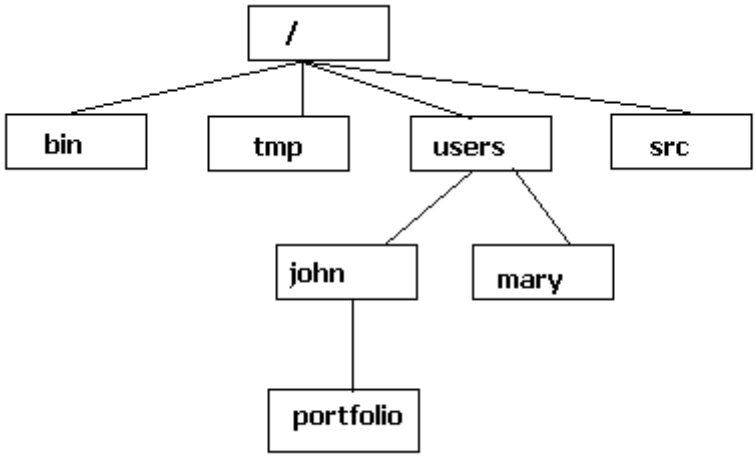
<FAT 의 directory entry>

위에 코드에서 STATHI 의 동작은 FAT 를 통해 데이터를 indexing 하기 위해서 첫 블록의 번호를 저장하는 변수.

여기서 의문점.

파일 시스템은 어떻게 엔트리를 찾을까?

순차 탐색도 방법이나 속도가 너무 느리다.

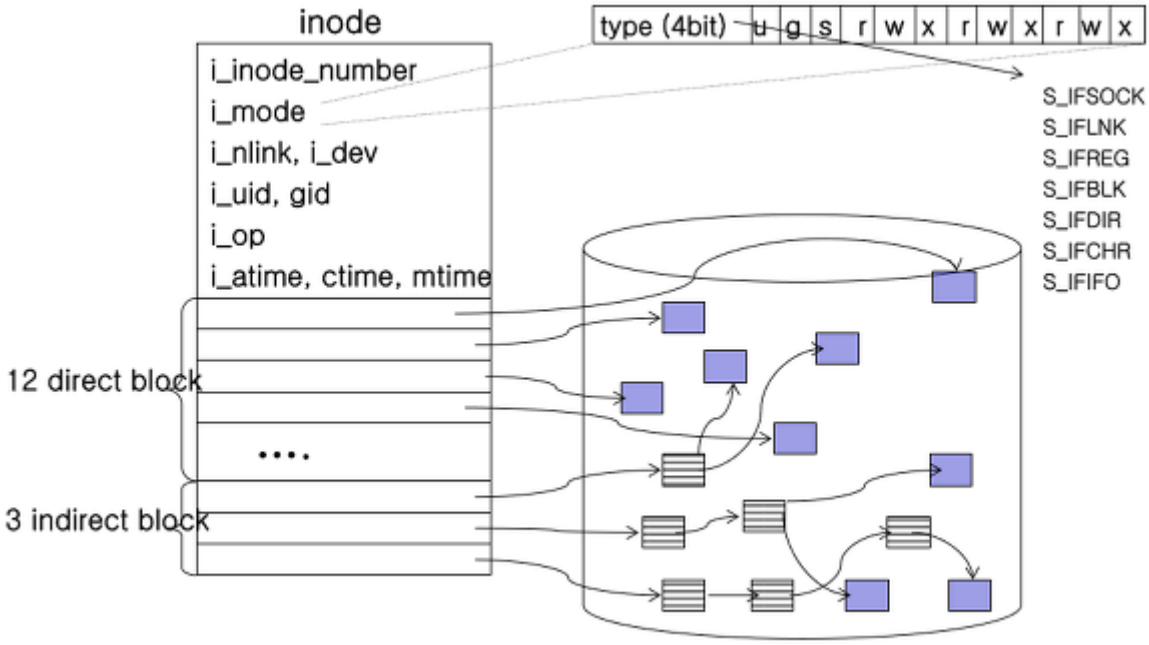


이와 같은 방법을 사용한다.
그림을 보면 알겠지만 효율적인 탐색을 하기 위해 트리 구조를 사용했다.

ex)/users/john/portfolio 의 파일 읽을라고 한다.
그럼 처음 "/"에서 다음 테이블 블록의 주소를 찾는다.만약 users 의 addr 이 6 이면 6 번지를 찾아간다.그 다음 6 번지에서 john 의 addr 을 찾아서 john 의 디렉토리를 보고 portfolio 를 찾아주는 방식이다.

3.inode

리눅스 디폴트 파일 시스템인 ext 계열은 inode 구조를 가지고 있다.



상위 4bit : 파일의 유형을 나타낸다

S_IFREG : 정규파일

S_IFDIR : 디렉토리

S_IFCHR : 문자 장치 파일

S_IFBLK : 블록 장치 파일

S_IFLINK : 링크 파일

S_IFFIFO : 파이프

S_IFSOCK : 소켓

12direct block : 직접 블록

실제 파일의 내용을 담고 있는 디스크의 데이터 블록을 가리키는 포인터

12direct block 에서 각각 한 블록당 4kb 이므로 총 48kb 이다.하드 디스크는 이것 보다 훨씬 용량이 크기 때문에 이걸로 는 부족하다.

3indirect block : 간접 블록

Index Block(디스크 블록을 가리키는 포인터를 갖는 블록)

그래서 추가한것이 3indirect block 이다. 위에 사진을 보면 3 개의 블록들이 각각 하나의 인덱스를 가르키고 있으며 하나의 index block 은 1024 개의 포인터를 사용 할 수 있다.그래서 단일 간접 블록이 지원하는 파일 크기는 1024*4kb=4MB 이다.하지만 이걸로는 훨씬 부족하다. 지금 저 기준은 32 비트 기준이고 64 비트는 너 엄청난 용량을 사용하는데 4TB 으로는 부족하기에 저 사진에서 더 추가를 해야 한다.

3indirect 에서 index block 을 한 개만 가르키고 있는것이 단일간접 블록이다.

2 중 간접 블록의 파일 크기는 1024*1024=4GB

3 중 간접 블록의 파일 크기는 1024*1024*1024=4TB 정도 이다.

Ext2:32bit

Ext4:64bit