Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

날 짜 : 2018 . 4. 12

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

< 리눅스 커널 내부 구조_ Chapter_ 5>

1. 파일 시스템 일반

1. disk 도 내, 외부 단편화가 일어날 수 있다.
2. 파일 시스템 종류가 여러개라 종류마다 소프트웨어적 처리가 다르다.

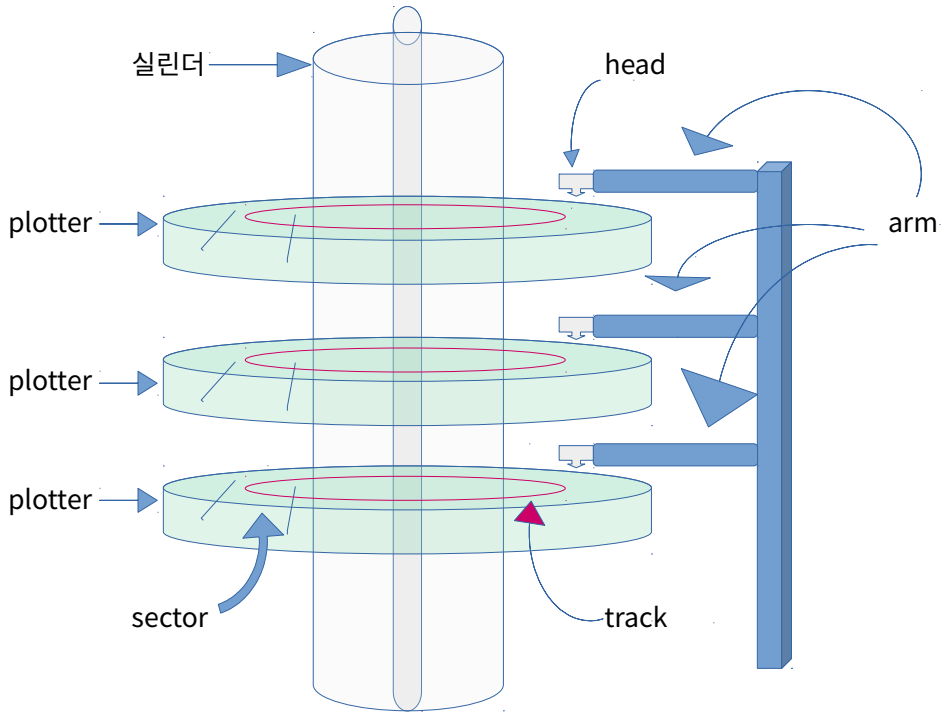
- 메모리 관리 기법과 파일 시스템은 모두 기억 장치를 관리한다.
  따라서, 메모리 관리 기법과 파일 시스템 모두 내부/외부 단편화를 최소화 해야 하기 위해 노력 해야 한다.

- 메모리 관리 기법과 파일 시스템의 차이는 '이름' 이다.
메모리 관리 기법 : 파일시스템 – 이름
파일시스템 : 메모리 관리 기법 + 이름

- 디스크에 저장하는 데이터는 크게 메타 데이터 (inode, 슈퍼 블럭) 와 사용자 데이터(user data)
메타 데이터 : 파일의 속성 정보, 데이터 정보, 인덱스 정보.
사용자 데이터 : 사용자가 실재 사용하려고 했던 데이터.

- 실재 할당하는 녀석은 '디바이스 드라이버' 이다. (디스크 블럭(4KB)등을 할당)

2. 파일 시스템 일반



– 디스크 구조 –

- 디스크는 원판, 팔 , 헤드 로 구성된다. 원판에 track 이 존재 하며, 같은 위치의 track 집합을 실린더라 한다.
- track 은 다시 몇개의 섹터로 구분된다. 섹터의 크기는 512byte(용량이 4k 일때, 섹터 8 개가 모이면 track)
- 헤드는 원판의 읽기/ 쓰기가 가능한 면마다 하나씩 존재 한다.
- track 마다 섹터가 몇개 인지 결정이 되면 전체 용량이 몇인지 알수 있다.

- 디스크에서 데이터에 접근하는데 걸리는 시간.
1. 탐색 시간. (헤드가 요청한 데이터 위치로 이동하는데 걸린 시간, 제일 느리다)
2. 회전 지연 시간. (요청한 섹터가 헤드 아래로 위치 될 때까지 디스크 원판 돌린 시간)
3. 데이터 전송 시간. (헤드가 섹터의 내용을 읽기/쓰기 하는데 걸리는 시간, 제일 빠르다.)

- 파일시스템은 물리적인 구조로 보지 않는다. 대신 디스크를 논리적인 디스크블록들의 집합으로 본다.

- 디스크 블록은 0, 1, 2, 등의 논리적인 번호를 하나씩 갖는다.
- 디스크 블록의 크기는 일반적으로 페이지 프레임과 같다. 4KB 이다.

- 파일 시스템 성능의 최대 병목 요소는 디스크 I/O 이다.
최근에 개발된 파일 시스템의 경우 버디할당자 처럼 (4K x 2^n)으로 블록 크기를 크게 할당한다. (빠름)

- 만약 이런 환경에서 파일 시스템이 14KB 크기의 파일 생성을 요청 받으면,
버디할당자 방식처럼 4 개의 디스크 블럭이 필요한 것이다.

- 디스크 블록은 2 가지 할당 방식이 있다.
연속 할당 : 통으로 가져와 탐색이 빠르다(탐색이 필요 없이 일렬로 되어 있음)
불연속 할당 : 공간 활용에 좋다. 대신 속도가 좀 느림.(빈공간에 마구 끼어 넣어도 되어서)

*디스크 조각 모음을 하면 일렬로 배치해 주어 속도가 빨라진다.

- 파일 크기가 커질 때 기존에 있던 블록들을 다른 곳으로 복사 하는 것은 성능 상 매우 큰 문제를 일으킨다.
따라서 설계시에 연속할당 방법만 사용하는 경우는 없다.

– 불연속 할당 방법 –

1. 블록 체인 기법.(비트코인 X) : 파일에 속한 디스크 블록들을 체인으로 연결해 놓는 방법.(블록에 포인터 둠)
(lseek() 이용해 끝을 읽을려 할때 어쩔수 없이 앞 데이터 블럭을 읽어야 한다.)

New file name	
start	size

2. 인덱스 블록 기법. : 블록들에 대한 위치 정보들을 기록한 인덱스 블록을 따로 사용한다.

New file name	
Index	size

3. FAT(File Allocation Table) : 인덱스 블록 기법은 파일마다 인덱스가 필요하지만 FAT 기법은 파일시스템은 전체적으로 하나의 FAT 가 존재. (FAT 자료 구조 기법: 블록체인 + 인덱스 블럭)

New file name	
start	size

* root file system 은 언제나 ‘인덱스 2’ 에 걸린다.
* 인덱스 기법은 인덱스 블럭이 유실되면 파일의 데이터 전체가 날아가는 것이다.
* FAT 기법도 FAT 구조가 유실 되면 파일 시스템 내의 전체 파일이 날라간다.

(the art of the computer programing - 도널드 크누스 - 이책에 알고리즘이 엄청나다.)