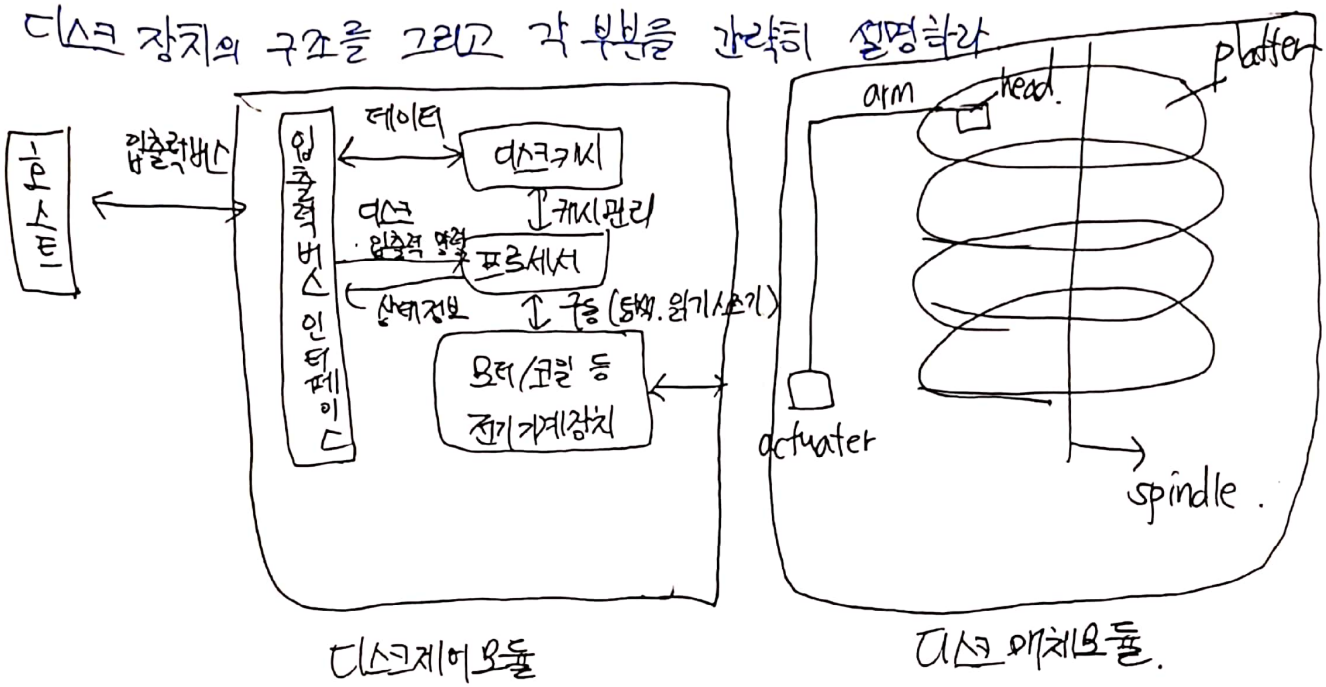
11장 파일 시스템관리 과제

1731017 >임민경

1. 디스크 장치의 구조를 그리고 각 부분을 간략히 설명하라.



디스크 장치는 크게 디스크 제어모듈과 디스크 매체 모듈로 나뉜다.
디스크 제어모듈은 입출력 버스 인터페이스, 디스크 캐시, 프로세서, 전기 기계장치 등으로 구성된다. 입출력 버스 인터페이스는 디스크 장치의 외부로서 컴퓨터 내 다른 하드웨어와 연결되는 부분으로 이 부분을 통해 데이터와 명령이 오고간다.
프로세서는 운영체제로부터 디스크 입출력 명령을 전달받고 해석하여 모터/코일 등의 전기 기계 장치를 구동시키고, 내부에 있는 명령 큐에 다수의 입출력명령을 받아 놓고 스케줄링을 통해 입출력 성능을 높이며 디스크 캐시를 관리한다.
디스크 캐시는 (MB에서 몇 십 MB 크기로 디스크 내부의 버퍼 역할을 하는 빠른 반도체 메모리이다. 데이터는 디스크 캐시에 저장되고 프로세서가 디스크 캐시의 데이터를 플래터에 저장한다. 읽기의 경우는 프로세스의 지시를 받아 플래터로부터 읽혀진 데이터는 먼저 디스크 캐시에 저장된 후 입출력 버스 인터페이스에 의해 운영체제가 요청한 메모리 영역으로 복사된다
디스크 매체 모듈은 크게 플래터, 헤드, 암으로 나뉜다.
플래터는 원판형 판으로서 자성체로 디지털 정보가 기록되며,
디스크 헤드는 플래터 위를 움직이면서 디지털 정보를 읽거나 디지털 정보를 쓰는 장치이고 암은 디스크 헤드를 원하는 위치로 움직이는 장치이다.

Q. 파일 옵셋, 논리블록 주소와 CHS 물리주소란 무엇인지 그 의미를 설명하고, 주소 변환에 대해서도 설명하라.

파일 옵셋은 읽거나 쓰려하는 데이터가 파일내 몇 번째 바이트인지 나타내는 값이다.
논리 블록주소는 디스크 상의 모든 블록들을 일차원 배열로 펼쳐 블록 단위로 번호를 붙인 주소이며, OS는 다양한 종류의 하드디스크를 다루어야 하는 운영체제를 위한 것이다.
CHS 물리주소는 섹터 단위로 입출력을 수행하는 디스크 장치의 물리주소로서 각각 실린더, 헤드, 섹터 번호 값으로 이루어져있다. 이와 같은 주소는 파일 내 바이트 주소 → 디스크 장치 내 논리블록 주소 → CHS 물리 주소 순으로 변환되는데, 파일 내 바이트 주소는 커널 내 버퍼 캐시의 내용을 참조할 때 논리블록 주소로 변환되며 논리 블록 주소는 디스크 장치에 입출력이 진행될 때 CHS 물리주소로 변환된다.

Q. 운영체제는 CHS 물리주소를 사용하지 않고 논리 블록 주소만 사용한다. 이렇게 함으로써 어떤 장점이 있는가?

운영체제는 논리블록 주소를 사용함으로서 사용자나 응용프로그램이 저장 매체의 하드웨어에 대한 지식이 없이도 파일 입출력을 할 수 있도록 돕는 장치로부터 분리시키는 역할을 한다. 이 특성은 또한 운영체제가 저장 장치의 종류나 실린더 수, 헤드 수, 트랙 당 섹터 수 등 디스크 장치의 하드웨어와 무관하게 개발되고 구현될 수 있도록 한다.

4. fread() 함수의 호출에서 시작하여 파일 블록이 읽혀오는 과정을 단계별로 간단히 나열하면서 설명하라.

① fread()를 호출하여 일정 바이트를 파일로부터 배열 buf[]로 읽어 들이도록 지시한다.

② C 라이브러리 버퍼에 데이터가 있으면 buf[]로 복사하고 종료한다. 그런데 만약 버퍼에 데이터가 없다면 시스템 호출로 파일 데이터 읽기를 지시한다.

③ 커널 파일 입력 함수를 실행한다. 파일 데이터 위치를 디스크 논리 블록 번호로 전환하고 커널 버퍼 캐시에 논리 블록이 존재하는지 확인한다. 만약 논리블록이 있다면 C 라이브러리 버퍼로 복사 후 리턴한다. 그러나 만약 논리블록이 없다면 디스크 장치 드라이버에게 디스크 장치로부터 논리블록을 읽어 들이도록 지시 한 후 컨텍스트 스위칭을 한다.

④ 디스크 장치 내 프로세서가 해당 논리블록이 디스크 캐시에 있는지 확인한다. 논리블록이 존재한다면 현재 과정을 건너뛴다. 그러나 만약 논리블록이 없다면 논리블록주소를 CHS 물리주소로 변환 후 디스크 암을 이동시켜 디스크 헤더로 섹터들을 읽어 디스크 캐시에 저장한다.

⑤ 디스크캐시에 저장한 후에 논리블록을 DMA 제어기가 커널 버퍼 캐시로 복사한다.

⑥ 인터럽트 신호를 보내어 인터럽트 서비스루틴을 실행하고, 해당 프로세스를 깨워 준비큐에 삽입한다.

⑦ 프로세스 실행 후 커널 버퍼 캐시에서 C 라이브러리 버퍼로 파일 블록 복사 후 사용자 모드로 전환한다.

⑧ C 라이브러리 버퍼에서 buf[]로 파일 데이터 복사후 응용프로그램으로 리턴한다.

5. 파일 시스템 메타 정보와 파일 메타 정보에는 어떤 것들이 있는지 나열하고, FAT 파일 시스템과 Unix 파일 시스템에서 이들이 각각 어디에 저장되어 있는지 설명하라.

파일 시스템 메타 정보로 여러 운영체제에 공통적으로 존재하는 것은 아래와 같다.

- 저장 매체에서 파일 시스템이 차지하는 전체 크기
- 저장 매체 속에 현재 파일이 저장되어 사용중인 크기
- 저장 매체에 비어있는 크기와 비어있는 블록 리스트

파일 메타 정보로 여러 운영체제에 공통적으로 존재하는 것은 아래와 같다.

- 파일 이름
- 파일 크기
- 파일이 만들어진 시간
- 파일이 수정된 시간
- 파일이 가장 최근에 액세스된 시간
- 파일을 만든 사용자 (소유자)
- 파일 속성 (접근권한)
- 파일이 저장된 매체 속의 위치

FAT 파일 시스템에서 메타 데이터는 루트 디렉터리 0, 1번의 특별히 예약된 항목에, 파일 메타 데이터는 각 디렉터리 항목에 저장되며, Unix 파일 시스템에선 파일 시스템 메타 데이터는 슈퍼 블록에, 파일 메타 데이터는 각 i-node에 저장된다.

6. /etc/code/os.c 파일의 i-node를 찾는 과정을 단계별로 간단히 글로 나열하면서 설명하라.

① 루트 디렉터리 (/)의 i-node 번호를 슈퍼 블록에서 알아낸다

② 루트 디렉터리의 i-node로 가서 알아낸 루트 디렉토리가 저장된 블록으로 가서 etc의 i-node 번호를 알아낸다.

③ /etc의 i-node로 가서 알아낸 /etc가 저장된 블록으로 가서 code의 i-node 번호를 알아낸다.

④ /etc/code의 i-node로 가서 알아낸 /etc/code가 저장된 블록으로 가서 os.c의 i-node 번호를 알아낸다.

⑤ /etc/code/os.c의 i-node로 가서 알아낸 /etc/code/os.c가 저장된 블록을 읽는다. 최종적으로 읽은 이 블록이 바로 os.c 파일이다.

7. File Allocation Table을 이용하여 파일 블록들의 위치를 정보를 얻는 방법과, i-node를 이용하여 파일 블록의 위치를 정보를 얻는 방법에 대해 각각, 한 파일의 모든 블록을 찾아 내는 과정을 간단히 설명하라.

FAT 방식: 루트 디렉터리로 접근하여 찾는 파일의 정보 중 블록 번호를 읽어 엑세스한다. 이와 같은 실행을 반복적으로 수행하며 최종적으로 파일을 찾으면, (파일의 크기 / 블록 크기) 개 만큼 할당된 블록을 블록 안의 다음 블록 번호를 읽어 차례로 엑세스한다. 마지막 블록의 경우 다음 블록 번호가 -1로 표시된다.

UFS의 i-node 방식: 슈퍼 블록에서 루트 디렉토리의 i-node 값을 알아낸 후, 반복적으로 i-node에 저장된 블록번호와 해당 블록 안에 있는 i-node 번호를 알아내어 단계적으로 엑세스한다.

8. i-node를 사용하여 파일 블록들의 위치를 저장할 때, 블록의 크기가 4KB인 경우, 하나의 파일의 최대 크기는 얼마까지 가능한가?

12개의 직접 인덱스 12, 1개의 간접 인덱스 1024, 1개의 2중 간접 인덱스 1024×1024, 1개의 3중 간접 인덱스 1024×1024×1024 이므로

(12 + 1024 + 1024×1024 + 1024×1024×1024) × 4KB = 총 48KB + 4MB + 4GB + 4TB 이다

9. FAT32의 경우, FAT 테이블의 한 항목의 크기가 32비트이고 블록의 크기가 4kB인 경우, 파일 시스템의 최대 크기는 얼마인가?

테이블 한 항목의 크기가 32비트라면, 이를 통해 접근가능한 블록의 수는 $2^{32}-2$ = 약 2^{32}개이다. 그리고 여기에 블록의 크기가 4kB이라면 파일 시스템이 저장할 수 있는 데이터의 최대한은 $2^{32} \times 4kB = 2^{32} \times 2^{12}$ byte

$= 2^{44}$ byte = 16TB이다.

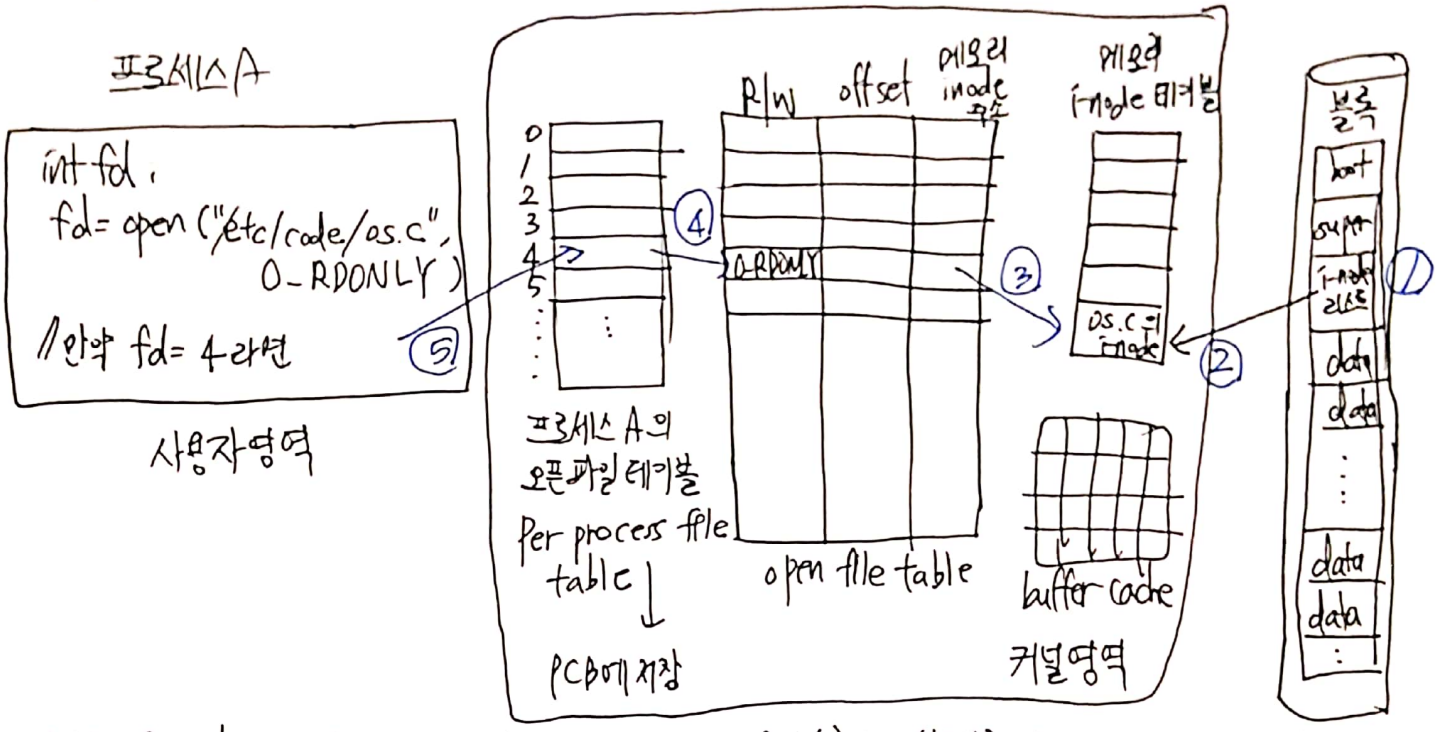
10. 운영체제 커널 내에 있는 버퍼 캐시는 여러 면에서 좋은 영향을 나쁜 면이 있다. 이들을 각각 설명하라.

버퍼 캐시의 장점으로 읽기의 경우 디스크 장치가 아닌 버퍼 캐시에서 읽도록 하고, 쓰기의 경우 번번한 수정과정을 디스크 장치가 아닌 버퍼 캐시에 저장함으로서 디스크 입출력 시간을 매우 줄일 수 있다는 점이 있다. 하지만 버퍼 캐시의 변경사항이 디스크에 저장되지 못한 상태에서 컴퓨터 전원이 꺼지거나 프로세스가 비정상적으로 종료되면 저장되지 않은 파일의 저장사항이 모두 사라지게 되어버리는 단점이 있다.

11. 파일 디스크립터란 어떤 것인가? 정수인가 주소인가? 파일 디스크립터 0, 1, 2의 용도는 무엇인가?

파일 디스크립터는 정수로, 이 같은 프로세스별 오픈 파일 테이블의 항목 번호를 의미한다. 또한 파일 디스크립터의 0, 1, 2 번은 각기 표준 입력, 표준 출력, 표준 오류를 가리키는 특별한 목적성을 지니며 이미 열린 상태로 프로세스가 실행된다. 표준 입력은 키보드, 표준 출력과 오류는 스크린이며, 이는 장치들이 운영체제에서 파일로 다루어지기 때문이다. 이 세 값을 이용하여 키보드에서 읽거나 스크린에 출력할 수 있다.

12. int fd = open("/etc/code/os.c", O_RDONLY) 이 실행되는 과정을 커널 내 자료 구조들의 변화와 함께 번호를 매겨가면서 그려라.
open()은 어떤 경우에 실패하는가?



메모리 i-node 테이블에 빈 공간이 없으면 open()은 실패하여 시스템호출은 오류를 리턴하게 된다.

13. (a)으로 열었을 때, write(fd, buf, 100)이 실행될때 write() 시스템 호출이 실패한다. write() 하는 과정 중 어떤 과정에서 실패를 판정하게 되는가?

스텝에서는 파일을 읽기 전용모드(O_RDONLY)로 열었기 때문에 열기모드가 O_RDWR 모드인지 확인하는 과정에서 파일 쓰기는 실패하게된다.

14. 유닉스 파일 시스템에서 슈퍼블록과 열린 파일의 i-node가 커널의 메모리에 적재되어야 하는 이유는 무엇인가?

커널은 파일이 생성될 때마다 비어있는 자유 I-node를 찾아야한다. 그리고 파일이 삭제될 때마다 사용된 I-node를 반환하기 위해 슈퍼블록을 읽고 써야하는 작업이 발생한다. 따라서 커널 코드의 실행을 빨리 하기위해 슈퍼블록과 열린 파일의 I-node는 메모리에 로딩되어 있어야한다. 그리고 주기적으로 메모리에서 운영되는 슈퍼블록은 디스크에 기록되어야 한다.