13장. 입/출력 시스템 (I/O System)

순천향대학교 컴퓨터공학과
이 상 정

강의 목표 및 내용

- 목표
  - 운영체제의 입/출력 서브시스템 구조
  - 입/출력 하드웨어의 원리와 복잡도
  - 입/출력 하드웨어와 소프트웨어의 성능

- 내용
  - 입/출력 하드웨어
  - 응용 입/출력 인터페이스
  - 커널 입/출력 서브시스템
  - 스트림
  - 성능

입/출력 하드웨어 (I/O Hardware)

- 다양하고 많은 종류 입출력 장치 (I/O device)
- 표준적인 개념
  - 포트(port)
    - 컴퓨터와 접속 연결점
  - 버스(bus)
    - 처리기-메모리 서브시스템을 고속 장치에 연결하는 PCI 버스(일반적인 PC system bus)
    - 키보드와 직렬, 병렬 포트처럼 상대적으로 느린 장치들을 연결하는 확장 버스(expansion bus)
  - 제어기(controller)
    - 포트나, 버스나, 입/출력 장치를 제어
- 입출력 명령으로 장치 제어
  - 특별한 입출력 명령 (direct I/O instruction)
  - 메모리 사상 입출력 (memory-mapped I/O)

PC 버스 구조

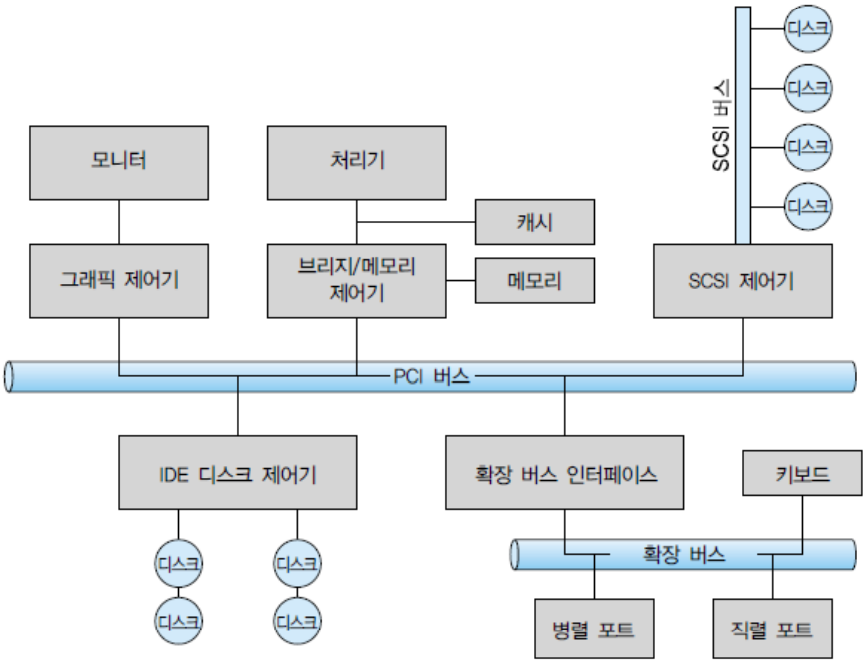


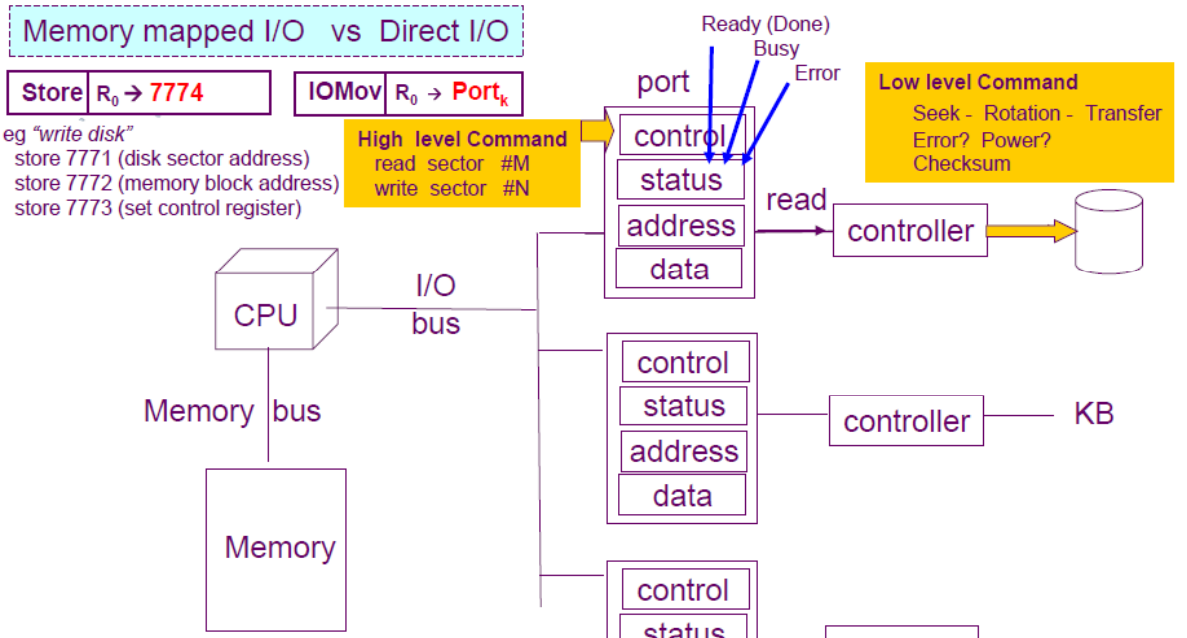
그림 13.1 전형적인 PC 버스 구조

PC 입/출력 포트 위치

입/출력 주소 범위(16진수)	장 치
000 – 00F	DMA 제어기
020 – 021	인터럽트 제어기
040 – 043	타이머
200 – 20F	게임 제어기
2F8 – 2FF	직렬 포트(2차)
320 – 32F	하드 디스크 제어기
378 – 37F	병렬 포트
3D0 – 3DF	그래픽 제어기
3F0 – 3F7	디스켓 드라이브 제어기
3F8 – 3FF	직렬 포트(주)

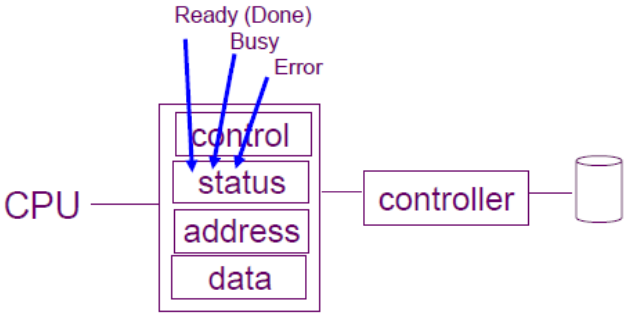
그림 13.2 PC에서 장치 입/출력 포트의 위치(일부)

포트-제어기-레지스터



폴링 (Polling)

- CPU는 입출력을 위해 장치 상태를 검사
  - 명령 준비 완료(command-ready)
  - 사용 중 (busy)
  - 에러

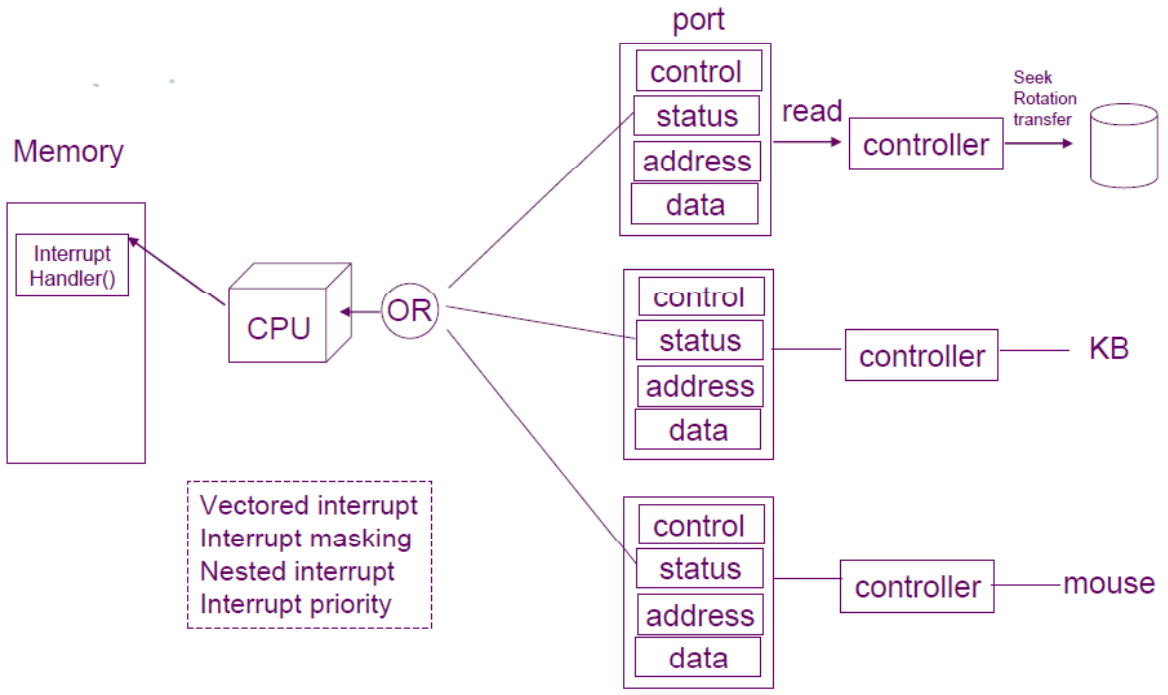


- 폴링(polling)
  - CPU의 입출력 동작 시 장치가 busy이면 바쁜 대기(busy-waiting)
  - 루프를 돌며 장치가 사용 가능할 때까지 검사를 반복

인터럽트 (Interrupt)

- 입/출력 장치가 CPU 인터럽트 요청 라인(interrupt request line)에 요청 신호
- 인터럽트 핸들러(interrupt handler) 수행
- 일부 인터럽트 요청은 지연처리 또는 무시
  - 마스크 가능 인터럽트(maskable interrupt)
  - 우선순위(priority) 기반 동작
  - 일부 인터럽트는 마스크 불가 인터럽트(nonmaskable interrupt)
- 인터럽트 벡터(interrupt vector)
  - 특정 인터럽트 핸들링 루틴을 선택하기 위해 사용되는 주소(번호)
- 운영체제는 인터럽트를 사용하여 여러 가지 예외(exception)도 처리

인터럽트 요청



인터럽트 구동식 입/출력 사이클

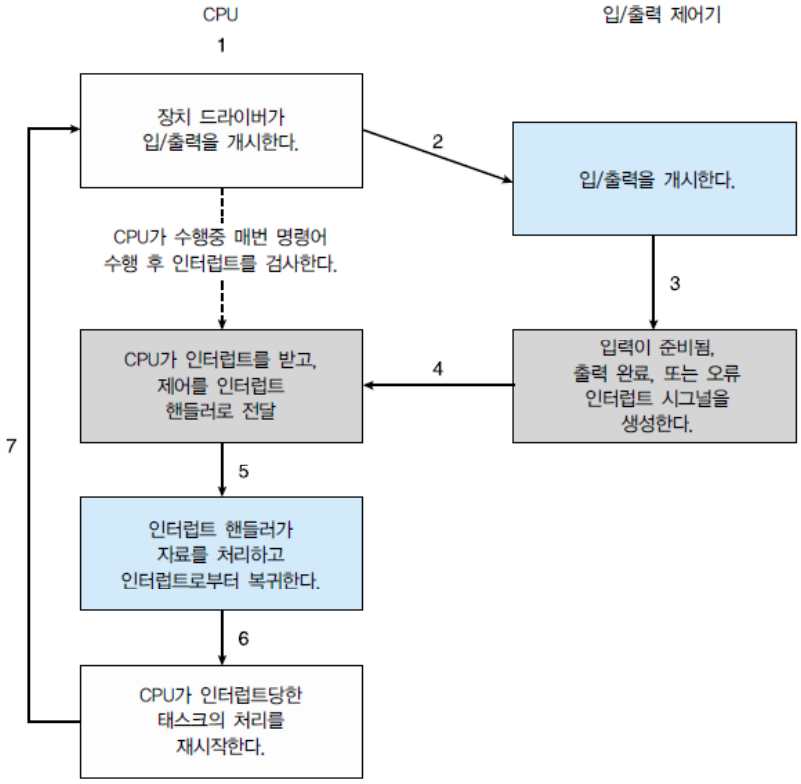


그림 13.3 인터럽트 구동식 입/출력 사이클

Intel Pentium 처리기 사건 벡터 테이블

벡터 숫자	설 명
0	나누기 에러
1	디버그 예외
2	null 인터럽트
3	중단 지점(breakpoint)
4	INTO-탐지 오버플로우
5	경계 범위 예외
6	불법 명령
7	장치가 이용 가능하지 않음
8	이중 폴트
9	협동 처리기 세그먼트 침범(유보됨)
10	불법 작업 상태 세그먼트
11	세그먼트가 존재하지 않음
12	스택 폴트
13	일반적 보호
14	페이지 폴트
15	(Intel이 유보, 사용되지 않음)
16	부동점 에러
17	위치 정렬(alignment) 체크
18	기계 체크
19-31	(Intel이 유보, 사용되지 않음)
32-255	마스크 가능한 인터럽트

그림 13.4 Intel Pentium 처리기 사건 벡터 테이블

직접 메모리 접근 (DMA, Direct Memory Access)

- 대용량 데이터 전송을 위해 PIO(programmed I/O) 대신 DMA 사용
  - PIO
    - CPU가 상태 비트를 반복적으로 검사 ₩하면서 1 바이트씩 옮기는 입출력 방식
- DMA 제어기(DMA Controller)
  - CPU의 도움 없이도 자신이 직접 버스를 통해 DMA 명령 블록을 접근하여 입/출력을 수행

DMA 전송의 단계

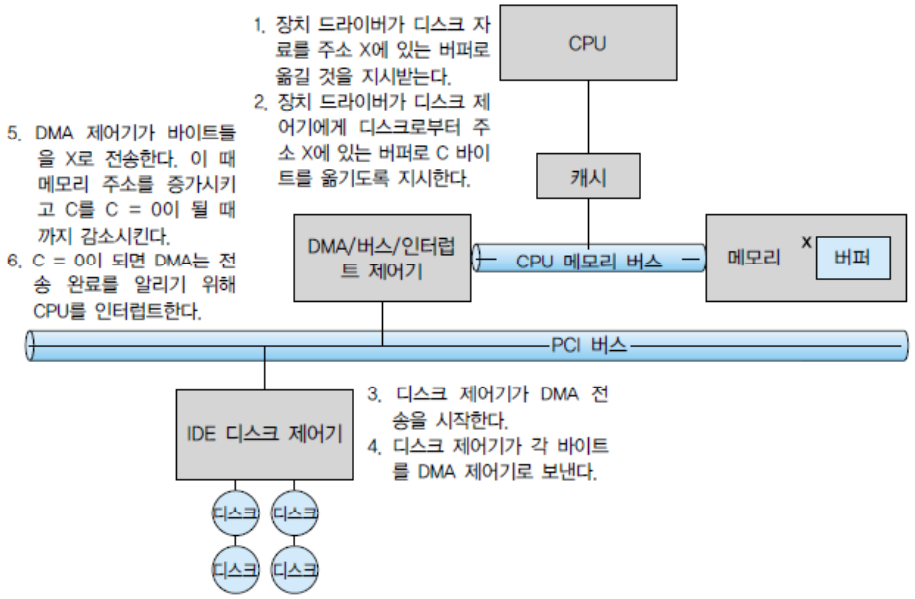


그림 13.5 DMA 전송의 단계

응용 입/출력 인터페이스 (Application I/O Interface)

- 입/출력 시스템 호출(system call)을 사용하여 다양한 입/출력 장치들을 추상화
  - 입/출력 장치들을 접근하기 위해 필요한 표준 함수(standard function)들을 정의, 예를 들면 read( ), write( )
    - 이러한 표준 함수들의 집합을 인터페이스라고 함
- 장치 드라이버(device driver)
  - 인터페이스의 표준 함수들을 수행하여 상위의 커널 입/출력 서브 시스템에 제공
- 다양한 장치의 특성
  - 문자 스트림과 블록(character stream or block)
  - 순차 접근과 임의 접근(sequential or random access)
  - 동기식과 비동기식(synchronous or asynchronous)
  - 공유와 전용(Sharable or dedicated)
  - 읽기/쓰기, 읽기 전용, 쓰기 전용(Read-write, read only, write only)

커널의 입/출력 구조

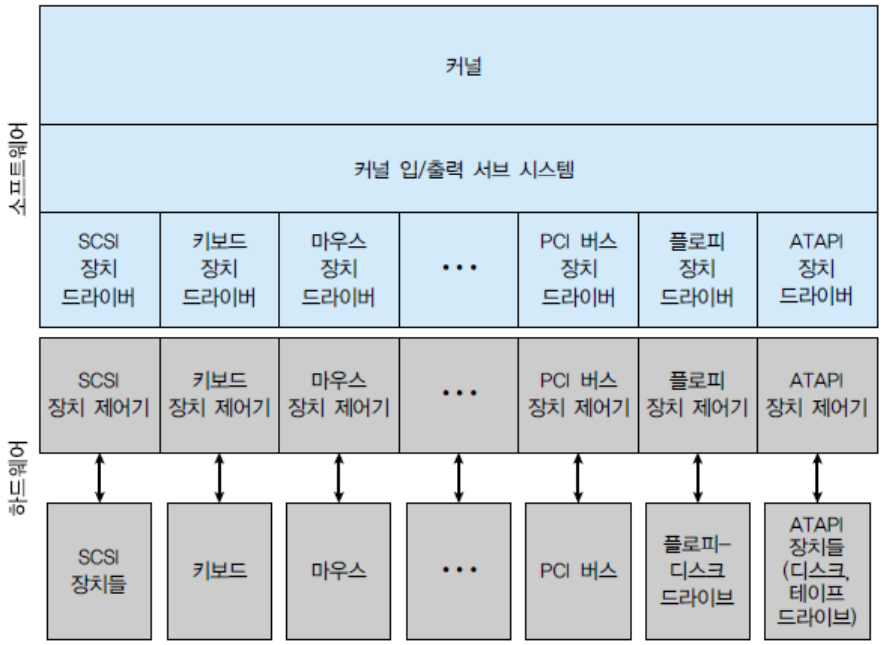


그림 13.6 커널의 입/출력 구조

입출력 장치의 특성

특 성	종 류	예
자료 전송 모드	문자 블록	터미널 디스크
접근 방법	순차 임의	모뎀 CD-ROM
전송 스케줄	동기식 비동기식	테이프 키보드
공유	전용 공유 가능	테이프 키보드
장치 속도	지연 탐색 시간 전송률 연산간의 지연	
입/출력 방향	읽기 전용 쓰기 전용 읽기-쓰기	CD-ROM 그래픽 제어기 디스크

그림 13.7 입/출력 장치의 특성

블록 장치와 문자 장치 (Block and Character Device)

- 블록 장치는 디스크와 같은 장치
  - 블록 단위 입출력
  - Read, write, seek 명령
  - 비가공 입/출력(raw I/O) 또는 파일 시스템 접근
  - 메모리 사상(memory mapped) 파일 접근
  - 임의 접근(random access) 허용
- 문자 장치는 키보드,마우스, 직렬포트와 같은 장치
  - 한 문자 단위 입출력
  - Get, put 명령
  - 순차 접근(sequential access)

네트워크 장치 (Network Device)

- UNIX나 Windows NT를 포함한 많은 운영체제에서 사용하는 인터페이스는 네트워크 소켓(socket) 인터페이스
  - 네트워크 연산과 네트워크 프로토콜을 분리
  - 일련의 소켓들을 관리하는 select( ) 함수를 제공
    - 수신 대기중인 패킷을 가지고 있는 소켓, 송신 패킷 여유 공간을 가지고 있는 소켓 등에 대한 정보 제공
    - 응용 프로그램이 네트워크 상태를 폴링할 필요가 없어서 네트워크 입/출력을 위한 바쁜 대기(busy-wait) 제거

클록과 타이머 (Clock and Timer)

- 하드웨어 클록과 타이머의 기본적인 기능
  - 현재 시간을 제공
  - 경과된 시간을 제공
  - T 시각이 되면 X 오퍼레이션을 실행

- 프로그래머블 인터벌 타이머(programmable interval timer)
  - 한 번 또는 주기적으로 인터럽트를 발생하도록 반복해서 하도록 설정
  - 많은 컴퓨터에서는 하드웨어 클록의 틱(tick)에 의해 생성되는 인터럽트율은 초 당 18에서 60틱(tick)

봉쇄형과 비봉쇄형 입/출력 (Blocking and Nonblocking I/O)

- 봉쇄형(blocking) 입/출력
  - 입/출력이 즉시 완료될 수 없을 경우 응용 프로그램은 봉쇄(block)
    - 실행 큐로부터 대기 큐로 이동
  - 봉쇄형 코드가 비봉쇄형 코드보다 이해하기 쉬움
- 비봉쇄형(nonblocking) 입/출력
  - 비봉쇄형 입/출력 호출은 바로 리턴
    - 전송된 바이트 값(전체 또는 일부)을 바로 리턴
  - 다중 쓰레드(multi-thread) 방식으로 프로그램을 작성
- 비동기식(asynchronous) 입/출력
  - 비봉쇄형과 같이 바로 리턴
  - 입/출력이 완료되면 입/출력이 완료됐다는 사실을 알림
    - 응용 프로그램의 변수를 셋팅 또는 신호를 전송
    - 소프트웨어 인터럽트를 걸거나 또는 응용과는별도로 수행되는 콜 백 루틴을 수행하여 알려줌

동기식과 비동기식 입출력

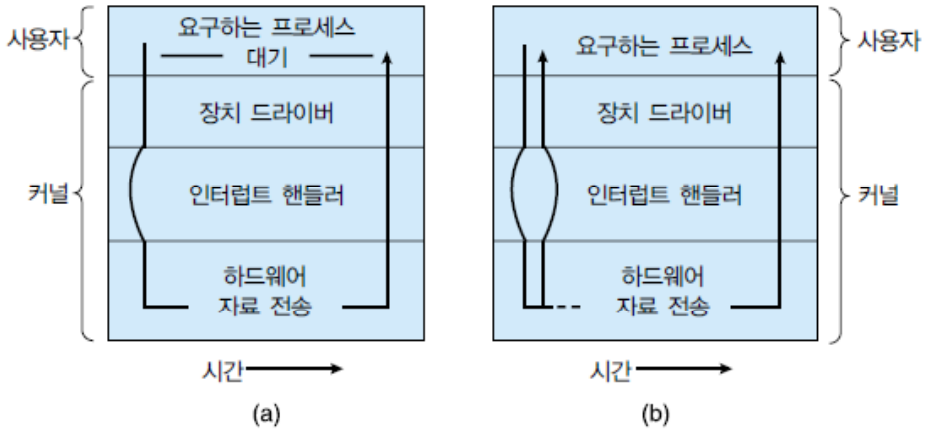


그림 13.8 두 입/출력 방법: (a) 동기적, (b) 비동기적

커널 입/출력 서브시스템 (Kernel I/O Subsystem)

❑ 입/출력 스케줄링(I/O Scheduling)

- 각각의 장치마다 대기 큐를 유지함으로써 스케줄링을 구현
- 운영체제는 통상은 공평(fairness)하게 일을 처리하려고 시도
- 비동기 입/출력 시 커널은 동시에 많은 입/출력 요청의 추적 필요
  - 운영체제는 각 장치 상태 테이블(device-status table)에 대기 큐를 연동

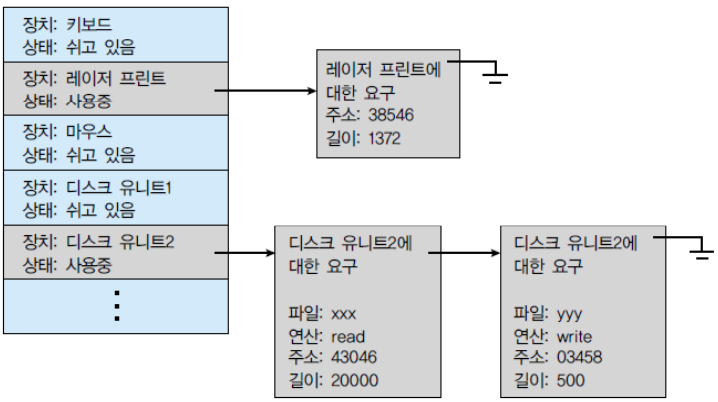
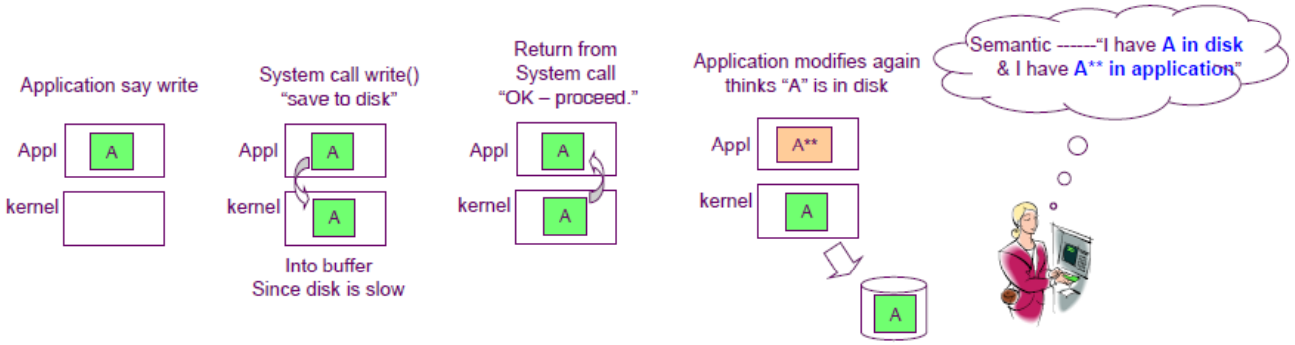


그림 13.9 장치 상태 테이블

버퍼링 (Buffering)

- 버퍼는 입/출력 장치와 응용 프로그램 사이에 자료가 전송되는 동안 그 자료를 임시로 저장하는 메모리 영역
- 버퍼링을 하는 이유
  - 자료의 생산자와 소비자 사이에 속도 차이를 대처
  - 서로 다른 장치들 사이에 사용되는 자료 전송 크기 차이를 극복
  - 응용 프로그램의 입/출력 복제 의미(copy semantic)를 지원



캐싱 (Caching)

- 캐시는 자주 사용될 자료의 복사본을 저장하는 빠른 메모리 영역
  - 버퍼는 그 자료를 가지고 있는 유일한 장소인 반면, 캐시는 자료의 복사본이고 보다 빠른 저장소에 추가로 저장한다는 점에서 차이
- 버퍼 캐시
  - 메모리의 한 영역이 두 가지 목적(버퍼, 캐시) 모두를 위해 사용
  - 예
    디스크 입/출력 스케줄링을 효율적으로 수행하기 위한 버퍼
    => 읽기/쓰기가 빈번하거나, 또는 프로세스들 간에 공유되어야 하는 파일들을 위한 캐시로도 사용

스풀링 및 장치 예약 (Spooling and Device Reservation)

- 스풀(spool)은 교차(interleave)해서 동작될 수 없는 프린터 같은 장치를 위해 출력 자료를 보관하는 버퍼
  - 프린터는 한 번에 하나의 작업만을 처리하고, 여러 응용 프로그램의 출력을 섞어서 번갈아 가며 출력시킬 수는 없음
  - 많은 응용 프로그램들은 동시에 프린터 출력 자료 생성
    - 운영체제는 응용 프로그램으로부터 프린터로 가는 모든 출력을 가로챔
    - 각 응용 프로그램의 출력은 각각 대응되는 디스크 파일에 저장(스풀)

- 장치 예약(device reservation)
  - 한 프로그램만이 장치를 독점적으로 사용하도록 허용
  - 장치를 할당(allocation), 반납(deallocation)하는 시스템 호출
  - 교착상태(deadlock) 주의

에러 처리(Error Handling)와 입/출력 보호(I/O Protection)

- 에러 처리
  - 운영체제는 일시적인 고장(디스크 읽기 실패, 네트워크 전송 오류 등)으로부터 효과적으로 극복(recover)을 시도
  - 입출력 요청 실패 시 에러 번호나 코드 리턴
  - 오류 로그(error-log) 기록 보관

- 입/출력 보호
  - 사용자 프로세스는 고의적이든 아니든 불법적인 입/출력 명령을 시도함으로써 정상적인 동작을 방해할 수 있음
    - 모든 입/출력 명령은 특권 명령(privileged instruction)으로 정의
    - 사용자는 입/출력 명령을 직접 수행할 수 없고, 운영체제가 입/출력을 대신 수행하도록 시스템 호출을 수행
    - 메모리 사상 메모리 또는 입/출력 포트 메모리의 위치는 메모리 보호 시스템에 의해 사용자로부터 보호

입/출력을 위한 시스템 호출의 사용

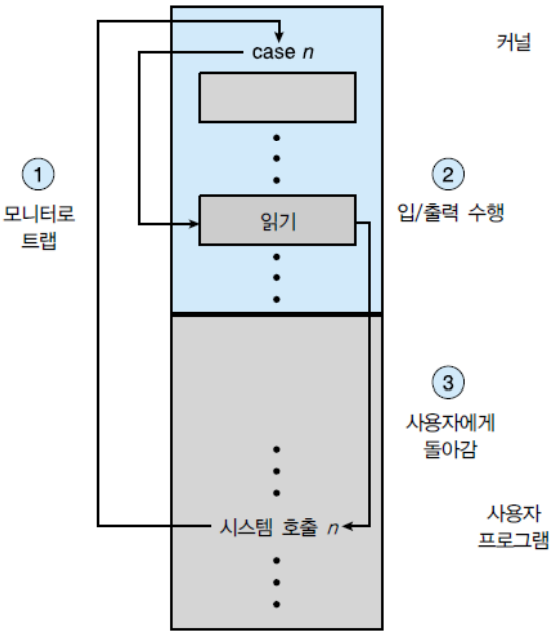


그림 13.11 입/출력을 위한 시스템 호출의 사용

커널 자료 구조 (Kernel Data Structure)

- 커널은 입/출력 구성 요소에 대한 상태 정보를 유지
  - 오픈 파일 테이블, 네트워크 연결, 문자장치 통신 상태

- UNIX는 파일 시스템 인터페이스를 사용하여 다양한 개체에 접근

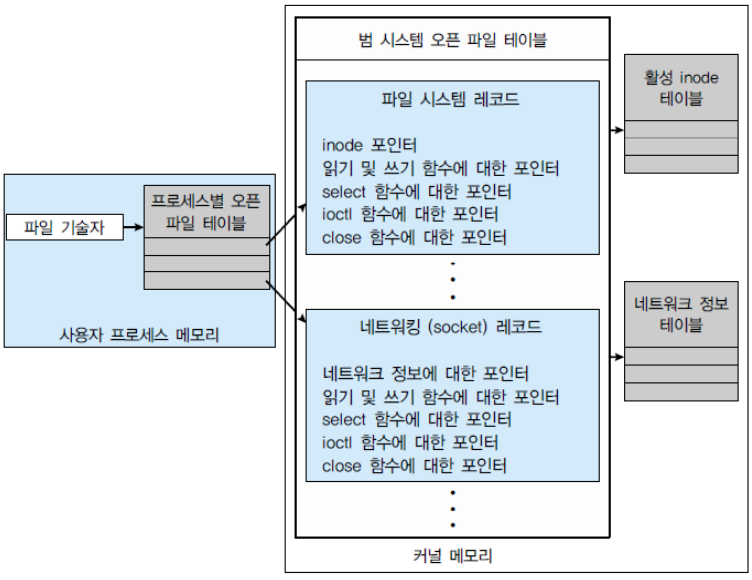


그림 13.12 UNIX 입/출력 커널 구조

커널 입/출력 서브시스템 요약 (Kernel I/O Subsystem Summary)

- 입/출력 서브시스템은 다음 사항들을 관리
  - 파일 및 장치의 이름 관리
  - 파일 및 장치에 대한 접근 제어
  - 작업 제어(예를 들어, 모뎀은 seek()를 할 수 없음)
  - 파일 시스템을 위한 공간 할당
  - 장치(device) 할당
  - 버퍼링, 캐싱 및 스풀링
  - 입/출력 스케줄링
  - 장치 상태 모니터링, 에러 처리 및 고장 복구
  - 장치 드라이버 구성(configuration) 및 초기화
- 입/출력 서브시스템의 상위 계층은 장치 드라이버가 제공하는 공통적인 인터페이스를 통해 장치에 접근

입/출력 요청을 하드웨어 연산으로 변환

- 파일을 디스크에서 읽는 경우를 고려
  - 응용 프로그램이 파일 이름으로 자료를 참조
  - 운영체제가 디스크 내에서 이 파일이 위치한 곳 찾음
  - 파일 이름을 장치 내부 표현(하드웨어 포트 주소, 제어기 레지스터)로 변환
  - 디스크의 데이터를 커널 버퍼로 읽음
  - 데이터를 복사(커널 버퍼 -> 사용자 버퍼)
  - 프로세스로 리턴

입/출력 요청의 전 사이클

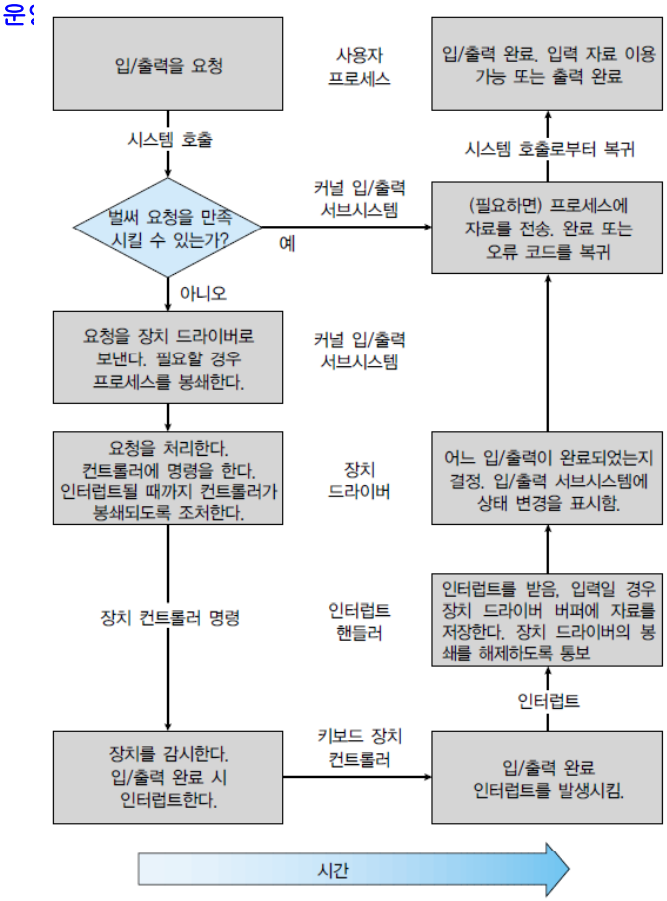


그림 13.13 입/출력 요청의 전 사이클

스트림 (STREAM)

- 스트림은 장치 드라이버와 사용자 수준 프로세스 사이의 완전 양방향 연결 (full-duplex communication channel)
  - UNIX 시스템 V
- 스트림 구성
  - 사용자 프로세스와 인터페이스하는 스트림 헤드
  - 장치를 제어하는 드라이버 엔드
  - 위 둘 사이에 존재하는 0 개 이상의 스트림 모듈
- 각 모듈은 읽기와 쓰기 큐, 즉 한 쌍의 큐를 가지고 있음
- 큐 사이에 자료를 전송하는 데 메시지 전달이 사용

스트림 구조

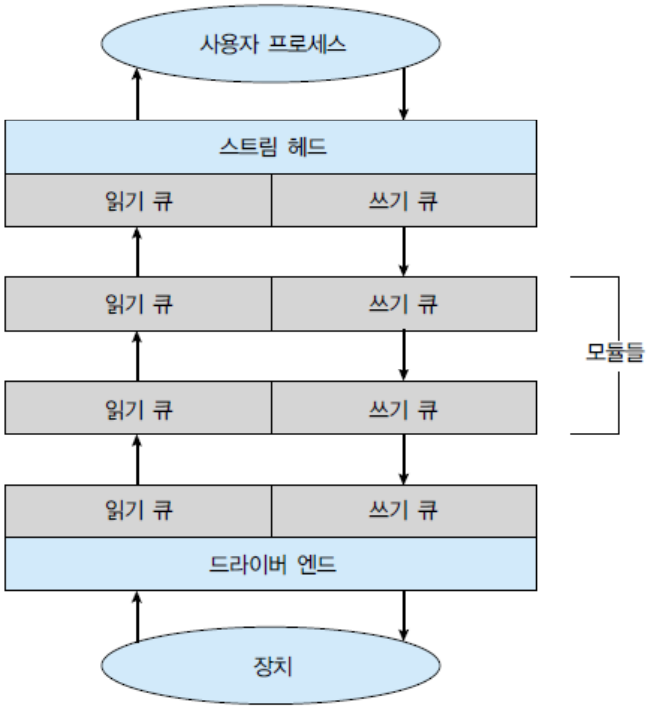


그림 13.14 스트림 구조

성능 (Performance)

- 입/출력은 시스템 성능에 매우 중요한 요소
  - 입/출력은 장치 드라이버 코드(커널 코드)를 실행
  - 인터럽트으로 인한 문맥교환 유발
  - 커널 버퍼와 응용 프로그램 자료 공간 사이에 자료를 복사
  - 네트워크의 트래픽은 성능에 부담

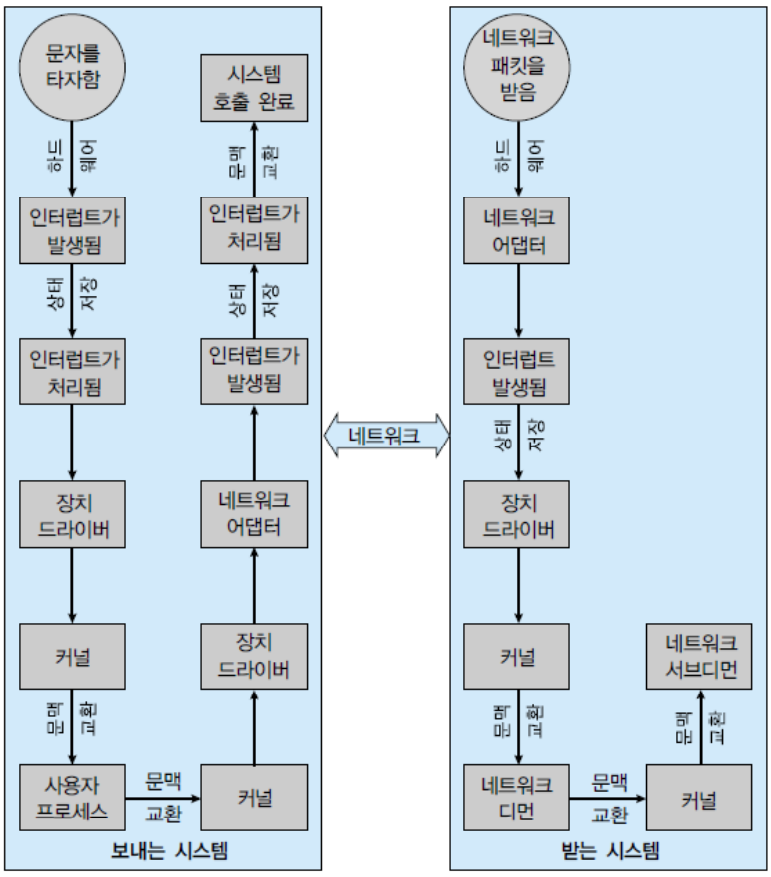


그림 13.15 프로세스간 통신

성능 개선

- 입/출력의 효율을 높이기 위해서는 다음과 같은 여러 가지 원칙들을 적용
  - 문맥 교환의 빈도를 줄임
  - 메모리에서 장치와 응용 프로그램 사이에 자료가 복사되는 횟수를 줄임
  - 대용량 전송, 지능적인 제어기, 폴링 등을 사용하여 인터럽트 빈도를 줄임
  - DMA나 채널 등을 사용
  - CPU, 메모리, 버스, 입/출력 등에 대한 부하가 균일하게 되도록 함

입출력 기능 구현 지점

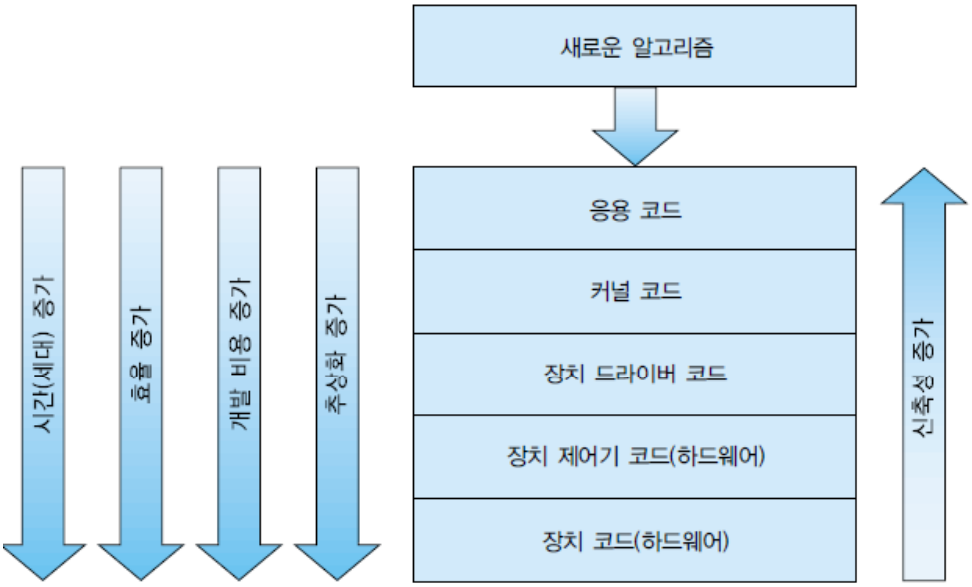


그림 13.16 장치 기능 추이