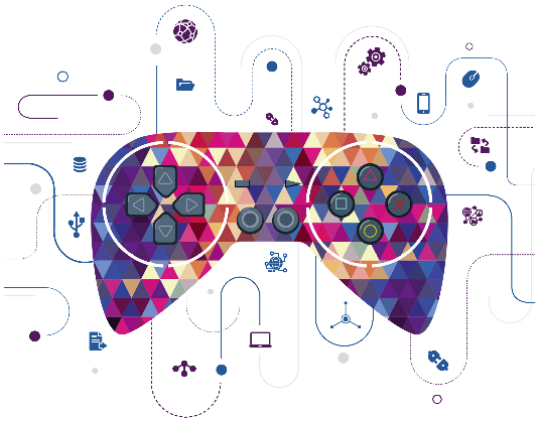


그림으로 배우는 구조와 원리

운영체제 개정 3판

Chatpter 02
운영체제의 소개

- 운영체제의 개념과 발전 목적
- 운영체제의 주요 기능
- 운영체제의 발전 과정에 따른 주요 특징
- 운영체제가 제공하는 주요 서비스
- 운영체제 설계에 필요한 모듈과 계층 구조의 개념

Section 01 운영체제의 개념과 발전 목적

■ 운영체제의 개념

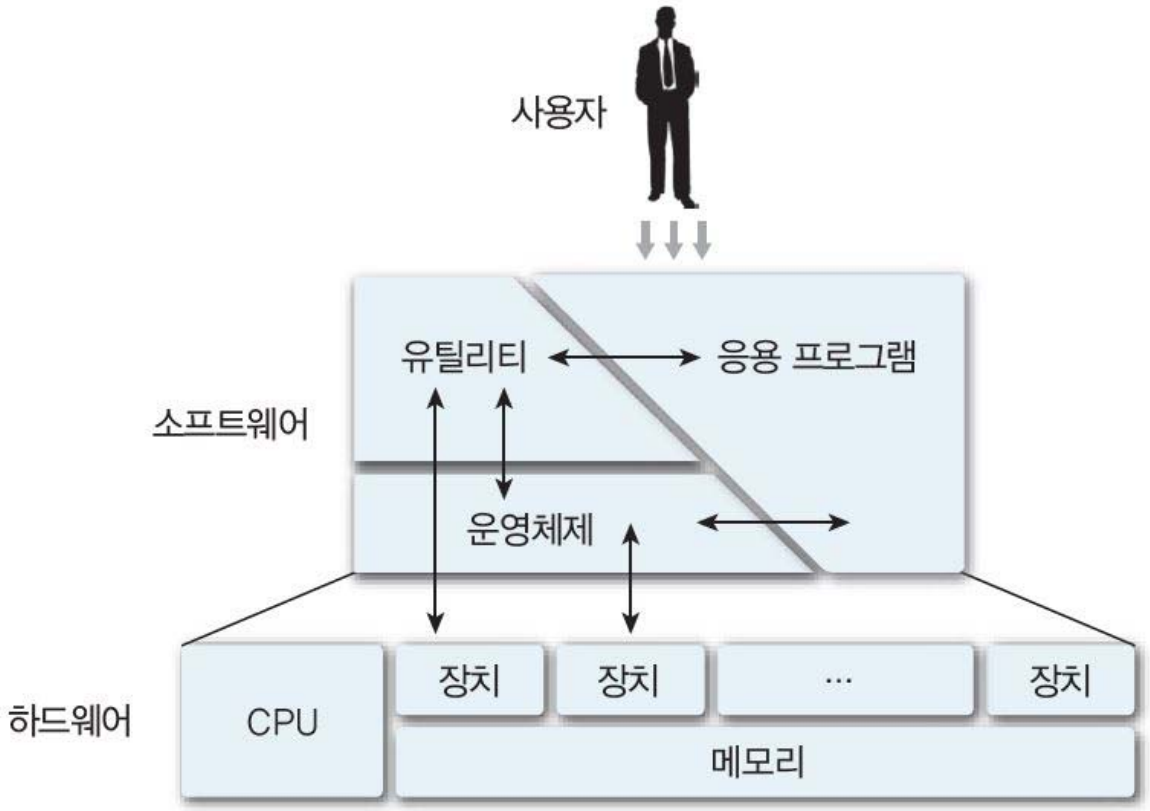


그림 2-1 컴퓨터 시스템의 구성 요소와 운영체제

- 사용자 : 컴퓨터를 사용하는 사람이나 장치, 다른 컴퓨터 등을 의미
- 소프트웨어 : 컴퓨터의 기능 수행에 필요한 모든 프로그램
- 하드웨어 : 기본 연산 자원을 제공하는 프로세서(CPU, 중앙처리장치), 메모리, 주변장치 등

1. 운영체제의 개념

- **컴퓨터 자원관리면에서 운영체제의 정의**

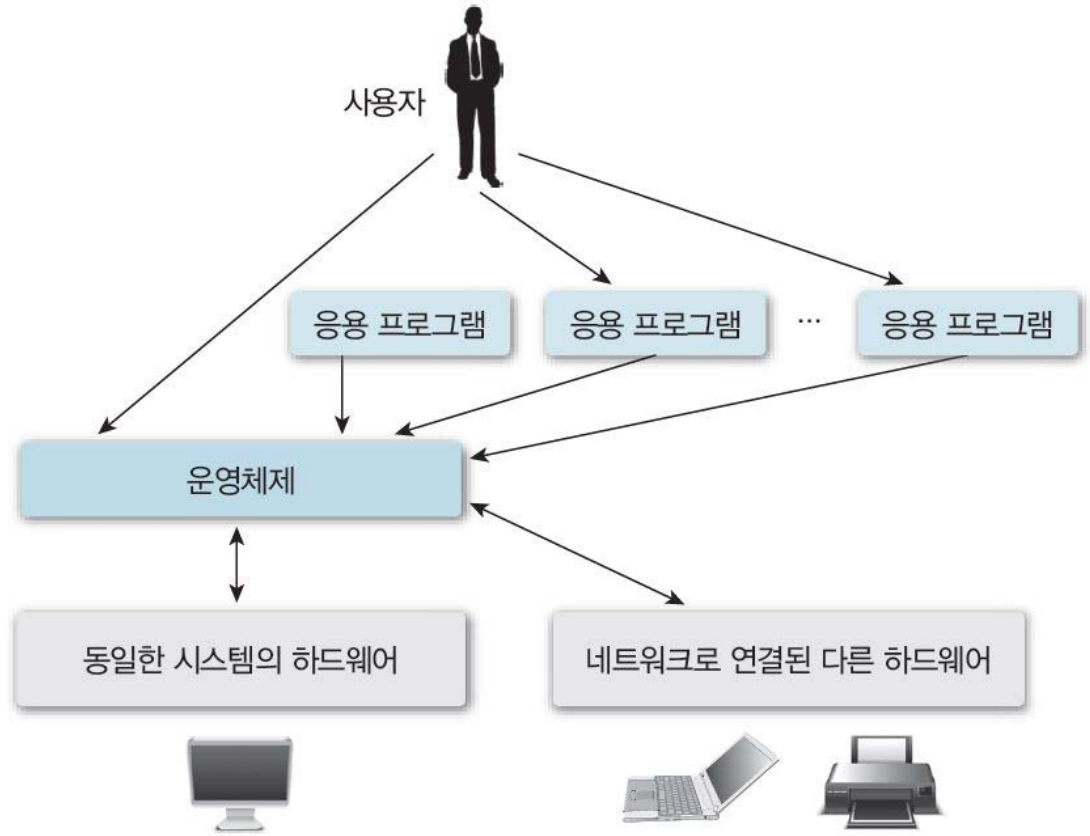


그림 2-2 운영체제의 역할

- 조정자 : 운영 요소 사용을 제어하면서 사용자와 응용 프로그램 간에 통신할 수 있게 함
  작업을 할 수 있는 환경만 제공
- 자원 할당자나 관리자 : 각 응용 프로그램에 필요한 자원 할당, 자원 할당 방법 결정
- 응용 프로그램과 입출력장치 제어자 : 다양한 입출력장치와 응용 프로그램 제어

1. 운영체제의 개념

■ 운영체제의 정의와 역할

▪ 정의

- 사용자와 하드웨어 사이의 중간 매개체로 응용 프로그램의 실행을 제어하고, 자원을 할당 및 관리하며, 입출력 제어 및 데이터 관리와 같은 서비스를 제공하는 소프트웨어

▪ 역할

- 하드웨어 및 사용자, 응용 프로그램, 시스템 프로그램 사이에서 인터페이스를 제공
- 프로세서, 메모리, 입출력장치, 통신장치 등 컴퓨터 자원을 효과적으로 활용하려고 조정·관리
- 메일 전송, 파일 시스템 검사, 서버 작업 등 높은 수준의 서비스를 처리하는 응용 프로그램을 제어
- 다양한 사용자에게서 컴퓨터 시스템을 보호하려고 입출력을 제어하며 데이터를 관리

2. 운영체제 발전의 목적

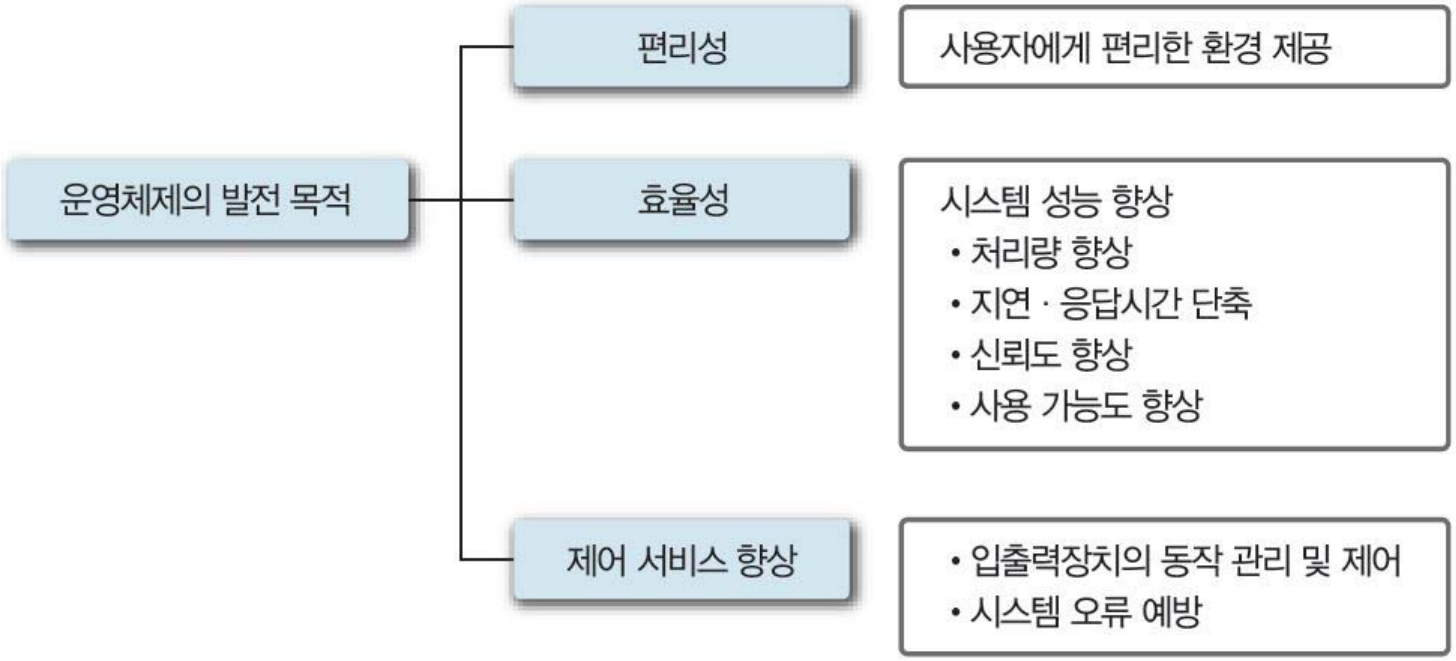


그림 2-3 운영체제의 발전 목적

- **편리성 : 사용자에게 편리한 환경 제공**
  - 프로그램 개발 환경뿐만 아니라 응용 프로그램에 대한 사용자 인터페이스, 즉 사용자와 컴퓨터 시스템이 정보 및 명령을 상호 교환할 수 있는 인터페이스 제공

2. 운영체제 발전의 목적

효율성 : 시스템 성능 향상

- 사용자가 많은 대형 컴퓨터 시스템에서 특히 중요. 운영체제는 각 프로그램을 유기적으로 결합하여 시스템 전체 성능을 향상

- 시스템 성능의 평가 기준
  - 처리량
  - 지연·응답시간(턴 어라운드 타임)
  - 신뢰도
  - 사용 가능도(가동률)

제어 서비스 향상

- 시스템 확장, 효율적 운영을 위해 새로운 기능의 효과적인 개발을 허용하는 방법으로 발전

- 입출력장치의 동작 관리 및 제어, 시스템 오류 예방 등으로 컴퓨터 자원을 여러 사용자에게 효율적으로 할당하고 관리할 수 있도록 제어 서비스를 발전

운영체제의 목표

운영체제의 목표

그림 1-6 운영체제의 역할과 목표

Section 02 운영체제의 기능

■ 운영체제의 역할에 따른 기능

그림 2-4 운영체제의 기능

1. 자원관리

■ 자원

- 컴퓨터 시스템의 메모리, 프로세스, 장치, 파일 등 구성 요소

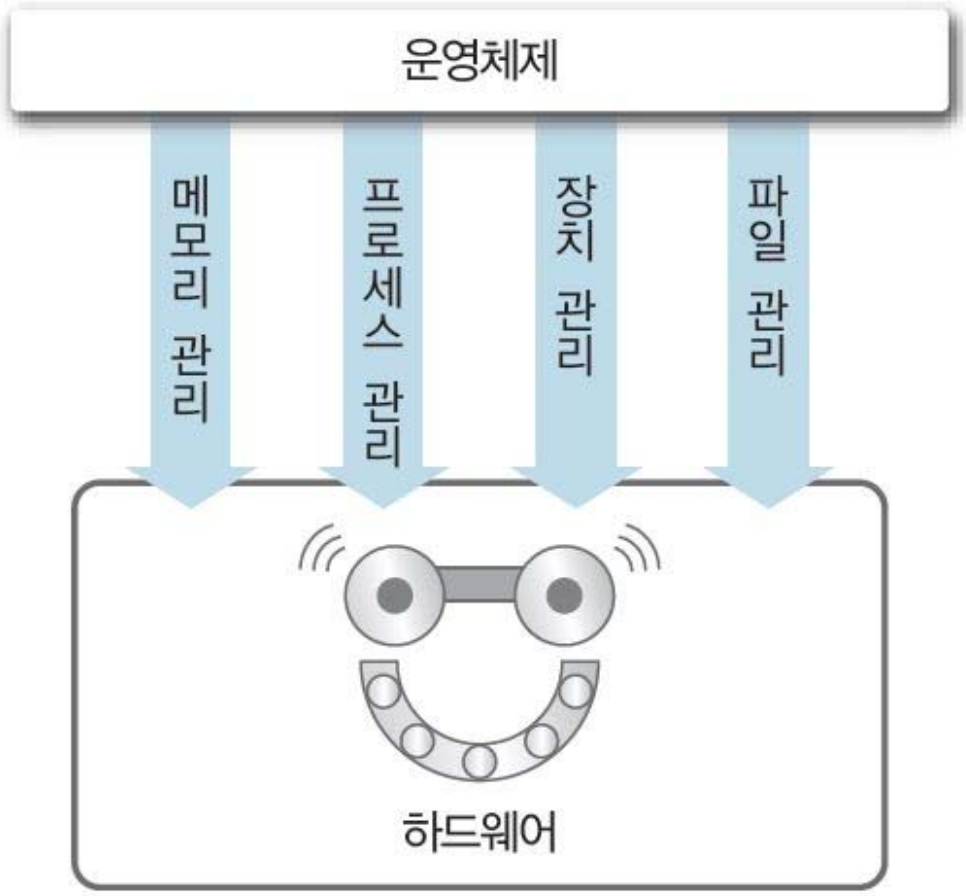


그림 2-5 운영체제의 자원 관리 기능

1. 자원관리

메모리 관리

- 메인 메모리 관리 : 프로세서가 직접 주소로 지정할 수 있는 유일한 메모리

- 메모리 관리의 기능
  - 메모리의 어느 부분을 사용하고, 누가 사용하는지 점검
  - 메모리에 저장할 프로세스 결정
  - 메모리를 할당하고 회수하는 방법 결정

- 보조기억장치 관리 : 메인 메모리는 공간이 제한되어 데이터와 프로그램을 계속 저장할 수 없어 보조기억장치 이용

- 보조기억장치 관리의 기능
  - 빈 여유 공간 관리
  - 새로운 파일 작성 시 저장 장소 할당
  - 메모리 접근 요청 스케줄링
  - 파일 생성하고 삭제

1. 자원관리

프로세스 관리

프로세스

- 하나의 프로세스는 프로세서, 메모리, 파일, 입출력장치와 같은 자원으로 구성
- 자원은 프로세스 생성할 때 제공하거나 실행 중에도 할당 가능

시스템

- 프로세스의 집합
- 시스템 코드 수행하는 운영체제 프로세스와 사용자 코드 수행하는 사용자 프로세스로 구분
- 모든 프로세스는 프로세서 분할 사용하여 병행 수행 가능

프로세스 관리를 위한 운영체제의 기능

- 프로세스와 스레드 스케줄링
- 사용자 프로세스와 시스템 프로세스 생성, 제거
- 프로세스 중지, 재수행
- 프로세스 동기화 방법 제공
- 프로세스 통신 방법 제공
- 교착 상태deadlock를 방지하는 방법 제공

1. 자원관리

■ 주변장치(입출력장치) 관리

- 운영체제는 특수 프로그램인 장치 드라이브를 사용하여 입출력장치와 상호작용

- 장치 드라이버는 특정 하드웨어장치와 통신할 수 있는 인터페이스를 제공하므로 특정 하드웨어에 종속된 프로그램

- 주변장치(입출력장치) 관리를 위한 운영체제의 기능
  - 임시 저장buffer-caching 시스템 기능 제공
  - 일반 장치용 드라이버 인터페이스 제공
  - 특정 장치 드라이버 제공

1. 자원관리

파일(데이터) 관리

- 입출력 파일의 위치, 저장, 검색 관리 의미
- 컴퓨터 시스템은 물리적으로 다양한 형태로 파일 저장 가능
- 운영체제는 데이터의 효율적 사용을 위해 단일화된 저장 형태 제공
- 운영체제는 파일의 용이한 사용을 위해 보통 디렉터리로 구성, 다수의 사용자가 여기에 접근하려고 할 때는 이 접근을 제어
- 파일 관리를 위한 운영체제의 기능
  - 파일 생성, 삭제
  - 디렉터리 생성, 삭제
  - 보조기억장치의 파일 맵핑
  - 안전한(비휘발성) 저장장치에 파일 저장

2. 시스템 관리

시스템 보호(사용자 권한 부여)

- 보호 : 컴퓨터 자원에서 프로그램, 프로세스, 사용자의 접근 제어 방법
- 운영체제는 파일 사용 권한 부여, 데이터 암호화 등 서비스를 제공, 데이터와 시스템 보안
- 컴퓨터 시스템에서는 여러 프로세스 동시 실행 가능하므로 상호 보호해야 함
- 네트워크로 파일 공유 사이트에 접속 시 다른 사용자의 프로그램에서 보호

네트워킹(통신)

- 프로세서는 다양한 방법으로 구성된 네트워크 이용, 완전 접속과 부분 접속 방법으로 연결
- 연결된 프로세서가 통신을 할 때는 경로 설정, 접속 정책, 충돌, 보안 등 고려(운영체제가 관리)

명령 해석기

- 명령 해석기command interpreter는 운영체제에서 중요한 시스템 프로그램
- 대화형으로 입력한 명령어를 이해하고 실행하는 사용자와 운영체제의 인터페이스
- 사용자가 입력한 명령은 제어문으로 운영체제에 전달하는데, 이 전달을 명령 해석기가 담당
- 인터페이스 역할을 할 뿐 운영체제는 아님
- 커널과 분리하는 것이 좋음(명령 해석기의 인터페이스 변경 가능)
  - 분리하지 않으면 사용자가 커널의 코드를 변경할 수 없어 인터페이스를 변경 불가

Section 03 운영체제의 발전 과정과 유형

■ 운영체제의 발전 과정

표 2-1 운영체제의 발전 과정

연도	운영체제	특징
1940년대	운영체제 없음(작업별 순차 처리)	• 기계어를 직접 사용 • 단순 순차(직렬) 처리
1950년대	일괄 처리 시스템	• 운영체제의 효시인 IBM 701 개발 • 작업별로 일괄 처리 • 버퍼링, 스풀링 방법 등장
1960년대	• 다중 프로그래밍 시스템 • 시분할 시스템 • 다중 처리 시스템 • 실시간 처리 시스템	• 가상 기억장치 등장 • 다중 프로그래밍, 다중 처리, 시분할 처리 등 개념 등장 • 운영체제를 고급 언어로 작성 • 데이터 통신 지원용 운영체제 사용
1970년대 초반	• 다중 모드 시스템 • 범용 시스템	• 일괄 처리, 시분할 처리, 실시간 처리, 다중 프로그래밍 등을 제공하는 다중 모드 시스템 등장 • 장치의 독립성 제공 • TCP/IP 통신 표준 활성화 • 운영체제가 네트워크와 보안을 아우르는 수준으로 발전
1970년대 중반	분산 처리 시스템	• 각종 응용 프로그램 개발 및 데이터베이스 활용 확대 • 네트워크 기술의 발전 • 하드웨어에 운영체제 개념이 포함된 펌웨어 개념 등장

1. 운영체제의 발전 과정

■ 운영체제의 발전 과정

표 2-1 운영체제의 발전 과정(계속)

연도	운영체제	특징
1990년대	병렬 계산과 분산 계산	• 월드와이드웹의 등장으로 분산 컴퓨팅 증가 • GUI 강화 • 개인용과 서버용 운영체제의 보편화
2000년대 이후	• 모바일 및 임베디드 • 가상화 및 클라우드 컴퓨팅	• 네트워크 기반의 분산 및 병렬 운영체제의 보편화 • 모바일 장치와 가전제품을 위한 모바일 및 임베디드 운영체제의 보편화 • 다양한 기능, 확장성과 호환성 극대화 • 다양한 통신망의 확대와 개발형 시스템 발달 • 여러 운영체제가 한 시스템의 자원을 공유할 수 있게 해주는 서버 가상화 기술의 확산 • 컴퓨팅 자원, 스토리지, 소프트웨어 등을 사용자에게 서비스 형태로 제공하는 클라우드 컴퓨팅의 등장

1. 운영체제의 발전 과정

1940년대 : 운영체제 없음(작업별 순차 처리)

- 사용자가 기계어로 직접 프로그램 작성, 실행하는 작업별 순차 처리 시스템 사용
- 운영체제 개념 존재하지 않음
- 컴퓨터에 필요한 모든 작업 프로그램에 포함
- 카드 판독기에 판독의 시작·종료 시점, 데이터 해석 방법 등 포함
- 프로세서에는 명령어 저장 방법, 계산 대상, 결과 저장 위치와 방법, 출력 시점, 위치 등 모두 명령어로 명시적으로 표현
- 모든 작업을 예약으로 진행하여 문제가 발생

1. 운영체제의 발전 과정

1950년대 : 일괄 처리 시스템

- IBM 701 : 1952년 초 자동차 제조회사 GM에서 운영체제의 효시 개발
- IBM 704 : 1955년 GM과 북아메리카 항공사가 공동 개발
- IBM 704 자체 운영체제 : 1957년까지 IBM 사용자협회에서 개발
- 초기 운영체제인 일괄 처리 시스템batch processing system
  - 작업을 올리는 시간과 해제하는 시간 줄이는 데 관심(일괄 처리, 버퍼링, 스풀링 등 방법 도입)

- 일괄 처리
  - 일괄batch 처리는 직렬 처리 기술과 동일
  - 작업 준비 시간을 줄이려고 데이터가 발생할 때마다 즉시 처리하지 않고 데이터를 일정 기간 또는 일정량이 될 때까지 모아 두었다가 한꺼번에 처리

- 일괄 처리 장점
  - 많은 사용자와 프로그램이 컴퓨터 자원 공유
  - 컴퓨터 자원을 덜 사용 중일 때는 작업 처리 시간 교대 가능
  - 시시각각 수동으로 개입, 감독하여 컴퓨터 자원의 유휴 회피 가능

1. 운영체제의 발전 과정

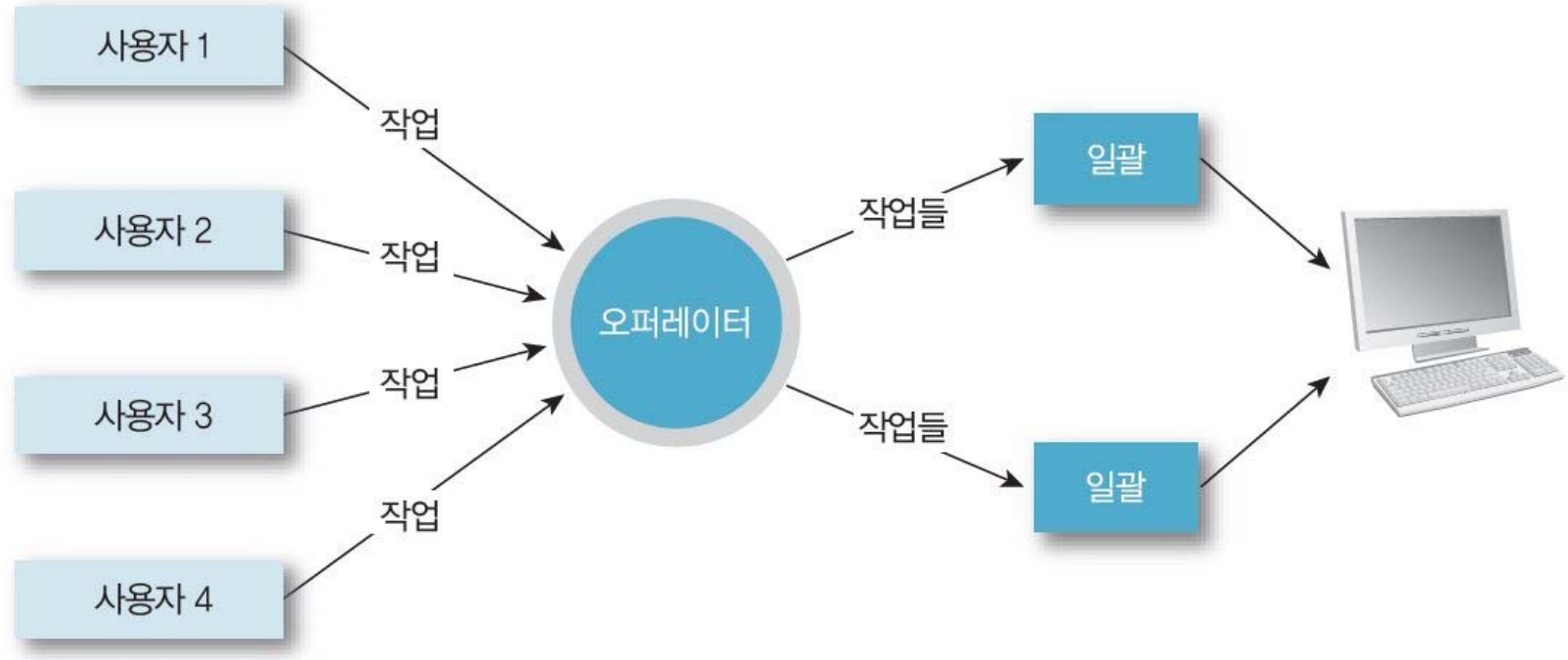


그림 2-6 **일괄 처리**

- 일괄 처리 단점
  - 준비 작업들의 유형이 동일해야 하고, 작업에 모든 유형의 입력 불가능
  - 입출력장치가 프로세서보다 속도 느려 프로세서의 유휴 상태 발생
  - 작업 우선순위 부여 곤란
  - 문제점 보완 위해 모니터링, 버퍼링, 스풀링 등 여러 방법 등장

1. 운영체제의 발전 과정

- 버퍼링buffering
  - 유휴시간이 없도록 입출력장치별로 입출력 버퍼 두어, 프로세서에서 연산 할 때 동시에 다른 작업 입출력하는 아주 간단한 방법

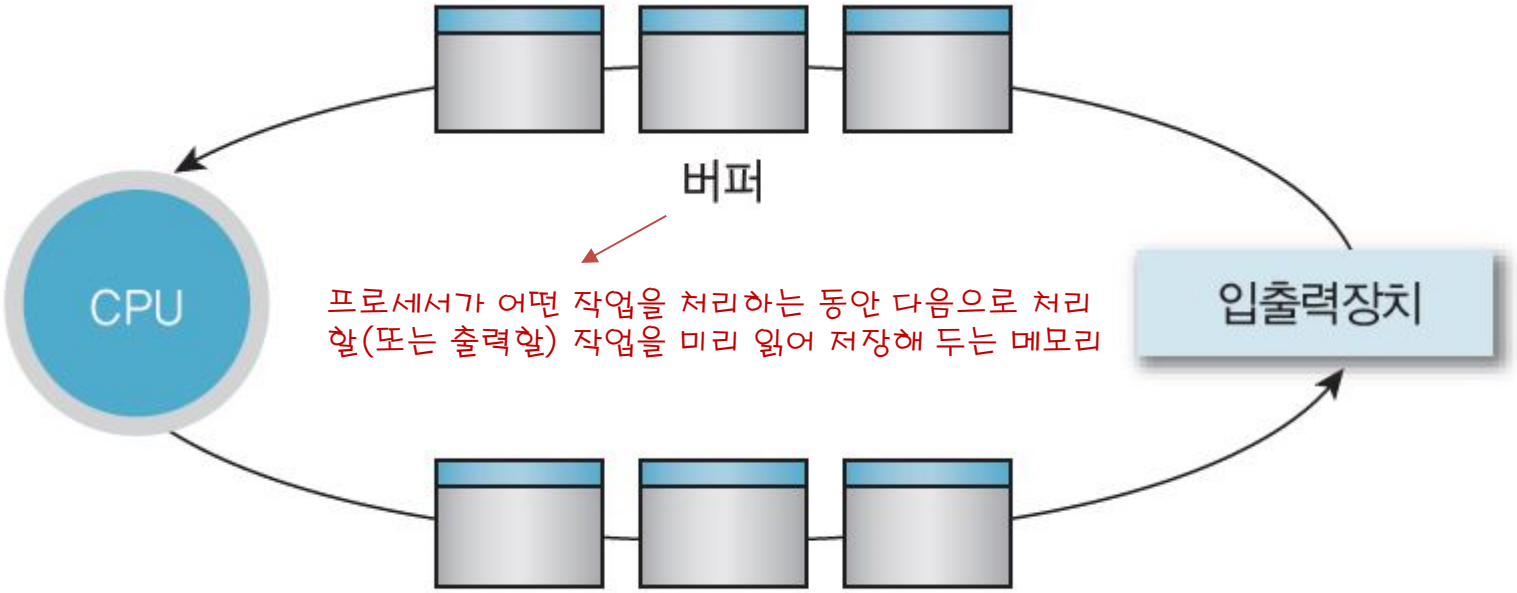


그림 2-7 버퍼링

1. 운영체제의 발전 과정

- 스풀링spooling, simultaneous peripheral operation on-line
  - 속도가 빠른 디스크를 버퍼처럼 사용 입출력장치에서 미리 읽는 것
  - 버퍼링이 컴퓨터 하드웨어의 일부인 버퍼를 사용 한다면, 스풀링은 별개의 오프라인 장치 사용
  - 버퍼링이 하나의 입출력 작업과 그 작업의 계산만 함께 할 수 있는 반면, 스풀링은 여러 작업의 입출력과 계산을 함께 할 수 있음
  - 프로세서에 일정한 디스크 공간, 테이블만 있으면 하나의 계산 작업과 다른 입출력 작업 중복 처리
  - 프로세서와 입출력장치가 고효율로 작업하게 함
  - 성능에 직접적으로 도움

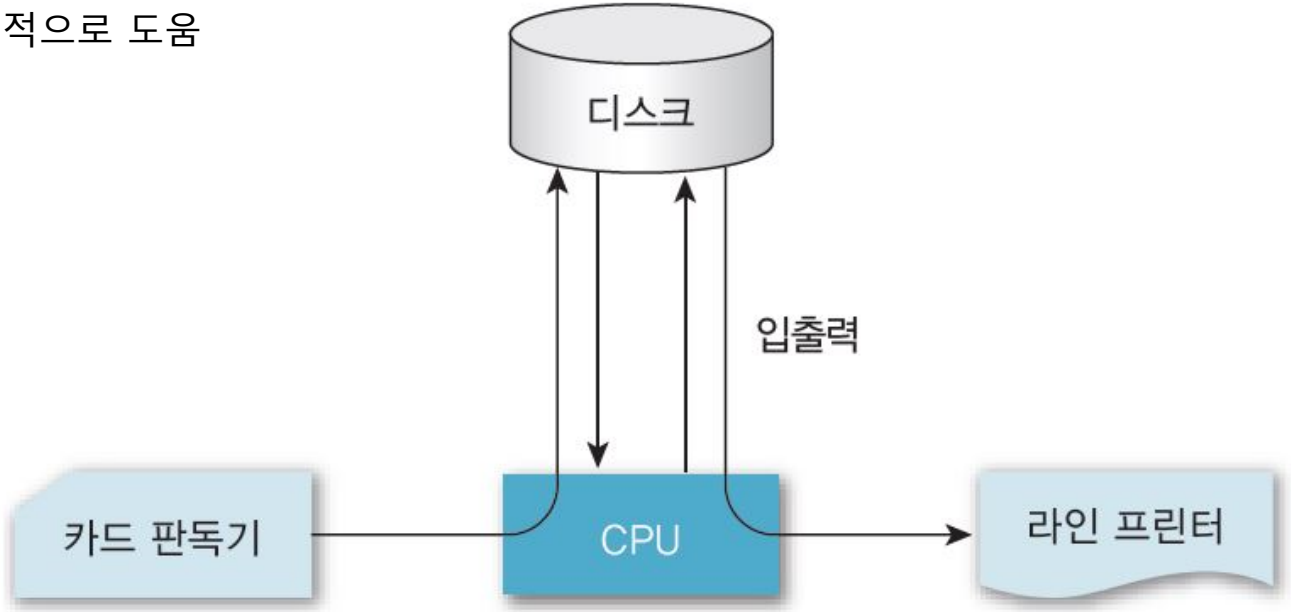


그림 2-8 스풀링

1. 운영체제의 발전 과정

1960년대 : 다중 프로그래밍·시분할·다중 처리·실시간 시스템

- 장치 독립성을 이용한 편리한 하드웨어 관리와 다중 프로그래밍
  - 여러 프로그램을 메모리에 나눠 적재한 후 프로세서를 번갈아 할당 프로세서 사용 극대화하여
  - 여러 프로그램을 동시에 실행
  - 장치 독립성 : 프로그램을 다른 입출력장치와 함께 실행할 수 있는 것

- 시분할, 다중 처리, 실시간을 이용한 시스템의 처리 능력 향상
  - 시분할 시스템 : 다중 프로그래밍 시스템에 프로세서 스케줄링이라는 개념을 더한 것
  - 다중 처리 시스템 : 하나의 시스템에서 프로세서를 여러 개 사용하여 처리 능력을 높인 것
  - 실시간 처리 시스템 : 즉시 응답

- 미항공사의 SABRE 예약 시스템을 개발
  - 멀리 떨어진 사용자가 단말기를 이용, 중앙 컴퓨터 시스템과 통신하는 트랜잭션 처리 시스템의 효시
  - 트랜잭션 처리 시스템은 사용자와 컴퓨터 시스템이 서로 대화를 하되, 사용자의 비교적 간단한 요구에 컴퓨터가 빠르게 응답하는 것(사용자 단말기를 컴퓨터의 온라인이라 함)

1. 운영체제의 발전 과정

1970년대 중반~1990년대 : 분산 처리 시스템, 병렬 계산과 분산 계산

- 컴퓨터 네트워크와 온라인 처리 방법 널리 사용
- 네트워크 이용하여 멀리 떨어진 컴퓨터 사용
- 마이크로프로세서가 등장, 개인용 컴퓨터 보유
- 사용자가 지역적으로 원격의 여러 시스템과 통신할 수 있어 정보 보호가 주요 관심사
- 1970년대 : 명령어 중심의 시스템 사용
- 1980년대 : 사용자에게 편리 한 메뉴 지향적인 시스템
- 1990년대 : GUI[Graphical User Interface] 시스템
- 분산 처리 개념을 확립하여 데이터 발생하는 곳으로 컴퓨터의 능력을 가져오는 데 관심

1. 운영체제의 발전 과정

2000년대 이후 : 모바일 및 임베디드, 가상화 및 클라우드 컴퓨팅

- 21세기에 접어들어 스마트폰이나 태블릿 같은 모바일 기기 대중화

- 모바일 운영체제Mobile Operating System : 모바일 장치나 정보 기기 제어 운영체제
  - 스마트폰용 : 노키아(심비안), 구글(안드로이드android,), 애플(iOS), RIM(블랙베리 OS), 마이크로소프트의 윈도우 등이 대표적이다.

- 사물 인터넷 IoT, Internet of Things 기술 등장 : 각종 사물에 컴퓨터칩과 통신 기능 내장
  인터넷에 연결
  - 초기에는 시스코 등 네트워크 기업, 나중에는 인텔, 퀄컴을 비롯한 반도체 칩 판매사가 사물인터넷을 전파하다가 이제는 애플이나 삼성전자 같은 제조사가 사물인터넷과 관련된 제품의 청사진과 플랫폼 구상 중
  - 특히 구글이나 아마존 등은 인터넷과 클라우드 기반 플랫폼에서 우위를 바탕으로 사물 인터넷 시장 선도하려고 하루가 멀다 하고 다양한 관련 기술과 서비스 발표.

1. 운영체제의 발전 과정

- 가상화virtualization 기술 본격 확산 : 물리적 자원 추상화, 논리적 자원 형태로 표현하는 기술
  - 처음 등장 시에는 도입 비용이 비싸고 사용 환경이 제한적이라 많이 사용하지 않음
  - 기술 발달로 경제성이 높아지면서 성능, 안정성, 효율성 향상 등 강점으로 본격적 확산
  - 적용 대상에 따라 서버 가상화, 데스크톱 가상화, 스토리지 가상화, 네트워크 가상화, 소프트웨어 가상화로 구분, 이 중 운영체제와 관련된 가상화 핵심은 서버 가상화

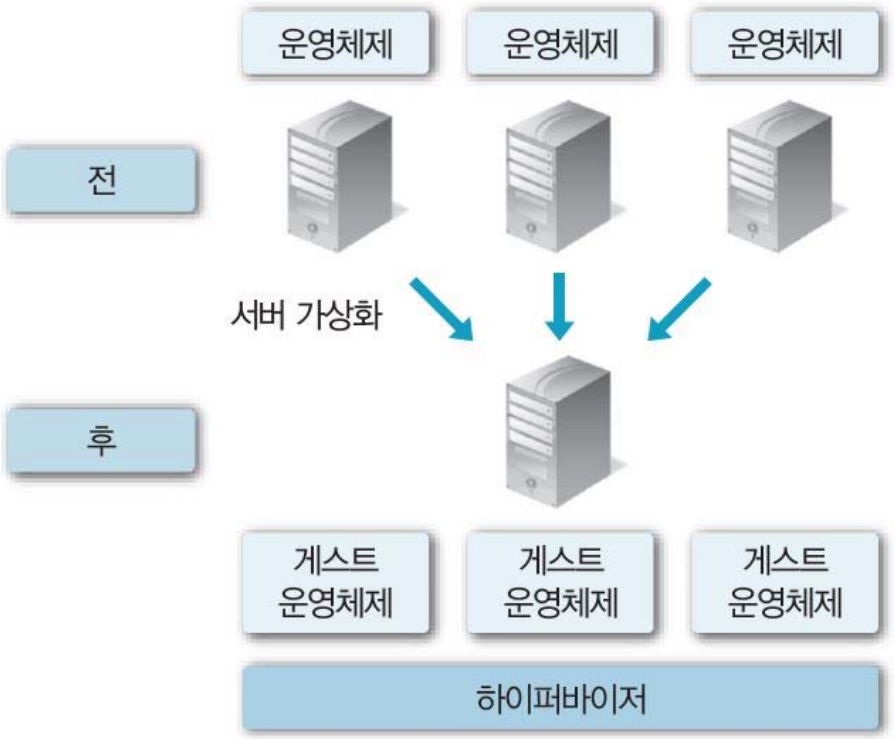


그림 2-9 서버 가상화의 개념

1. 운영체제의 발전 과정

- 서버 가상화 : 물리적 서버 하나에 가상 서버를 여러 개 구성하는 방법
  - 서버 하나에서 각 응용 프로그램과 운영체제를 독립된 환경으로 사용할 수 있어 여러 운영체제가 한 시스템의 자원 공유
  - 호스트기반 가상화 : 호스트 운영체제에서 가상 머신 구동
  - 베어메탈bare-metal 기반 가상화 : 호스트 운영체제 설치 전 가상화 솔루션을 탑재하여 가상의 CPU, 메모리, 디스크, 네트워크 카드 등 생성

그림 2-10 서버 가상화 종류

1. 운영체제의 발전 과정

▪ 클라우드 컴퓨팅

- 서버 가상화 기술 이용 사용자에게 컴퓨팅 자원, 스토리지, 소프트웨어 등을 서비스 형태로 제공

- 그리드 컴퓨팅의 분산 컴퓨팅 개념, 유틸리티 컴퓨팅의 과금 모델, 서버 기반 컴퓨팅의 처리 모델을 적용하여 다음 세 가지 특징을 보인다.
  - 클라우드 데이터 센터에서 원하는 만큼 컴퓨터 자원을 무한대로 사용
  - 컴퓨터 자원을 원할 때 원하는 만큼 늘리거나 줄일 수 있음
  - 컴퓨터 자원을 사용한 만큼 사용료 지불

- 클라이드 컴퓨팅 기술 이용 서비스 : IaaS(Infrastructure as a Service), PaaS(Platform as a Service), SaaS(Software as a Service)
  - IaaS : 데이터 센터에 있는 서버, 스토리지, 네트워크 등 인프라나 자원 가상화하여 인터넷으로 제공
    아마존(Amazon)의 EC2(Elastic Cloud Computing)와 S3(Simple Storage Service)가 있음
  - PaaS : 응용 프로그램의 구축, 테스트 및 설치가 가능한 통합 개발 환경을 웹으로 제공
    구글, 다음, 네이버에서 제공하는 Open API가 PaaS의 일종
  - SaaS : 특정 소프트웨어를 인터넷으로 제공, 해당 소프트웨어와 관련된 데이터를 클라우드에서 관리
    사용자는 웹 브라우저로 접속하여 소프트웨어(온디맨드(on demand)) 사용

2. 운영체제의 유형

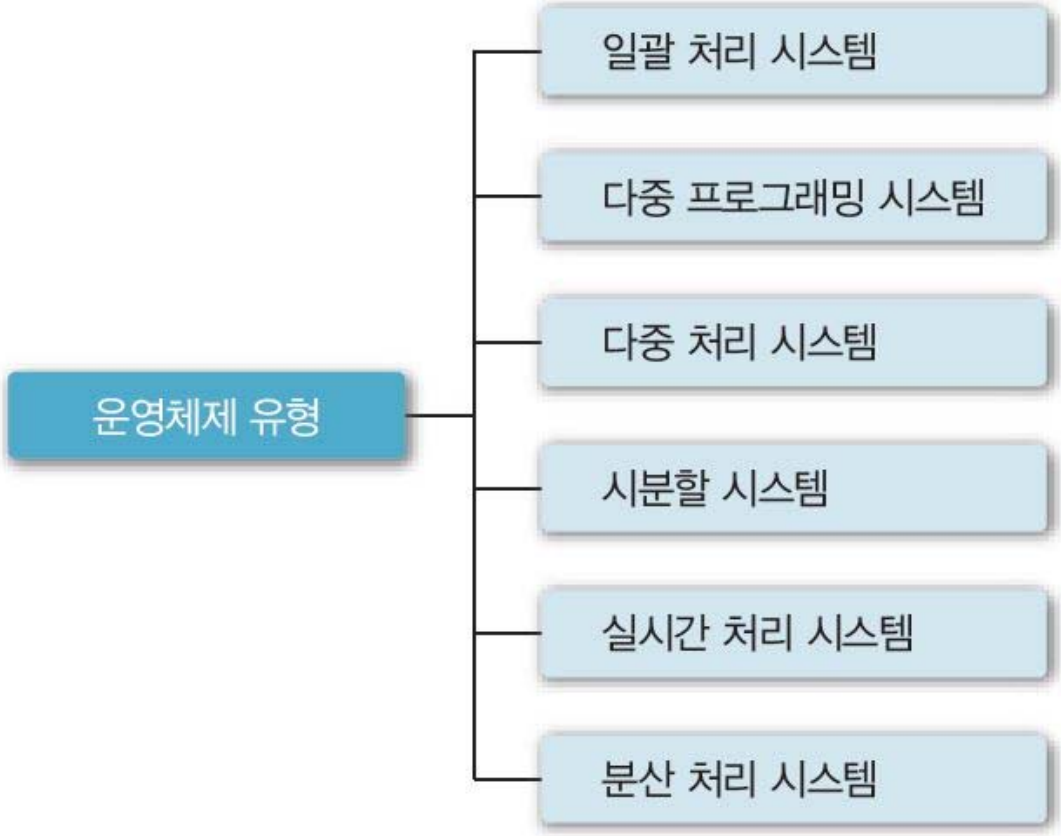


그림 2-11 운영체제의 유형

2. 운영체제의 유형

■ 다중 프로그래밍 시스템

- 프로세스가 다른 작업 수행 시 입출력 작업 불가능하여 프로세서와 메인 메모리의 활용도 떨어지는 일괄 처리 시스템의 큰 문제를 다중 프로그래밍 도입하여 해결

- 프로세서가 유휴 상태일 때 실행 중인 둘 이상의 작업이 프로세서를 전환 (인터리빙)하여 사용할 수 있도록 동작

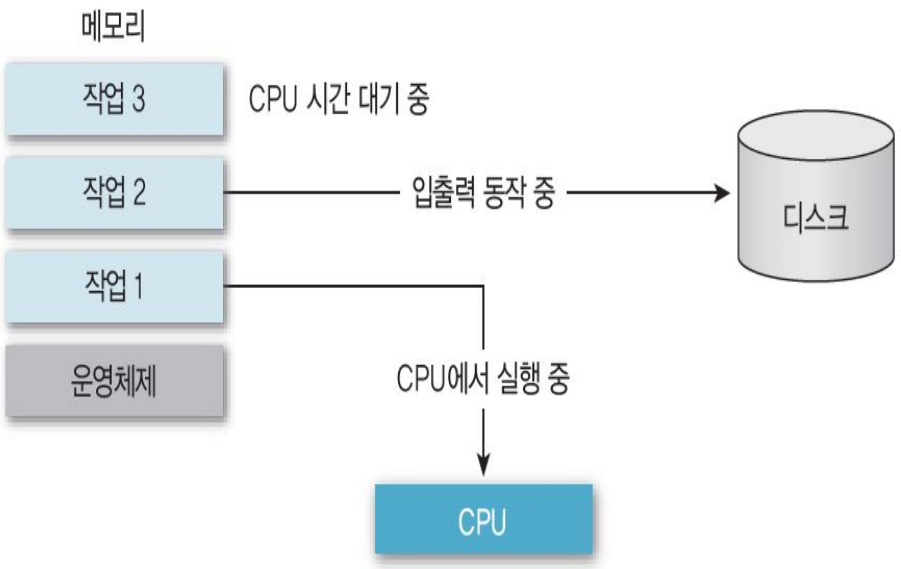


그림 2-12 다중 프로그래밍 시스템

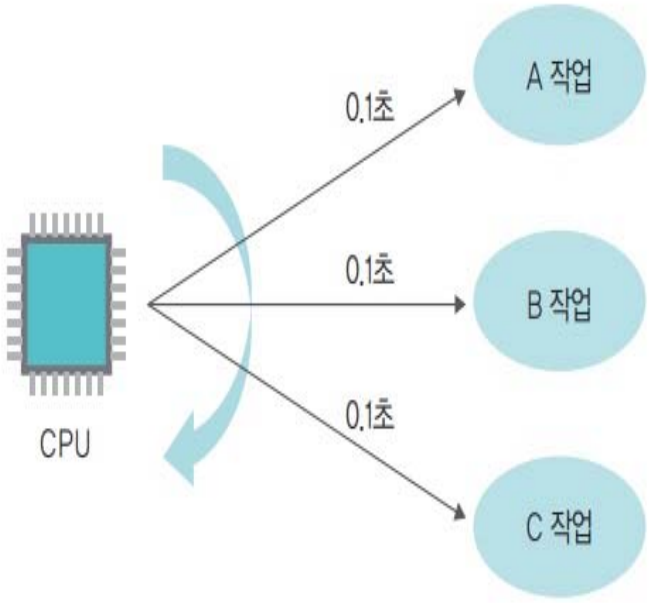


그림 1-12 다중 프로그래밍 시스템의 시간 분배

2. 운영체제의 유형

- 다중 프로그래밍의 특징
  - 높고 효율적인 프로세서 사용률(효율적인 운영) 증가
  - 많은 사용자의 프로그램이 거의 동시에 프로세서를 할당받는 듯한 느낌
  - 다중 프로그래밍 운영체제는 아주 복잡
  - 여러 작업을 준비 상태로 두려면 이를 메모리에 보관, 일정 형태의 메모리를 관리해야 함
  - 여러 작업이 수행할 준비를 갖추고 있으면, 이 중 하나를 선택하는 결정 방법 필요

    (인터럽트 이용 수행하는 프로세서 스케줄링의 다중 프로그래밍으로, 현재 운영체제의 중심 주제)

그림 2-13 다중 프로그래밍 시스템의 처리 방법 예

2. 운영체제의 유형

시분할 시스템TSS, Time Sharing System

- 다중 프로그래밍을 논리적으로 확장한 개념, 프로세서가 다중 작업을 교대로 수행
- 다수의 사용자가 동시에 컴퓨터의 자원을 공유할 수 있는 기술
- CTSSCompatible Time Sharing System : MIT 에서 개발, 1961년 IBM 709에 탑재, 사용
- 1970년 초까지는 시분할 시스템 만들기가 아주 어렵고 비용도 많이 들어 일반화 못함
- 각 프로그램에 일정한 프로세서 사용 시간 또는 규정 시간량 할당, 컴퓨터와 대화하는 형식으로 실행
- 여러 사용자에게 짧은 간격으로 프로세서 번갈아 할당, 마치 자기 혼자 프로세서를 독점하고 있는 양 착각하게 하여 여러 사용자가 단일 컴퓨터 시스템을 동시 사용 가능

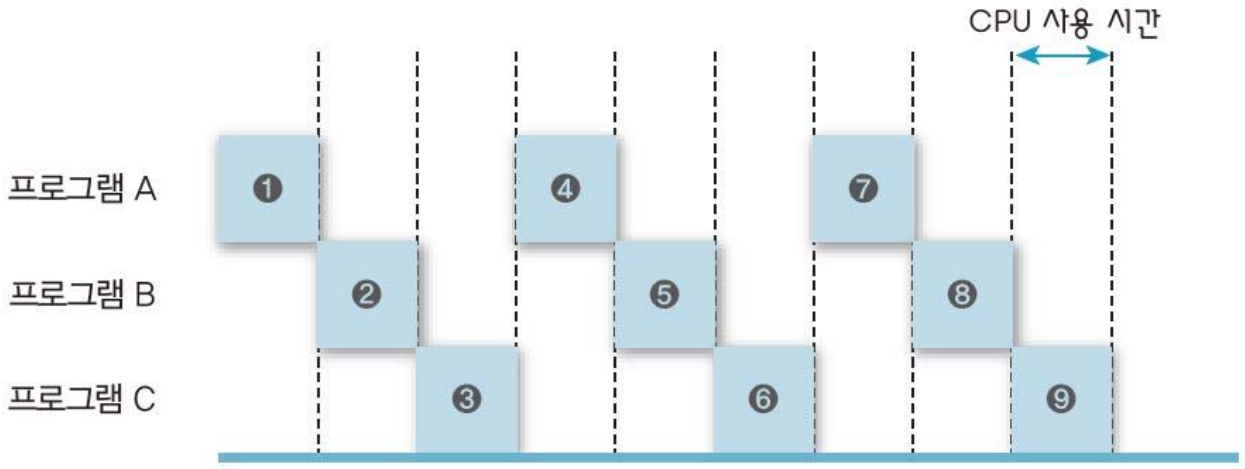


그림 2-14 시분할 시스템의 처리 방법 예

2. 운영체제의 유형

- 다중 프로그래밍 시스템과 시분할 시스템 특징
  - 메모리에 여러 프로그램을 적재하므로 메모리 관리 필요
  - 어떤 프로그램을 먼저 실행할지 결정하는 스케줄링 개념 필요
  - 다중 프로그래밍 시스템의 목표 : 프로세서 사용 최대화
  - 시분할 시스템의 목표 : 응답시간 최소화

표 2-2 시분할 시스템의 장점과 단점

장점	• 빠른 응답 제공 • 소프트웨어의 중복 회피 가능 • 프로세서 유휴시간 감소
단점	• 신뢰성 문제 • 보안 의문 및 사용자 프로그램과 데이터의 무결성 • 데이터 통신의 문제

2. 운영체제의 유형

다중 처리multiprocessing 시스템

- 단일 컴퓨터 시스템 내에서 둘 이상의 프로세서 사용, 동시에 둘 이상의 프로세스 지원
- 여러 프로세서와 시스템 버스, 클록, 메모리와 주변장치 등 공유
- 빠르고, 프로세서 하나가 고장 나도 다른 프로세서 사용하여 작업 계속, 신뢰성 높음
- 프로세서 간의 연결, 상호작용, 역할 분담 등을 고려해야 함
- 다중 처리 시스템을 구성하는 방법에는 비대칭(주종)적 구성과 대칭적 구성이 있음

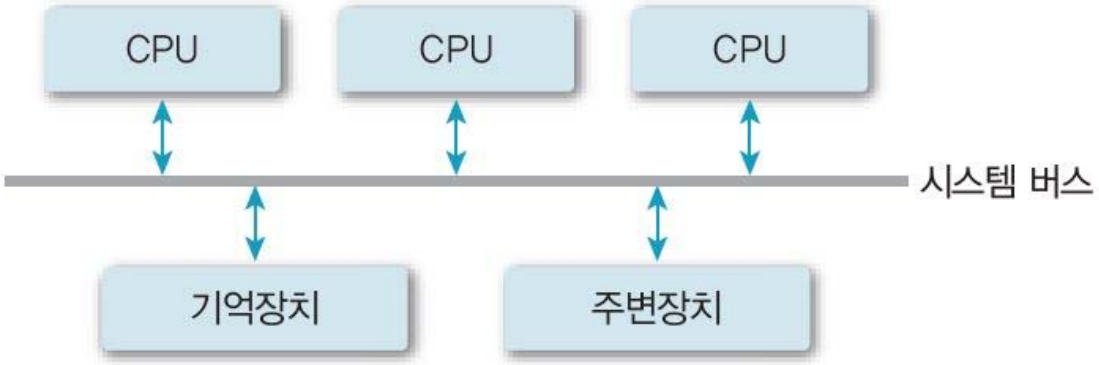


그림 2-15 다중 처리 시스템

2. 운영체제의 유형

실시간 처리 시스템real time processing system

- 데이터 처리 시스템으로 정의
- 입력에 응답하는 데 필요한 시간 간격이 너무 짧아 환경 제어
- 온라인 시스템은 실시간으로 할 필요 없지만, 실시간 처리 시스템은 항상 온라인 상태
- 입력 및 업데이트된 정보 요구 처리 후 디스플레이에 응답하는 시스템에 소요 시간을 반응(응답) 시간으로 함
- 반응시간은 프로세서에 이미 고정(반응시간이 온라인 처리에 비해 매우 짧음)
- 더 높은 적시 응답을 요구하는 장소에서 사용하거나 데이터 흐름 또는 프로세서 연산에 엄격한 시간 요구가 있을 때 사용 가능, 전용 응용 프로그램의 제어장치로도 사용
- 고정 시간 제약을 잘 정의하지 않으면 시스템 실패.
- 실시간 처리 시스템의 두 가지 유형
  - 경성 실시간 처리 시스템hard real time processing system
    - 작업의 실행 시작이나 완료에 대한 시간 제약 조건을 지키지 못할 때 시스템에 치명적인 영향을 주는 시스템
    - 무기 제어, 발전소 제어, 철도 자동 제어, 미사일 자동 조준 등이 이에 해당
    - 보장되는 컴퓨팅, 시간의 정확성과 컴퓨팅 예측성을 갖게 해야 함
  - 연성 실시간 처리 시스템soft real time processing system
    - 작업 실행에서 시간 제약 조건은 있으나, 이를 지키지 못해도 전체 시스템에 치명적인 영향을 미치지 않는 시스템
    - 동영상은 초당 일정 프레임frame 이상의 영상을 재생해야 한다는 제약이 있으나, 일부 프레임을 건너뛰어도 동영상을 재생 시스템에는 큰 영향을 미치지 않음

2. 운영체제의 유형

분산 처리 시스템distributed processing system

- 시스템마다 독립적인 운영체제와 메모리로 운영, 필요 시 통신하는 시스템
- 사용자에게는 중앙집중식 시스템처럼 보이는데, 다수의 독립된 프로세서에서 실행
- 데이터를 여러 위치에서 처리·저장, 여러 사용자가 공유
- 하나의 프로그램을 여러 프로세서에서 동시에 실행

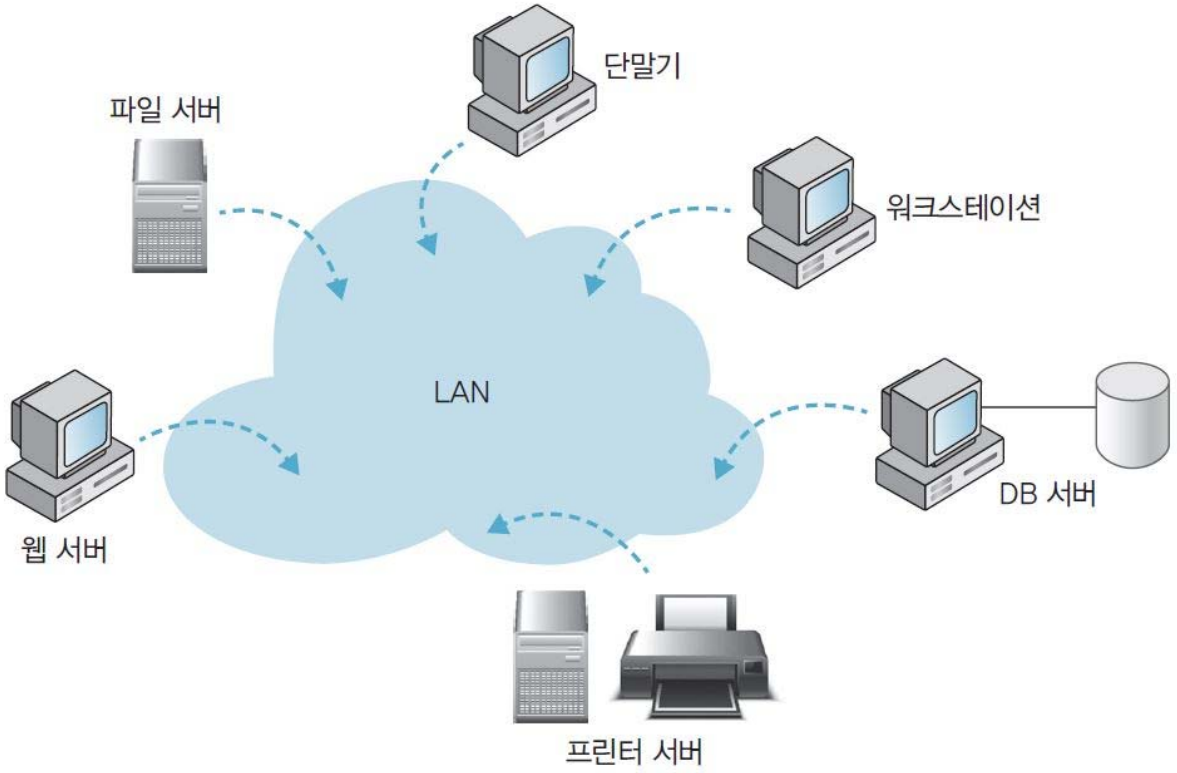
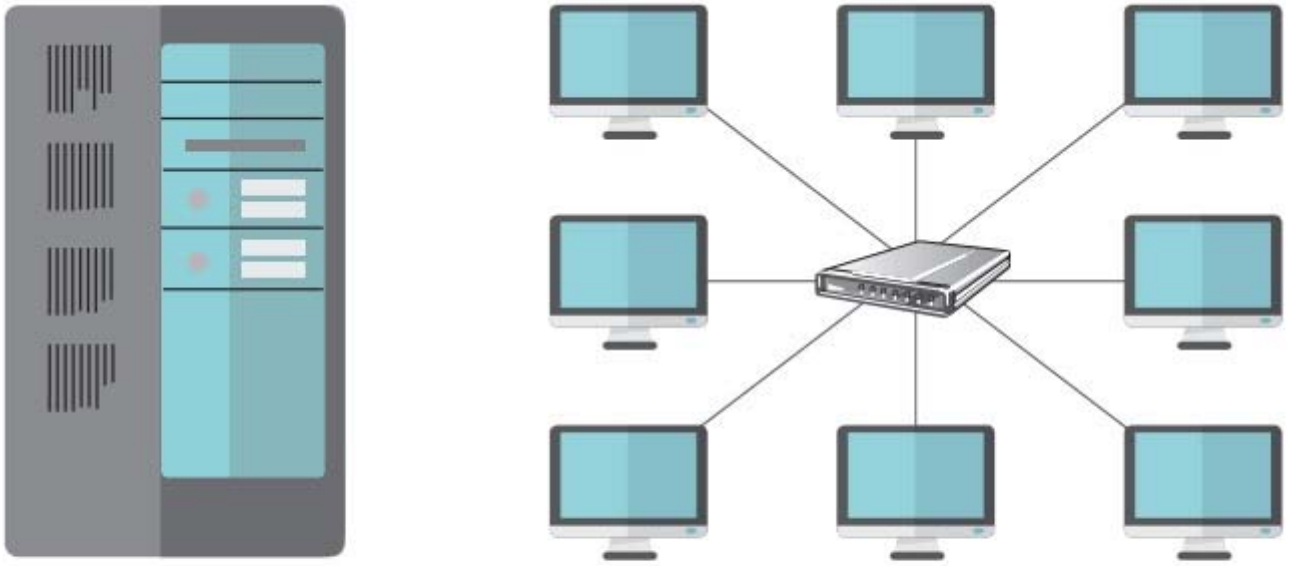


그림 2-16 분산 처리 시스템

2. 운영체제의 유형

■ 분산 처리 시스템distributed processing system

- 개인용 컴퓨터와 인터넷이 보급되면서 값이 싸고 크기가 작은 컴퓨터를 하나로 묶어서 대형 컴퓨터의 능력에 버금가는 시스템을 만들 수 있게 됨
- 네트워크상에 분산되어 있는 여러 컴퓨터로 작업을 처리하고 그 결과를 상호 교환하도록 구성한 시스템



(a) 메인프레임 (b) 분산 시스템

그림 1-13 메인프레임과 분산 시스템

2. 운영체제의 유형

클라이언트/서버 시스템

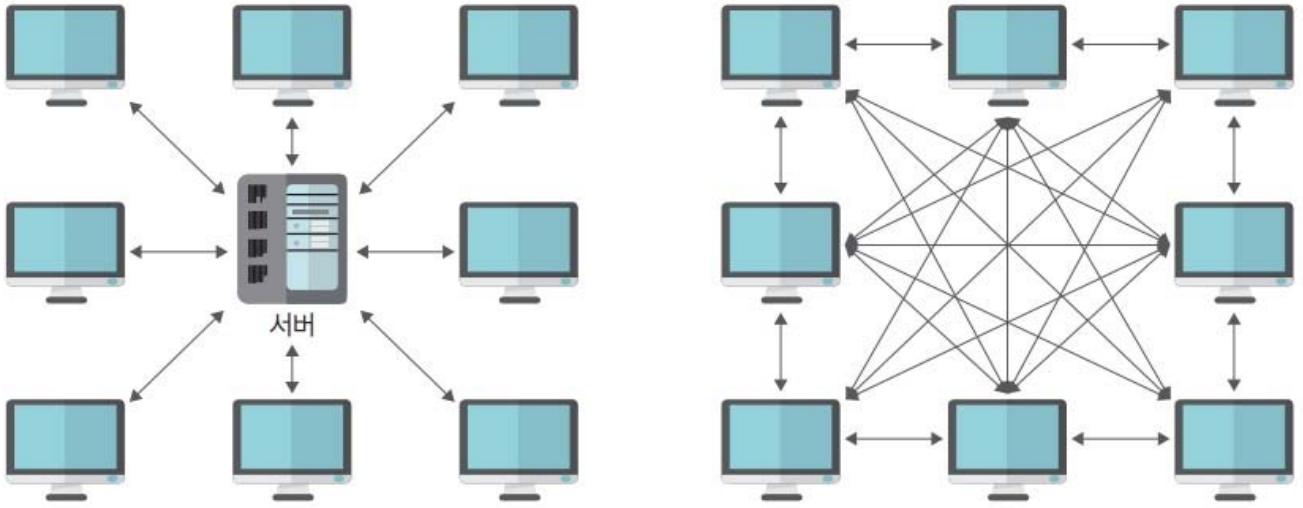
- 작업을 요청하는 클라이언트와 거기에 응답하여 요청받은 작업을 처리하는 서버의 이중구조로 나뉨
- 웹 시스템이 보급된 이후 일반인들에게 알려짐

그림 1-14 클라이언트/서버 구조

P2P 시스템(2000년대 초반~현재)

P2P 시스템

- 클라이언트/서버 구조의 단점인 서버 과부하를 해결하기 위해 만든 시스템
- 서버를 거치지 않고 사용자와 사용자를 직접 연결
- 냅스터(mp3공유 시스템)에서 시작하여 현재는 메신저나 토렌트 시스템에서 사용



(a) 클라이언트/서버 시스템 (b) P2P 시스템

그림 1-15 클라이언트/서버 시스템과 P2P 시스템

P2P 시스템(2000년대 초반~현재)

P2P 시스템의 예 : 메신저

- P2P 기술은 불법 소프트웨어 기술 규제 때문에 발전하지 못하다가 메신저 프로그램에 도입되어 큰 발전을 이룸
- 수만 명이 동시에 채팅을 하고 파일을 주고받는 메신저 시스템은 P2P 기술을 이용하면 서버의 부하 없이 구현할 수 있음

P2P 시스템의 예 : 파일 공유

- 10명에게 데이터를 받는다면 1명에게 데이터를 받을 때보다 속도가 10배 빠를 뿐 아니라, 데이터를 받는 도중 1~2명이 프로그램을 중단해도 다른 사람에게 나머지를 받을 수 있음

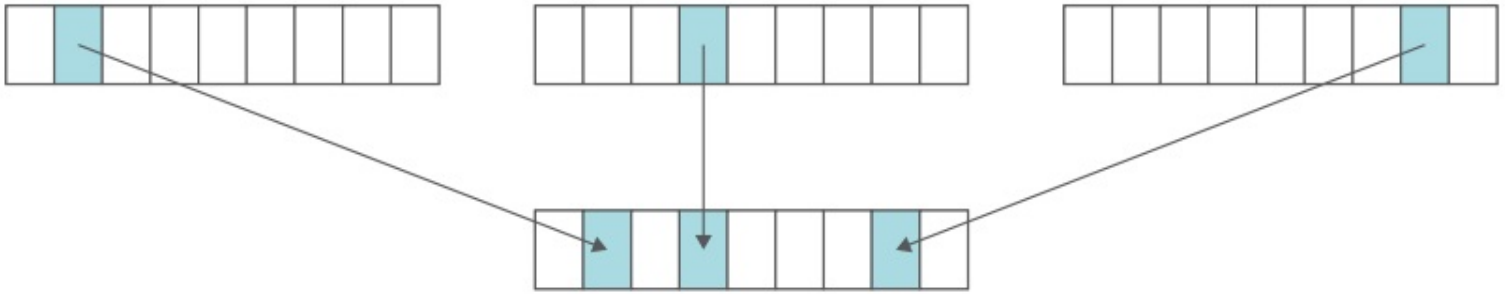


그림 1-17 대용량 파일 공유 P2P 시스템

기타 컴퓨팅 환경(2000년대 초반~현재)

그리드 컴퓨팅

- 필요한 기간만큼만 컴퓨터를 사용하고 사용한 금액만큼만 돈을 지불할 수 컴퓨팅 환경
- 서로 다른 기종의 컴퓨터들을 묶어 대용량의 컴퓨터 풀을 구성하고 이를 원격지와 연결하여 대용량 연산을 수행하는 컴퓨팅 환경
- 그리드가 하드웨어적인 컴퓨팅 환경의 통합이라고 한다면 SaaS(Software as a Service; 사스)는 사용자가 필요한 소프트웨어 기능만을 필요할 때 이용하고, 이용한 기능만큼만 비용을 지불하는 개념
- CPU 관리, 저장소 관리, 보안 조항, 데이터 이동, 모니터링과 같은 서비스를 위한 표준 규약 생성에 기여

Section 04 운영체제의 서비스

운영체제의 서비스 제공

- 부팅 서비스 : 컴퓨터 하드웨어 관리, 프로그램을 실행할 수 있도록 컴퓨터에 시동
- 사용자 서비스 : 프로그래머가 프로그래밍 작업을 쉽게 수행할 수 있도록 함
- 시스템 서비스 : 시스템의 효율적인 동작 보장
- 시스템 호출 : 프로그램이 운영체제의 기능을 서비스 받을 수 있는 프로그램과 운영체제 간의 인터페이스 제공

1. 부팅 서비스

■ 부팅booting 또는 부트스트래핑bootstrapping

- 운영체제를 메인 메모리에 적재하는 과정
- 부트 로더는 부트스트랩 로더bootstrap loader 줄인 말로 하드디스크와 같은 보조기억장치에 저장된 운영체제를 메인 메모리에 적재하는 ROM에 고정시킨 소규모 프로그램

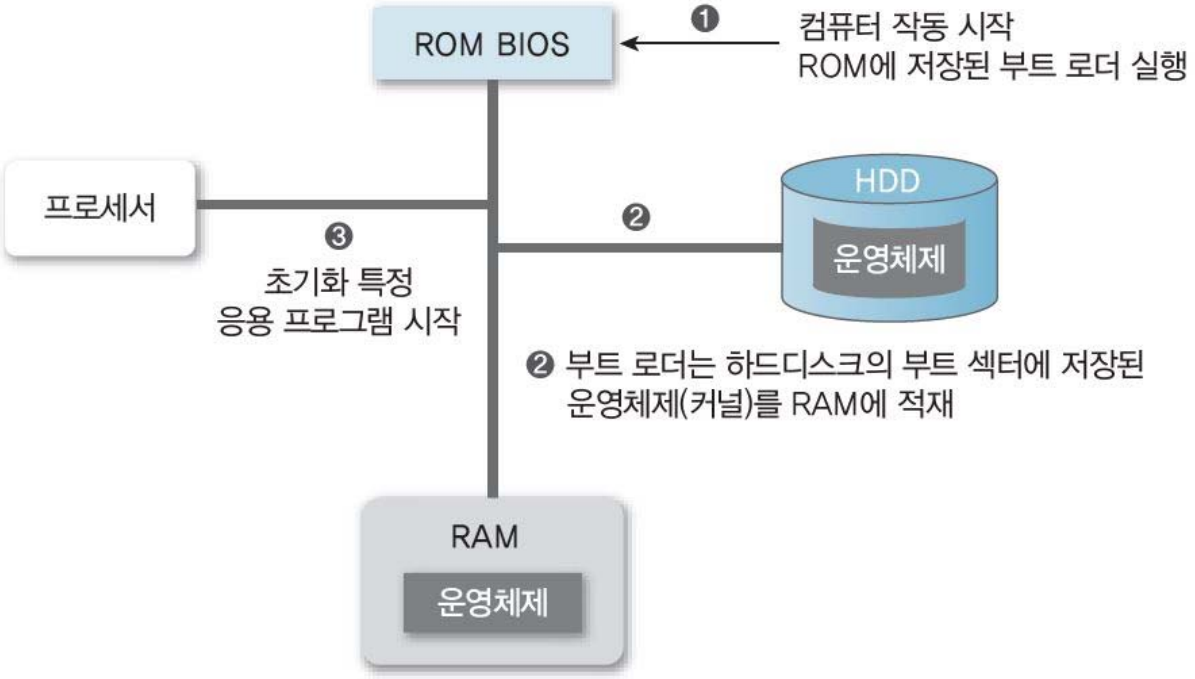


그림 2-17 부팅 과정

2. 사용자 서비스

■ 사용자 인터페이스 제공

▪ 사용자 인터페이스 : 사용자와 컴퓨터 간의 상호작용 발생 공간(CLI, 메뉴, GUI 등 구현)

- CLI Command Line Interface(명령 라인 인터페이스)

  사용자가 키보드 등으로 명령어 입력하여 시스템에서 응답 받은 후, 또 다른 명령어를 입력하여 시스템을 동작하게 하는 텍스트 전용 인터페이스. 사용자가 프롬프트에서 명령어를 입력하여 컴퓨터와 상호작용할 수 있고, 명령어 입력한 후 반드시 Enter 를 눌러야 함

- 메뉴 인터페이스

  메뉴 등을 사용하여 시스템과 상호작용. 사용 매우 편리, 배우거나 기억해야 할 명령 없음

  iPad나 휴대폰, 현금 자동 인출기ATM 등이 대표적

- GUI Graphical User Interface(그래픽 사용자 인터페이스)

  윈도우 환경에서 사용자에게 정보와 작업을 표현하는 텍스트, 레이블이나 텍스트 탐색과 함께 그래픽 아이콘과 시각적 표시기, 버튼이나 스크롤바와 같은 위젯widget 그래픽 제어 요소를 사용

  컴퓨터와 상호작용 할 수 있는 가장 보편적인 유형

  마이크로소프트의 윈도우나 애플의 맥 OS에 사용하는 방법이 대표적

2. 사용자 서비스

프로그램 실행

- 프로그램 실행하려면 먼저 메모리에 적재, 프로세서 시간 할당
- 운영체제는 프로그램을 실행하려고 메모리 할당이나 해제, 스케줄링 등 중요 작업 처리

입출력 동작 수행

- 수행 중인 프로그램은 입력이 필요, 사용자가 제공하는 입력 처리 후에는 출력을 생성
- 운영체제는 입출력 동작 직접 수행할 수 없는 사용자 프로그램의 입출력 동작 방법 제공

파일 시스템 조작

- 사용자는 디스크에서 파일 열고, 저장, 삭제하는 등 다양하게 파일 조작
- 디스크에 파일을 저장하면 특정 블록에 할당 저장, 파일을 삭제하면 파일 이름 제거되면서 할당한 블록이 자유롭게 됨.
- 운영체제는 파일 시스템 조작 서비스 제공, 사용자가 파일 관련 작업을 쉽게 할 수 있게 함.

2. 사용자 서비스

통신(네트워크)

- 프로세스가 다른 프로세스와 정보를 교환하는 방법
  - 동일한 컴퓨터에서 수행하는 프로세스 간의 정보 교환
  - 두 번째는 네트워크로 연결된 컴퓨터 시스템에서 수행하는 프로세스 간의 정보 교환
- 운영체제는 다중 작업 환경에서 공유 메모리를 이용하거나 메시지 전달로 다양한 유형의 프로세스와 통신 지원

오류 탐지

- 운영체제는 가능한 모든 하드웨어와 소프트웨어 수준에서 오류 탐지, 시스템 모니터링하여 조정함으로써 하드웨어 문제 예방
- 입출력 장치에 관련된 오류와 메모리 오버플로, 하드디스크의 불량 섹터 검출, 부적당한 메모리 접근과 데이터 손상 등
- 운영체제는 다음 오류 유형을 감지한 후 유형별로 적절히 조치
  - 프로세서, 메모리 하드웨어와 관련된 오류 : 기억장치 메모리 오류, 정전
  - 입출력장치 오류 : 테이프의 패리티 오류, 카드 판독기의 카드 체증jam, 프린터의 종이 부족
  - 사용자 프로그램 오류 : 연산의 오버플로, 부적당한 기억장치 장소 접근, 프로세서 시간 과다 사용

3. 시스템 서비스

- **시스템 서비스 : 사용자가 아닌 시스템 자체의 효율적 동작 보장하는 기능**
  - 여러 사용자가 사용하는 시스템에서 컴퓨터 자원 공유하여 시스템 자체의 효율성 높임
- **자원 할당**
  - 운영체제는 다수의 사용자나 작업 동시 실행 시 운영체제가 자원을 각각 할당하도록 관리
  - 프로세서 사이클, 메인 메모리, 파일 저장 장치 등은 특수한 할당 코드를 갖지만, 입출력장치 등은 더 일반적인 요청과 해제 코드 가질 수 있음
- **계정**
  - 운영체제는 각 사용자가 어떤 컴퓨터 자원을 얼마나 많이 사용하는지 정보 저장 추적
  - 이 정보는 사용자 서비스 개선을 위해 시스템 재구성하는 연구자에게 귀중한 도구가 됨
- **보호와 보안**
  - 운영체제는 다중 사용자 컴퓨터 시스템에 저장된 정보 소유자의 사용을 제한
  - 서로 관련이 없는 여러 작업을 동시에 수행할 때는 한 작업이 다른 작업이나 운영체제를 방해하지 못하게 해야 함.
  - 보호 : 시스템 호출 하려고 전달한 모든 매개변수의 타당성 검사하고, 시스템 자원에 모든 사용자 접근을 제어하도록 보장하는 것
  - 보안 : 잘못된 접근 시도에서 외부 입출력장치 방어, 외부에 사용자 인증을 요구하는 것

4. 시스템 호출system call

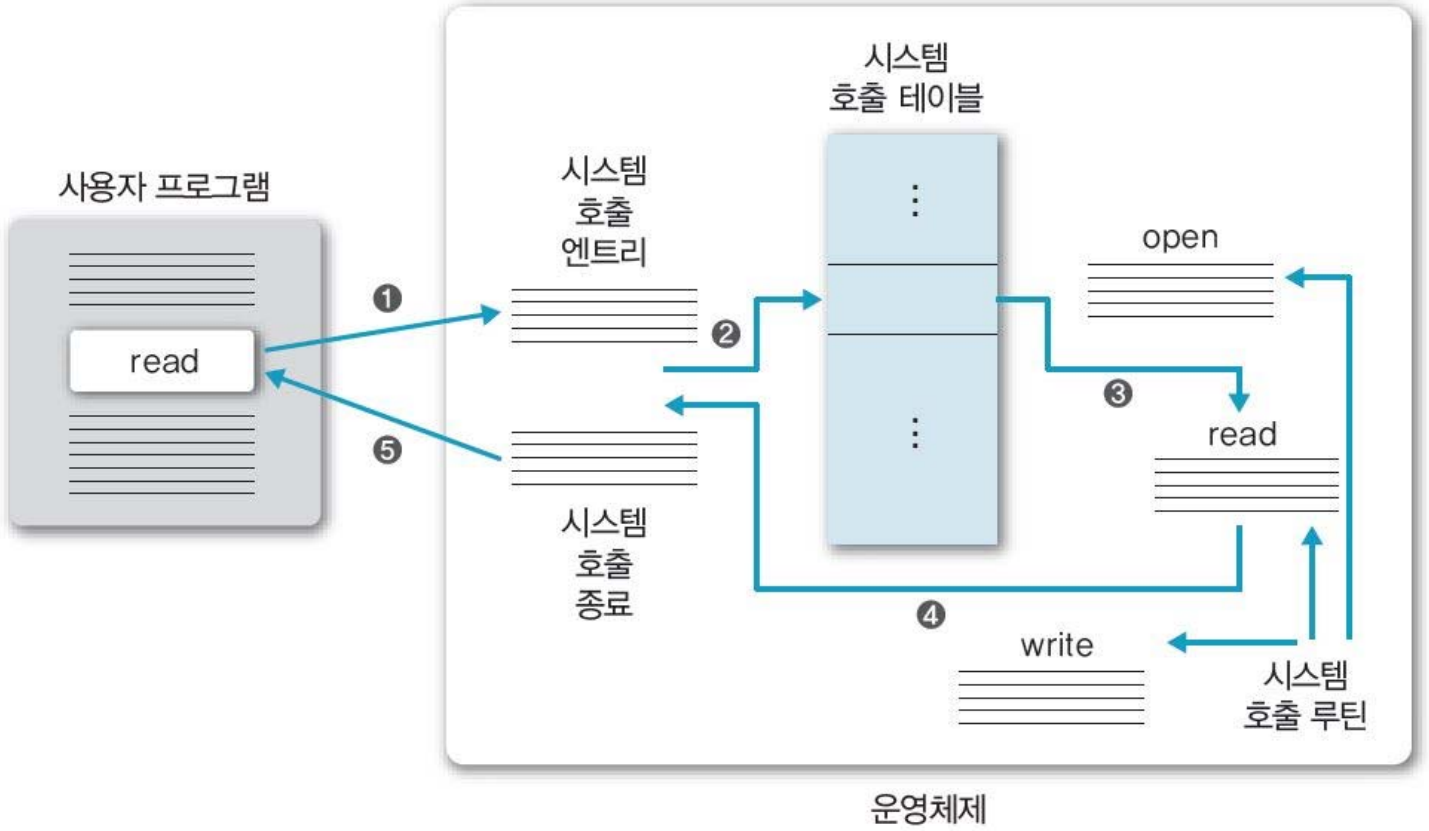
- 실행 중인 프로그램과 운영체제 간의 인터페이스, APIApplication Programming Interfaces라고도 함
- 사용자 프로그램은 시스템 호출을 하여 운영체제의 기능 제공 받음
- 핵심 커널 서비스와 통신, 새로운 프로세스의 생성과 실행, 하드웨어 관련 서비스 등이 있음
- 시스템과 상호작용하는 동작은 대개 사용자 수준 프로세스에서는 사용할 수 없으나, 시스템 호출을 하여 운영체제에 서비스를 요청할 수 있음
- 시스템 호출 방법
  - 프로그램에서 명령이나 서브루틴의 호출 형태로 호출
  - 시스템에서 명령 해석기를 사용하여 대화 형태로 호출
- 운영체제가 제공하는 일반적인 시스템 호출
  - 프로세스 제어, 파일 조작, 장치 관리, 정보 유지 등

4. 시스템 호출system call

표 2-3 운영체제의 시스템 호출 서비스

호출 서비스	설명	
프로세스 제어	• 종료와 취소 • 적재load와 실행 • 프로세스 생성과 종료	• 프로세스 속성 획득과 지정 • 대기와 대기 이벤트, 신호 이벤트 • 메모리 할당과 해제
파일 조작	• 파일 생성과 삭제 • 파일 열기open와 닫기close	• 파일 읽기와 쓰기, 파일 재배치reposition • 파일 속성 획득과 지정
장치 조작	• 장치 요구와 해제 • 장치 읽기와 쓰기, 재배치	• 장치 속성 획득과 설정 • 논리적 부착이나 장치 제거
정보 관리	• 시간과 날짜의 설정과 획득 • 데이터의 설정과 획득 • 프로세스, 파일, 장치 속성의 설정과 획득	
통신	• 통신 연결의 생성과 제거 • 메시지의 송수신	• 정보 상태 전달 • 원격 장치의 부착 및 제거

4. 시스템 호출system call



❶ 시스템 호출 서비스 요청

❷ 스위치 모드 : 서비스 인자와 검증

❸ 시스템 호출 테이블을 이용하여 서비스 루틴으로 분기

❹ 스위치 모드 : 서비스 루틴에서 반환

❺ 시스템 호출에서 반환

그림 2-18 **시스템 호출 예**

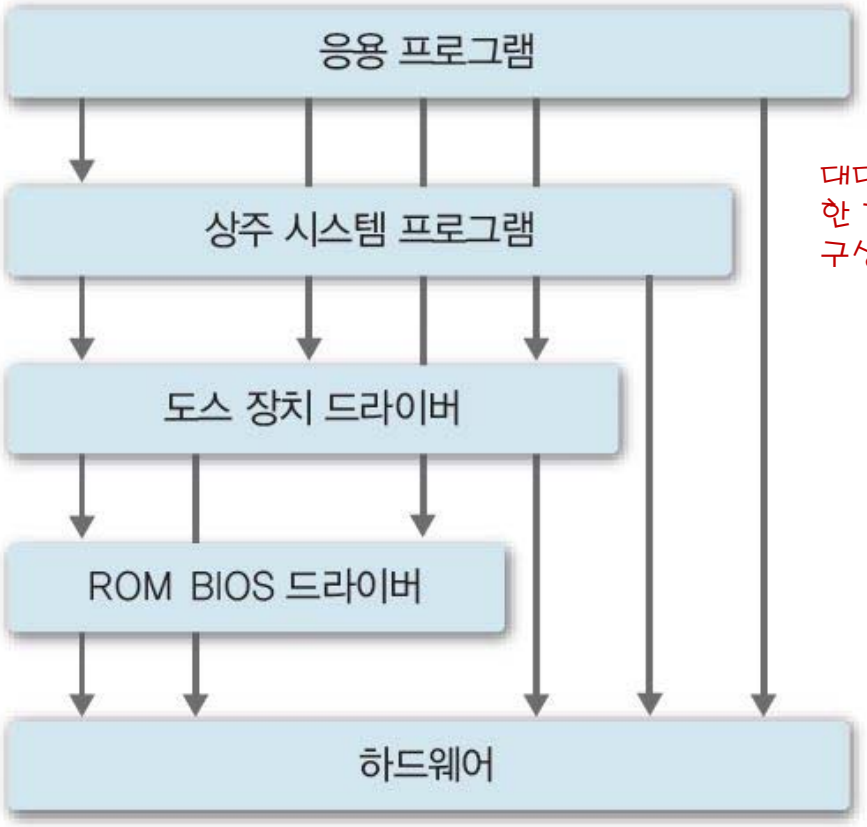
Section 05 운영체제의 구조

- 운영체제가 점점 더 다양한 하드웨어와 소프트웨어 지원하면서 구조 또한 복잡해짐
- 복잡한 시스템은 설계, 구현, 테스트, 유지 보수 등 모든 면에서 어려움
- 이것의 해결을 위해 운영체제를 설계하는 다양한 방법 등장

1. 단일 구조 운영체제

단일monolithic 구조(또는 모놀리식 커널 구조) 운영체제

- 초기에 생겨난 가장 보편적 형태
- 운영체제의 모든 기능을 커널과 동일한 메모리 공간에 적재 후 시스템 호출만으로 사용
- 작고 간단하면서 시스템 기능이 제한된 구조



대다수 기능을 최소한의 영역으로 제공하는데, 한 계층으로는 결합할 수 없는 많은 기능으로 구성

그림 2-19 단일 구조 운영체제 예 1 : 도스

1. 단일 구조 운영체제

- 특징
  - 대부분의 기능을 커널에 그룹화해서 구현, 직접 통신하여 시스템 자원을 효율적 관리
  - 커널 크기가 상대적으로 커지면서 버그의 원인이나 기타 오류 구분 어려움
  - 새 기능을 추가하는 수정과 유지 보수 매우 어려움
  - 동일한 메모리에서 실행하므로 한 부분에서 발생한 문제로 시스템 전체에 심각한 영향 가능
  - 악성 코드로 피해 입기 쉬움

그림 2-20 단일 구조 운영체제 예 2 : 초기 유닉스

1. 단일 구조 운영체제

단일형 구조 운영체제

- 장점
  - 모듈 간의 통신 비용이 줄어들어 효율적인 운영이 가능

- 단점
  - 모든 모듈이 하나로 묶여 있기 때문에 버그나 오류를 처리하기가 어려움
  - 운영체제의 여러 기능이 서로 연결되어 있어 상호 의존성이 높기 때문에 기능상의 작은 결함이 시스템 전체로 확산될 수 있음
  - 다양한 환경의 시스템에 적용하기 어려움
  - 현대의 운영체제는 매우 크고 복잡하기 때문에 완전 단일형 구조의 운영체제를 구현하기가 어려움

2. 계층 구조 운영체제

- 운영체제가 점점 커지고 복잡해지면서 순수 단일 구조만으로는 다루기가 어려워졌다. 이 문제를 해결하려고 등장
- 계층 구조에서는 비슷한 기능을 수행하는 요소를 그룹화하여 계층적으로 구성
- 사용자 프로세스의 요청을 수행할 때 여러 계층을 거쳐야 하므로, 한 계층에서 다음 계층으로 데이터를 전달할 때마다 추가적인 시스템 호출 발생
- 호출 한 번으로 서비스를 받는 단일 구조보다는 성능 낮음
- 단일 구조 운영체제보다 모듈화가 잘 되어 있음
- 시스템 검증과 오류 수정 용이
- 첫 번째 계층은 기본 하드웨어 사용하여 기능을 만들어 나머지 시스템에 의문을 가지지 않고 오류를 수정 가능
- 첫 번째 계층의 오류를 수정하면 기능이 정확하다고 가정하여 두 번째 계층 만듬(이 과정 반복)
- 특정 계층에서 오류 발견해도 하위 계층은 오류 수정했기 때문에 해당 계층에 오류가 없다고 할 수 있음
- 시스템을 계층으로 나누면 시스템 설계나 구현 단순해짐

2. 계층 구조 운영체제

- THE 운영체제에서 처음 사용(1968년 다익스트라Dijkstra 개발)
- 계층 정의 어려움
- 각 계층은 자신의 하위 계층만 사용할 수 있으므로 신중히 설계해야 함
- 대형 시스템의 프로세서 스케줄러에서 메모리에 적재할 수 있는 것보다 활동 중인 프로세스를 더 많이 처리하려면 교체(스와핑swapping) 기능 필요
- 모든 계층이 시스템에 제한 없이 접근할 수 있어 오류나 악성 코드에 민감하게 반응할 수 있음

2. 계층 구조 운영체제

그림 2-22 계층 구조 운영체제 예 : THE 운영체제

3. 마이크로 커널 구조 운영체제

- 단일 커널의 문제점 해결 위해 1980년대 카 네기멜론대학교에서는 모듈화 된 마이크로 커널micro-kernel을 사용하여 매크Mach를 만듬
- 커널 기능 많이 제외
- 커널에는 최소 기능만 포함시켜 크기를 대폭 줄이고 기타 기능은 사용자 공간으로 옮겨 사용자 영역에서 수행하는 서버 구현
- 하드웨어 초기화, 메모리 관리(주소 공간 관리), 프로세스(스레드) 관리, 프로세스 간 통신, 프로세스 간 협력 동기화 기능 등 기본 기능만 실행
- 네트워크 시스템, 파일 시스템 상호작용과 장치 관리 등 대부분의 운영체제 구성 요소는 커널 외부, 즉 사용자 영역의 서버로 옮겨 구현

3. 마이크로 커널 구조 운영체제

- 운영체제 서비스를 사용자 영역의 독립적인 서버에서 수행, 서버에서 잘못 수행하더라도 다른 서버와 커널에 치명적인 영향을 주지 않음
- 운영체제의 많은 기능을 사용자 영역의 서버로 구현할 수 있기 때문에 서버 개발 용이, 운영체제의 기능을 쉽게 변경 가능
- 모듈화 정도가 높아 확장성, 이식성, 규모 확장성이 높음
- 모듈 간 통신이 빈번하게 발생 하여 성능이 떨어질 수 있음
- 프로세스 간 통신 발생을 최소화시키는 것이 중요 과제
- 응용 프로그램과 서버 간에 자료를 교환하려고 커널을 출입하는 문맥 교환 때문에 속도가 느림
- 커널 내부에서 발생 지연이 적고 예측 가능하여 실시간 시스템에 활용
- 대표적인 운영체제 중 마이크로 커널 구조를 전적으로 선택한 것은 없지만, 모듈화된 구성 요소는 포함

3. 마이크로 커널 구조 운영체제

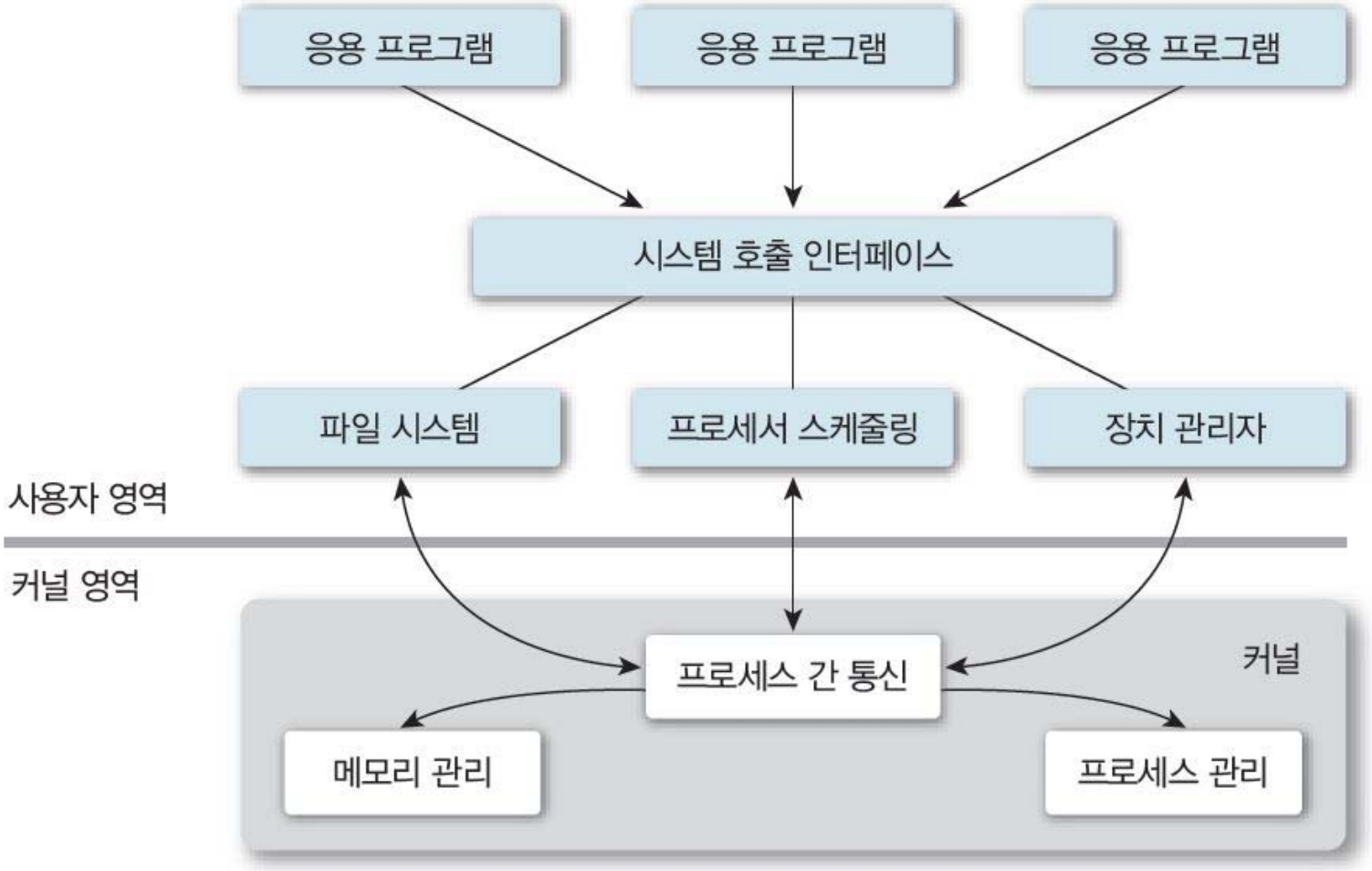


그림 2-23 마이크로 커널 구조 예

네트워크 운영체제와 분산 운영체제

네트워크 운영체제

- 네트워크에 있는 다른 컴퓨터의 자원에 접근 가능
- 네트워크 파일 시스템

분산 운영체제

- 한 대 이상의 컴퓨터에 있는 자원을 관리하는 특별한 운영체제
- 구현 복잡
- 프로세스가 공유 데이터에 접근하기 때문에 복잡한 알고리즘 요구