Chapter 01

운영체제의 개요

Contents

학습목표

- 운영체제의 필요성과 정의, 역할, 목표를 이해한다.
- 초창기부터 현재까지 운영체제의 발전 과정을 살펴본다.
- 현대의 컴퓨팅 환경을 파악한다.

1-1 일상생활 속의 운영체제

운영체제(OS, Operating System)

- 일반 컴퓨터, 노트북, 스마트폰의 전원을 켜면 가장 먼저 만나게 되는 소프트웨어
- [예] PC 운영체제(윈도우, Mac OS, 유닉스, 리눅스 등)
  모바일 운영체제(iOS, 안드로이드 등)

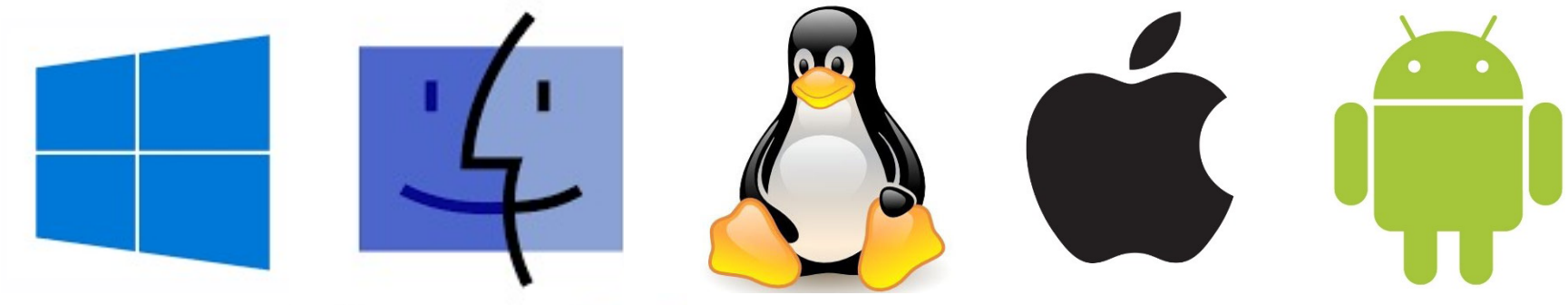


그림 1-1 다양한 운영체제(왼쪽부터 윈도우, Mac OS, 리눅스, iOS, 안드로이드)

1-1 일상생활 속의 운영체제

임베디드 운영체제

- CPU의 성능이 낮고 메모리 크기도 작은 시스템에 내장하도록 만든 운영체제
- 임베디드 운영체제가 있는 기계는 기능을 계속 향상할 수 있음

그림 1-2 임베디드 운영체제를 사용하는 스마트 시계와 스마트 TV

1-2 운영체제의 필요성

표 1-1 운영체제와 관련된 질문과 답의 요약

질문	답
컴퓨터는 운영체제가 없어도 작동하는가?	컴퓨터는 운영체제가 없어도 작동하지만 기능에 제약이 따른다.
운영체제가 있는 기계와 없는 기계는 어떤 차이가 있는가?	운영체제가 있는 기계는 다양한 응용 프로그램을 설치하여 사용할 수 있고 성능 향상을 위한 새로운 기능을 쉽게 추가할 수 있다.
운영체제는 성능을 향상하는 데에만 필요한가?	운영체제는 컴퓨터의 성능을 향상할 뿐 아니라 자원을 관리하고 사용자에게 편리한 인터페이스 환경을 제공한다.
운영체제는 자원을 어떻게 관리하는가?	운영체제는 사용자가 직접 자원에 접근하는 것을 막음으로써 컴퓨터 자원을 보호한다.
사용자는 숨어 있는 자원을 어떻게 이용할 수 있는가?	운영체제가 제공하는 사용자 인터페이스와 하드웨어 인터페이스를 이용하여 자원에 접근한다.

1-3 운영체제의 정의

레스토랑에 비유한 운영체제

- 레스토랑에서 음식을 주문하면 웨이터가 그 음식을 주방에 알려주고 손님은 주문한 음식을 제공받음
- 손님이 멋대로 주방 기구를 만지거나 조리를 하면 주방이 난장판이 됨
- 손님이 주방에 들어가서 직접 조리하거나 주방 기구를 만지지 않는 것처럼 운영체제가 컴퓨터 자원을 직접 관리하고 그 결과만을 사용자에게 알려줌으로써 자원을 보호 함

그림 1-3 레스토랑에 비유한 운영체제

1-3 운영체제의 정의

운영체제의 정의

- 응용 프로그램이나 사용자에게 컴퓨터 자원을 사용할 수 있는 인터페이스를 제공하고 그 결과를 돌려주는 시스템 소프트웨어
- 응용 프로그램이나 사용자에게 모든 컴퓨터 자원을 숨기고 정해진 방법으로만 컴퓨터 자원을 사용할 수 있도록 제한

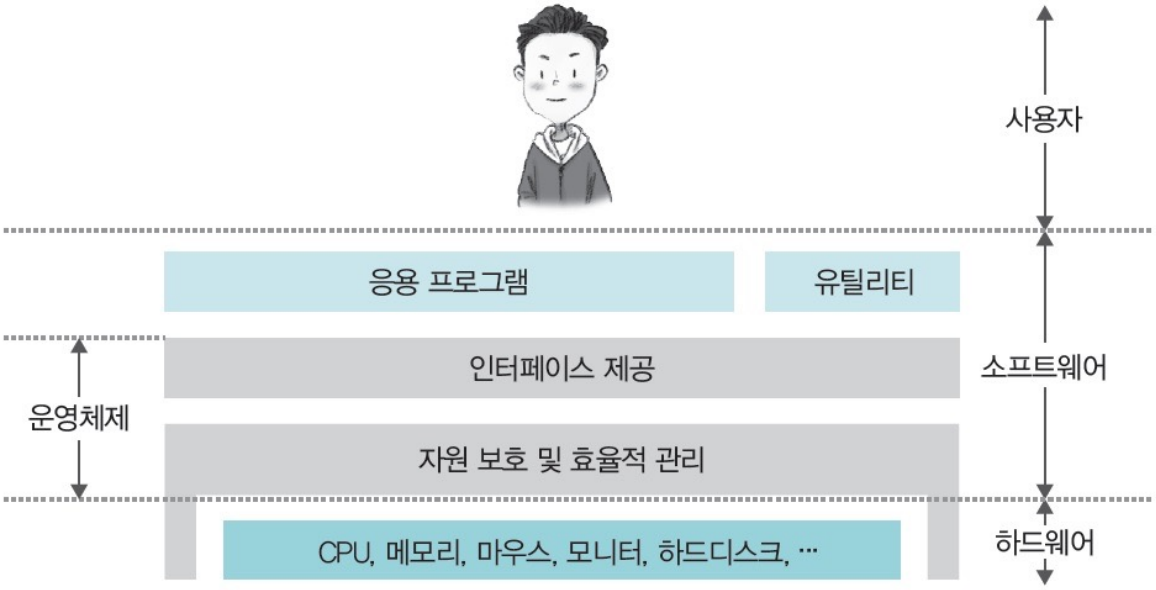


그림 1-4 운영체제의 정의

1-4 운영체제의 역할

자원 관리

- 컴퓨터 시스템의 자원을 응용 프로그램에 나누어주어 사용자가 원활하게 작업할 수 있도록 함
- 자원을 요청한 프로그램이 여러 개라면 적당한 순서로 자원을 배분하고 적절한 시점에 자원을 회수하여 다른 응용 프로그램에 나누어줌

자원 보호

- 비정상적인 작업으로부터 컴퓨터 자원을 보호

하드웨어 인터페이스 제공

- 사용자가 복잡한 과정 없이 다양한 장치를 사용할 수 있도록 해주는 하드웨어 인터페이스 제공
- CPU, 메모리, 키보드, 마우스와 같은 다양한 하드웨어를 일관된 방법으로 사용할 수 있도록 지원

사용자 인터페이스 제공

- 사용자가 운영체제를 편리하게 사용하도록 지원(예: 윈도우의 그래픽 사용자 인터페이스(GUI))

1-5 운영체제의 목표

운영체제의 목표

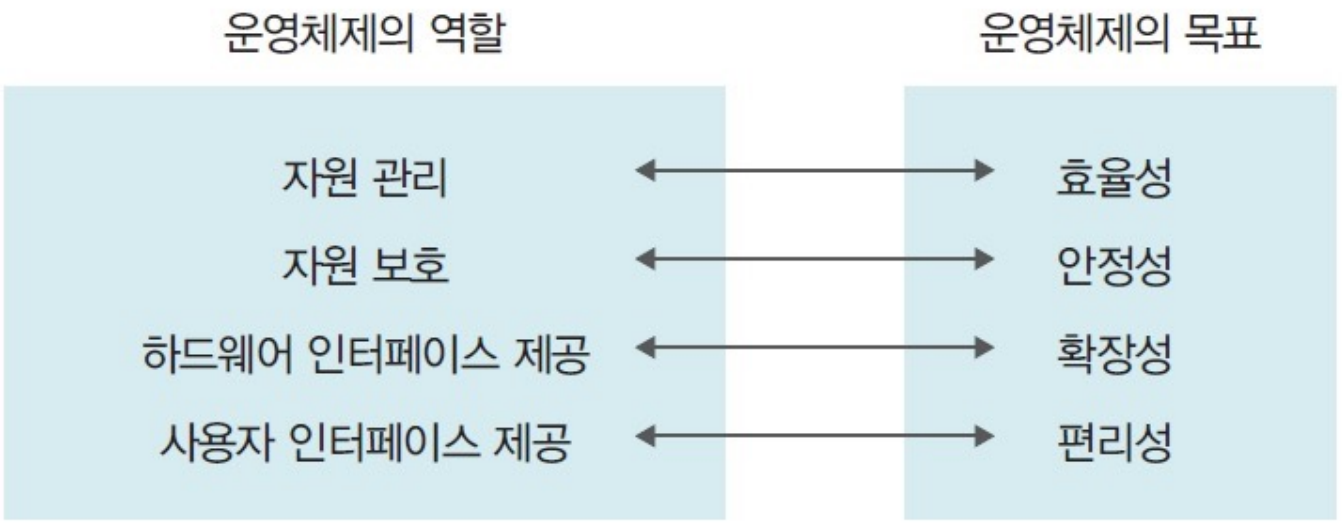


그림 1-6 운영체제의 역할과 목표

1-5 운영체제의 목표

효율성

- 자원을 효율적으로 관리하는 것
- 같은 자원을 사용하여 더 많은 작업량을 처리하거나, 같은 작업량을 처리하는 데 보다 적은 자원을 사용하는 것

안정성

- 작업을 안정적으로 처리하는 것
- 사용자와 응용 프로그램의 안전 문제와 하드웨어적인 보안 문제 처리
- 시스템에 문제가 발생했을 때 이전으로 복구하는 결함 포용 기능 수행

확장성

- 다양한 시스템 자원을 컴퓨터에 추가하거나 제거하기 편리한 것

편리성

- 사용자가 편리하게 작업할 수 있는 환경을 제공하는 것

2-1 운영체제 역사

주변 기기의 발전을 중심으로 살펴본 운영체제의 역사

표 1-2 운영체제의 역사

구분	시기	주요 기술	특징
0기	1940년대	없음	• 진공관(0과 1) 사용
1기	1950년대	카드 리더, 라인 프린터	• 일괄 작업 시스템 • 운영체제의 등장
2기	1960년대 초반	키보드, 모니터	• 대화형 시스템
3기	1960년대 후반	C 언어	• 다중 프로그래밍 기술 개발 • 시분할 시스템
4기	1970년대	PC	• 개인용 컴퓨터의 등장 • 분산 시스템
5기	1990년대	웹	• 클라이언트/서버 시스템
6기	2000년대	스마트폰	• P2P 시스템(메신저, 파일 공유) • 그리드 컴퓨팅 • 클라우드 컴퓨팅 • 사물 인터넷

2-2 초창기 컴퓨터(1940년대)

에니악

- 백열전구 같은 모양의 진공관이라는 소자를 사용하여 진공관이 켜지면 1, 꺼지면 0이라고 판단
- 전선을 연결하여 논리회로를 구성하는 '하드와이어링' 방식으로 동작
- 운영체제가 없음

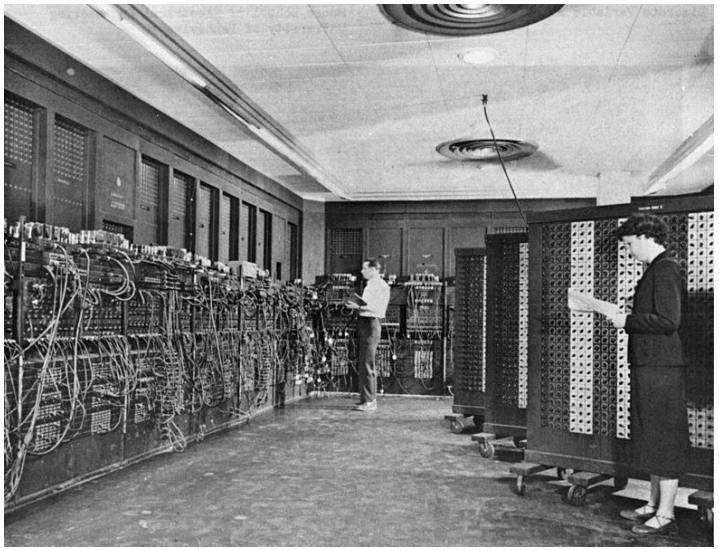


그림 1-7 에니악

2-3 일괄 작업 시스템(1950년대)

천공카드 시스템

- 천공카드 리더를 입력장치로, 라인 프린터를 출력장치로 사용
- 프로그램을 구성한 후 카드에 구멍을 뚫어 컴퓨터에 입력하면 프로그램이 실행되는 구조로서 프로그램의 실행 결과가 라인 프린터를 통해 출력

그림 1-8 홀러리스의 천공카드 시스템

2-3 일괄 작업 시스템(1950년대)

일괄 작업 시스템

- 천공카드리더기(입력)와 라인프린터(출력) 사용 : 모든 작업을 한꺼번에 처리하고 프로그램 실행 중간에 사용자가 데이터를 입력하거나 수정하는 것이 불가능한 시스템
- 운영체제 사용(메인메모리가 운영체제의 상주 영역과 사용자의 영역으로 나뉨)

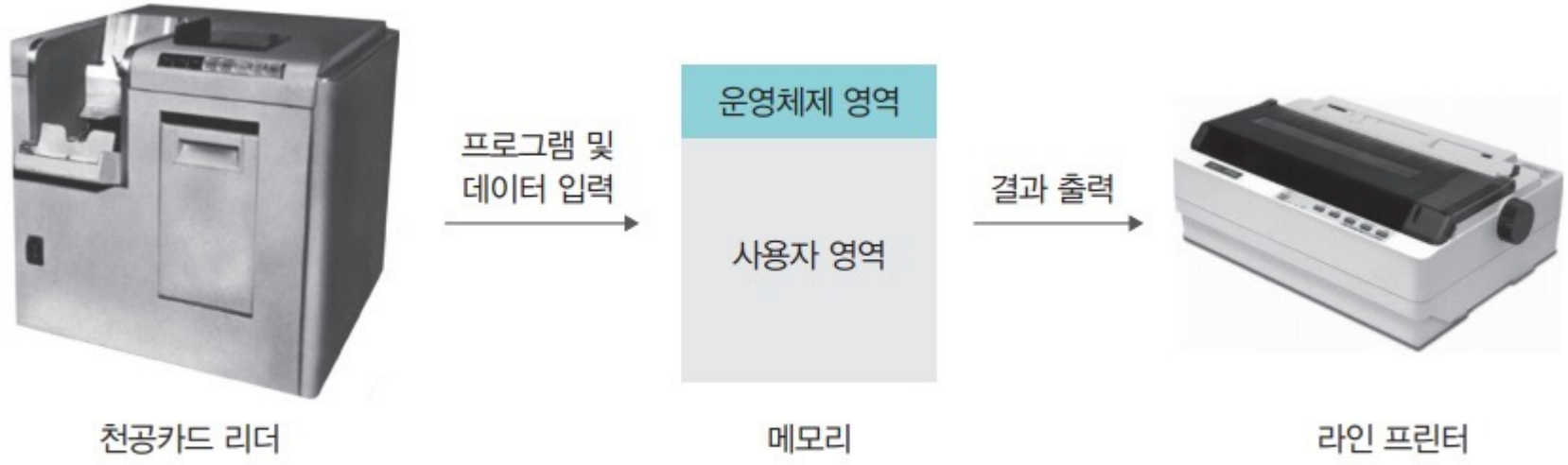


그림 1-9 일괄 작업 시스템의 구성

2-4 대화형 시스템(1960년대 초반)

대화형 시스템

- 모니터와 키보드의 등장 : 프로그램이 진행되는 도중에 사용자로부터 입력을 받을 수 있어 입력값에 따라 작업의 흐름을 바꾸는 것이 가능한 시스템
- 대화형 시스템의 등장으로 문서 편집기, 게임과 같은 다양한 종류의 응용 프로그램을 만들 수 있게 됨

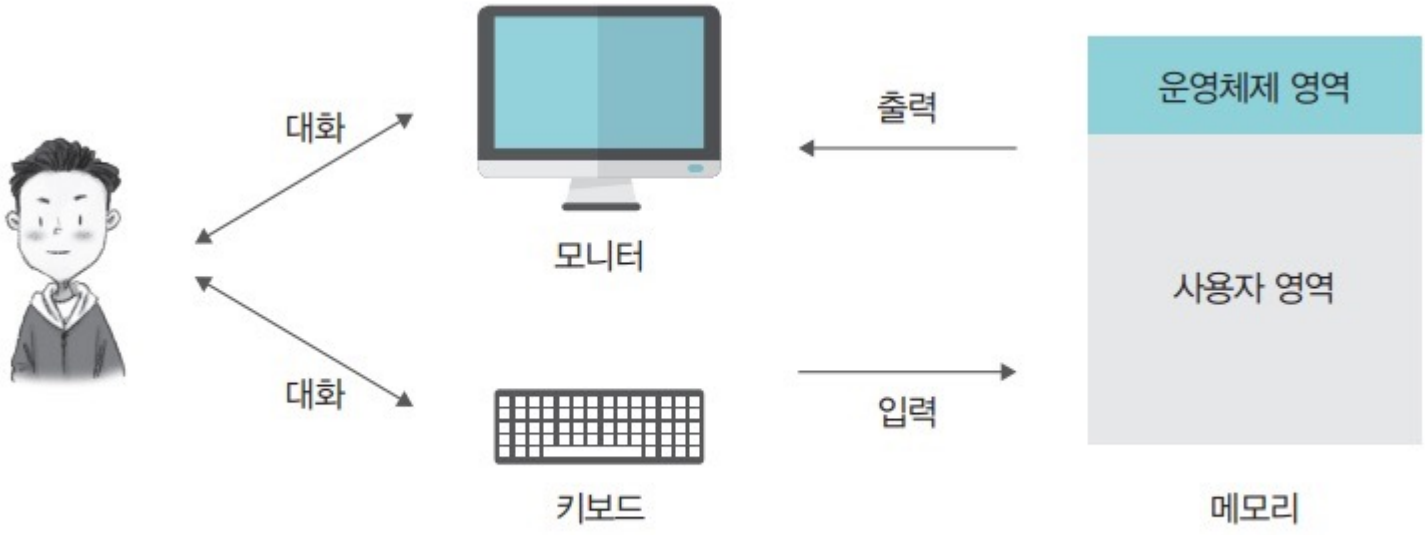
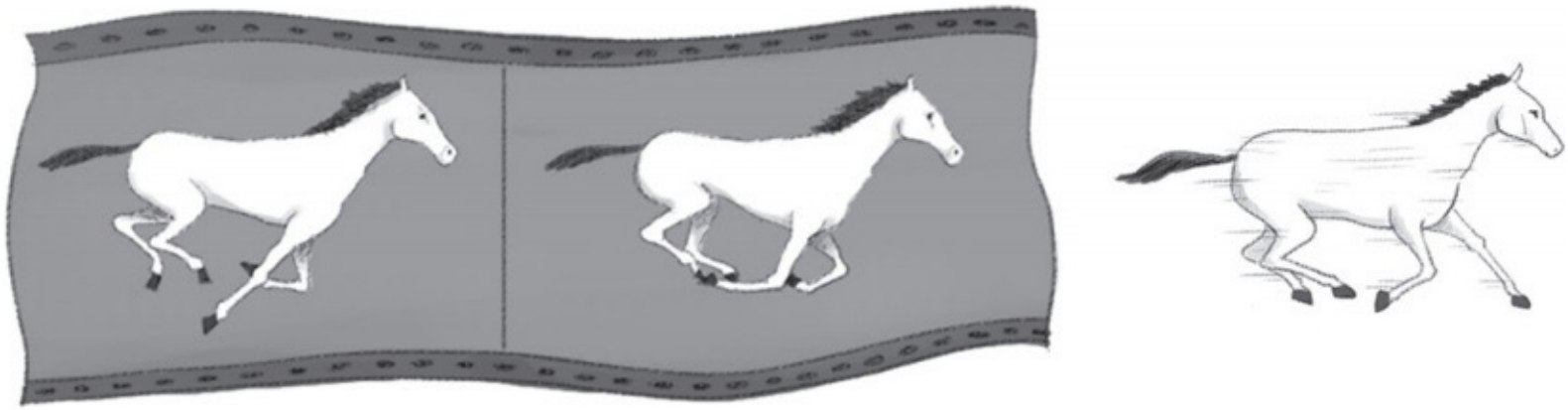


그림 1-10 대화형 시스템의 구성

2-5 시분할 시스템(1960년대 후반)

다중 프로그래밍

- 하나의 CPU로 여러 작업을 동시에 실행하는 기술
- 한 번에 하나의 작업만 가능한 일괄 작업 시스템에 비해 효율성이 뛰어남
- 시간을 분할하는 방법 때문에 여러 작업이 동시에 실행되는 것처럼 보임



(a) 정지 영상 (b) 동영상

그림 1-11 시분할 시스템의 작동 원리

2-5 시분할 시스템(1960년대 후반)

다중 프로그래밍

- 다중 프로그래밍 시스템에서는 CPU 사용 시간을 아주 잘게 쪼개어 여러 작업에 나누어줌

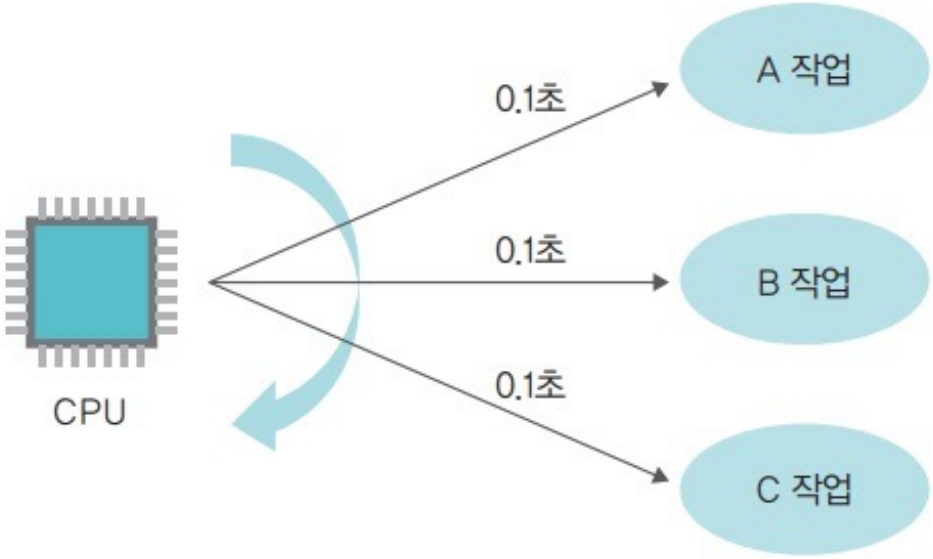


그림 1-12 다중 프로그래밍 시스템의 시간 분배

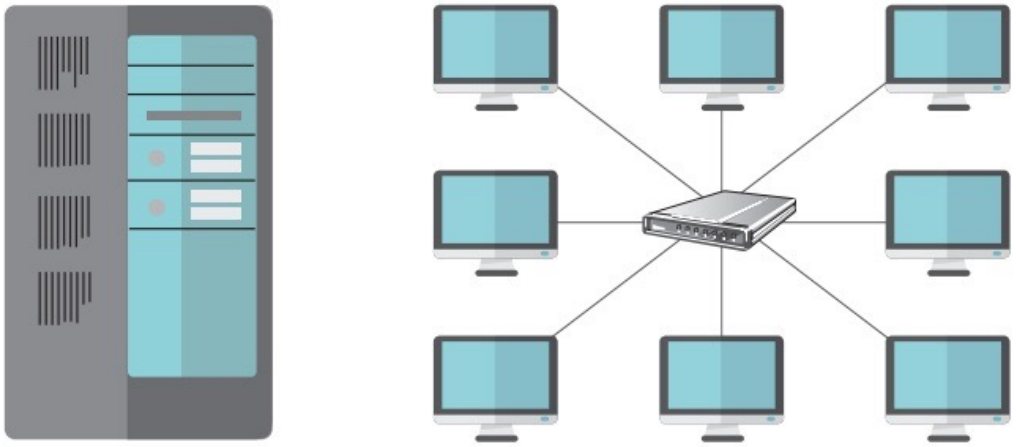
시분할 시스템

- CPU 사용 시간을 잘게 쪼개어 작업들에 나누어줌으로써 모든 작업이 동시에 처리되는 것처럼 보임
- 잘게 나뉜 시간 한 조각을 타임 슬라이스 또는 타임 퀀텀이라고 함
- 오늘날의 컴퓨터에는 대부분 시분할 시스템이 사용됨

2-6 분산 시스템(1970년대 후반)

분산 시스템

- 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

2-7 클라이언트/서버 시스템(1990년대~현재)

클라이언트/서버 시스템

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

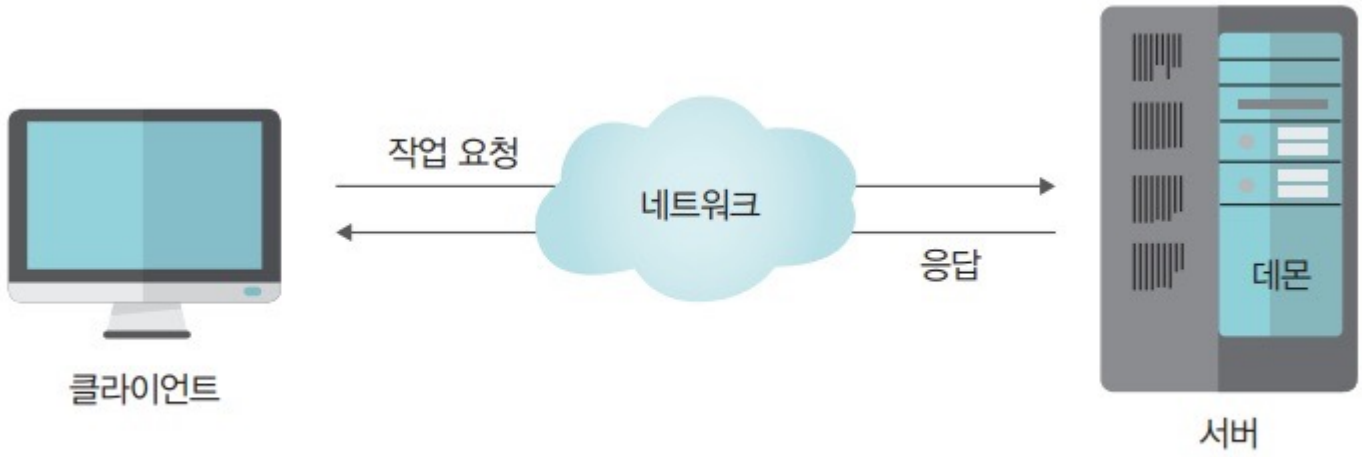
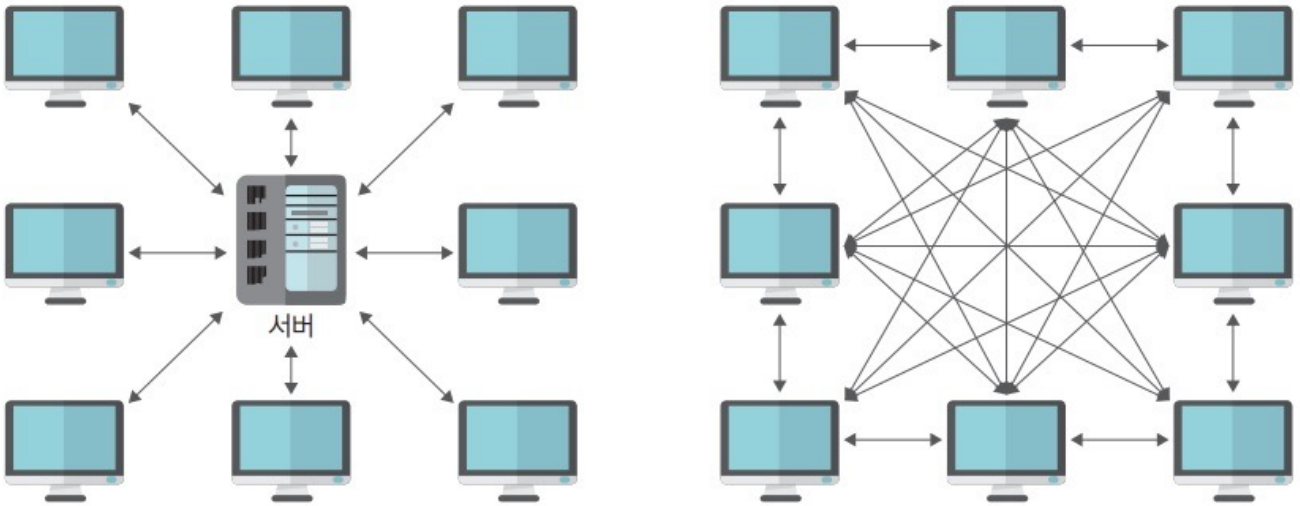


그림 1-14 클라이언트/서버 구조

2-8 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

2-8 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

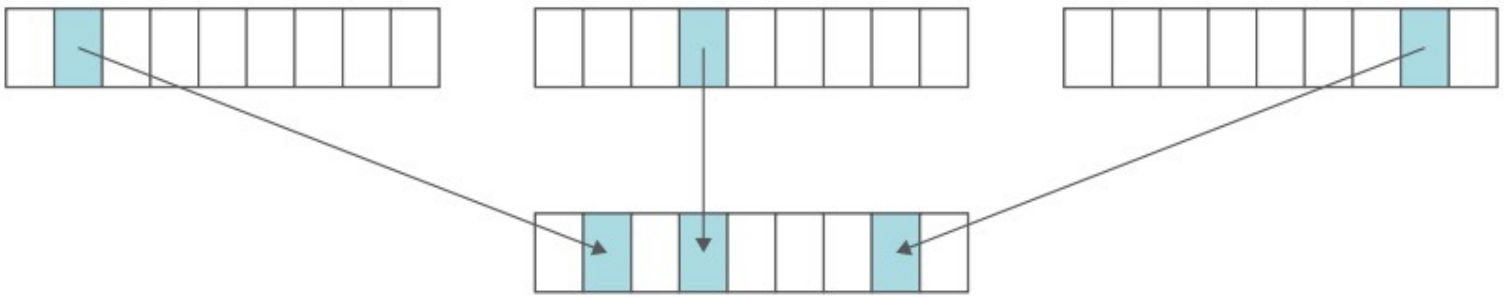


그림 1-17 대용량 파일 공유 P2P 시스템

2-9 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

2-9 기타 컴퓨팅 환경(2000년대 초반~현재)

클라우드 컴퓨팅

- 언제 어디서나 응용 프로그램과 데이터를 자유롭게 사용할 수 있는 컴퓨팅 환경으로 그리드 컴퓨팅과 SaaS를 합쳐 놓은 형태
- 클라우드 컴퓨팅은 PC, 핸드폰, 스마트 기기 등을 통하여 인터넷에 접속하고, 다양한 작업을 수행하며, 데이터 또한 기기들 사이에서 자유롭게 이동이 가능한 컴퓨팅 환경
- 하드웨어를 포함한 시스템이 구름에 가려진 것처럼 사용자에게 보이지 않는 컴퓨팅 환경이라는 의미

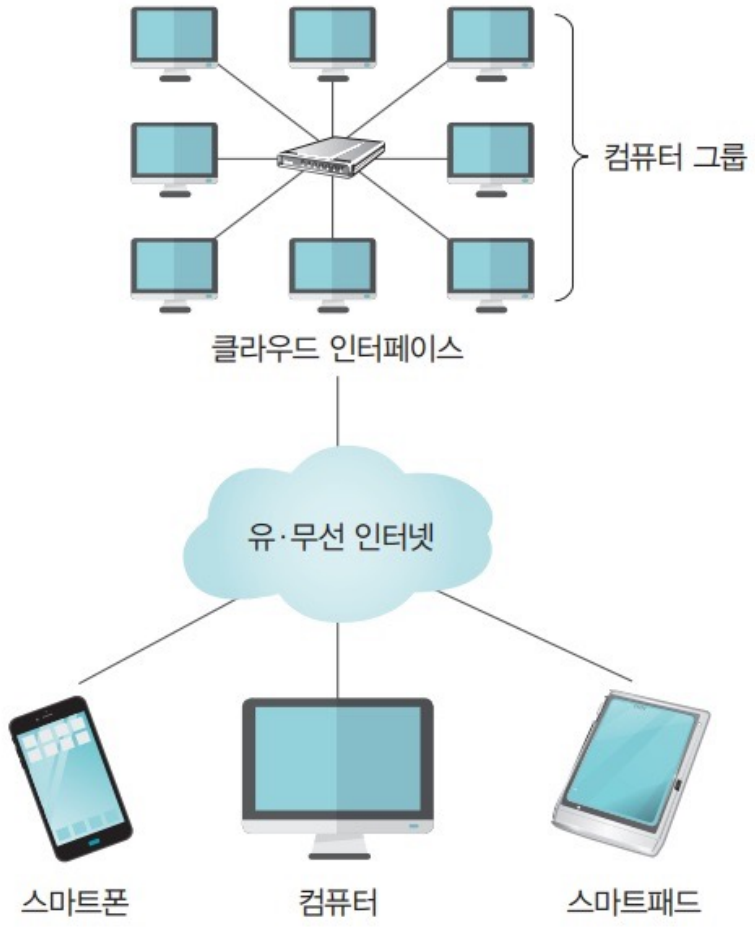


그림 1-18 클라우드 컴퓨팅 환경

2-9 기타 컴퓨팅 환경(2000년대 초반~현재)

사물 인터넷(Internet of Thing; IoT)

- 사물에 센서와 통신 기능을 내장하여 인터넷에 연결하는 기술
- 예를 들어, 전철이나 버스의 도착 예정시간을 표시한다던가, 각종 전자제품을 스마트 폰으로 제어 하거나 알림 문자를 받는 서비스가 있음.
- 컨넥트 카, 에네지를 제어하는 스마트 그리드, 공공기물을 관리하는 스마트 시티, 사물인터넷을 응용한 재난 방지 시스템 등 다양한 분야에 적용
- 인터넷으로 연결된 사물들이 데이터를 주고받아 스스로 분석하고 학습한 정보를 사용자에게 제공하거나 새로운 서비스를 창출

그림 1-19 사물 인터넷

3-1 커널과 인터페이스

커널

- 프로세스 관리, 메모리 관리, 저장장치 관리와 같은 운영체제의 핵심적인 기능을 모아놓은 것

인터페이스

- 커널에 사용자의 명령을 전달하고 실행 결과를 사용자에게 알려주는 역할
- 그래픽을 사용한 인터페이스를 GUI(graphical User Interface)라 부름

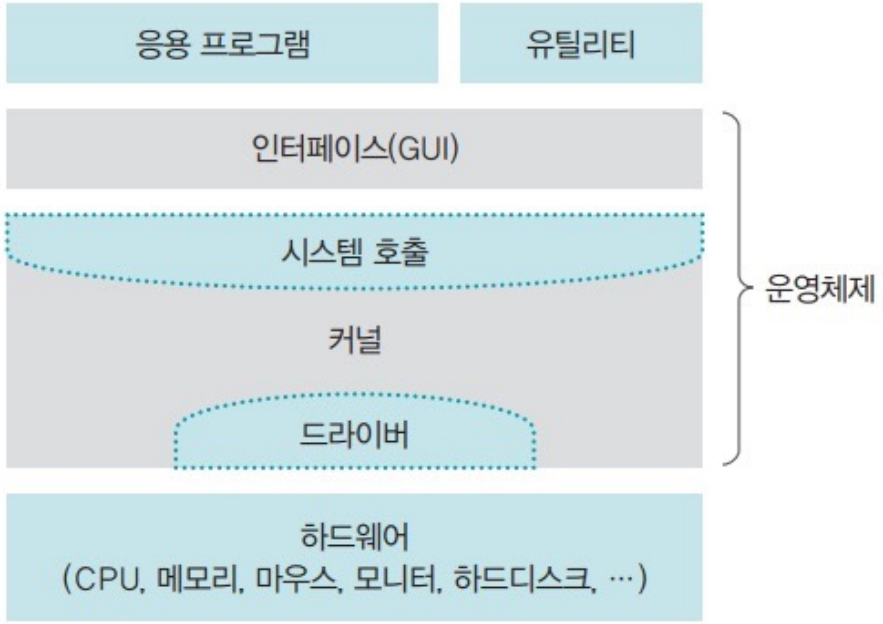


그림 1-20 컴퓨터 시스템의 구조

3-2 시스템 호출과 디바이스 드라이버

시스템 호출

- 커널이 자신을 보호하기 위해 만든 인터페이스
- 커널은 사용자나 응용 프로그램으로부터 컴퓨터 자원을 보호하기 위해 자원에 직접 접근하는 것을 차단

(a) 커피를 직접 만드는 경우 (b) 다른 사람에게 커피를 만들어 달라고 부탁하는 경우

그림 1-21 **커피를 직접 만드는 경우와 다른 사람에게 만들어달라고 부탁하는 경우**

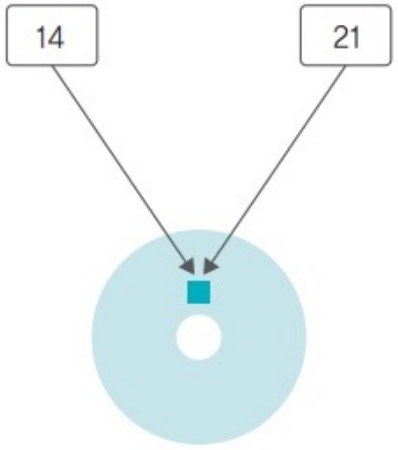
3-2 시스템 호출과 디바이스 드라이버

직접 접근

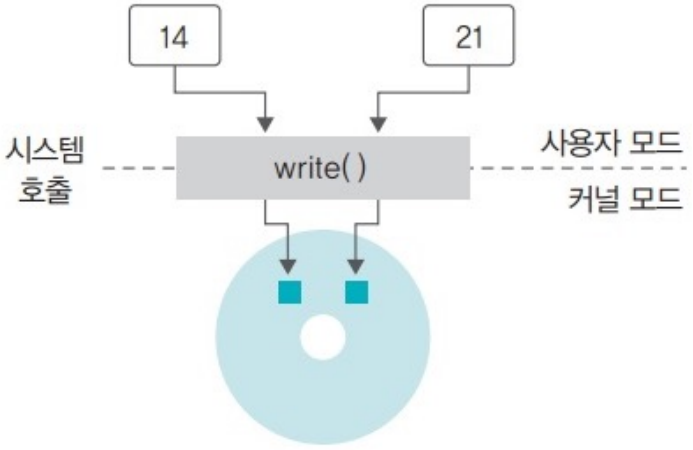
- 두 응용 프로그램이 자기 마음에 드는 위치에 데이터를 저장하려고 함
- 다른 사람의 데이터를 지울 수도 있고 내 데이터가 다른 사람에 의해 지워질 수도 있음

시스템 호출을 통한 접근

- 응용 프로그램이 직접 하드디스크에 데이터를 저장하지 않고 커널이 제공하는 write( ) 함수를 사용하여 데이터를 저장해달라고 요청
- 커널이 데이터를 가져오거나 저장하는 것을 전적으로 책임지기 때문에 컴퓨터 자원 관리가 수월



(a) 직접 접근



(b) 시스템 호출을 통한 접근

3-2 시스템 호출과 디바이스 드라이버

시스템 호출 정리

- 시스템 호출은 커널이 제공하는 시스템 자원의 사용과 연관된 함수
- 응용 프로그램이 하드웨어 자원에 접근하거나 운영체제가 제공하는 서비스를 이용하려 할 때는 시스템 호출을 사용해야 함
- 운영체제는 커널이 제공하는 서비스를 시스템 호출로 제한하고 다른 방법으로 커널에 들어오지 못하게 막음으로써 컴퓨터 자원을 보호
- 시스템 호출은 커널이 제공하는 서비스를 이용하기 위한 인터페이스이며, 사용자가 자발적으로 커널 영역에 진입할 수 있는 유일한 수단임

3-2 시스템 호출과 디바이스 드라이버

드라이버

- 커널과 하드웨어의 인터페이스 담당하며 디바이스 드라이버라고도 불림
- 마우스와 같이 간단한 제품은 드라이버를 커널이 가지고 있으나, 그래픽카드와 같이 복잡한 하드웨어의 경우 제작자가 드라이버를 제공 함

그림 1-23 운영체제의 구조

3-3 커널의 구성

커널

표 1-3 커널이 하는 일

핵심 기능	설명
프로세스 관리	프로세스에 CPU를 배분하고 작업에 필요한 제반 환경을 제공한다.
메모리 관리	프로세스에 작업 공간을 배치하고 실제 메모리보다 큰 가상공간을 제공한다.
파일 시스템 관리	데이터를 저장하고 접근할 수 있는 인터페이스를 제공한다.
입출력 관리	필요한 입력과 출력 서비스를 제공한다.
프로세스 간 통신 관리	공동 작업을 위한 각 프로세스 간 통신 환경을 지원한다.

3-3 커널의 구성

단일형 구조 커널

- 초창기의 운영체제 구조
- 커널의 핵심 기능을 구현하는 모듈들이 구분 없이 하나로 구성

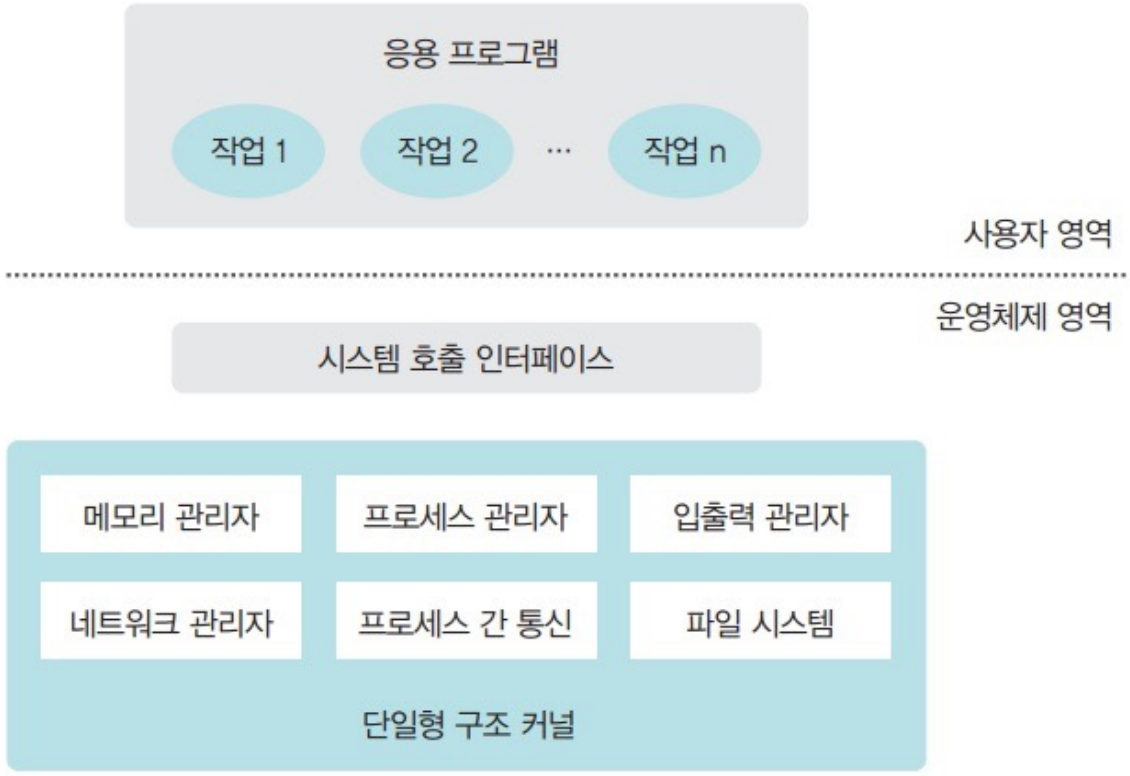


그림 1-24 **단일형 구조 커널**

3-3 커널의 구성

단일형 구조 커널

- 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

3-3 커널의 구성

계층형 구조 커널

- 비슷한 기능을 가진 모듈을 묶어서 하나의 계층으로 만들고 계층 간의 통신을 통해 운영체제를 구현하는 방식

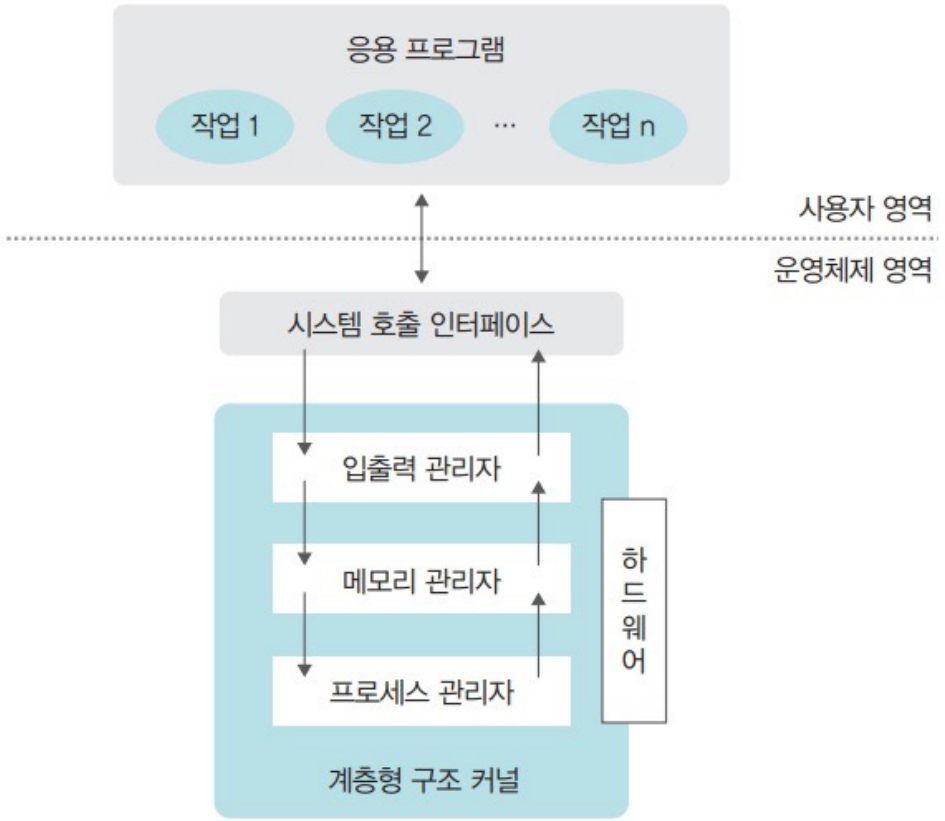


그림 1-25 계층형 구조 커널

3-3 커널의 구성

마이크로 구조 커널

- 프로세스 관리, 메모리 관리, 프로세스 간 통신 관리 등 가장 기본적인 기능만 제공
- 커널의 각 모듈은 세분화되어 존재하고 모듈 간의 정보 교환은 프로세스 간 통신을 이용하여 이루어짐

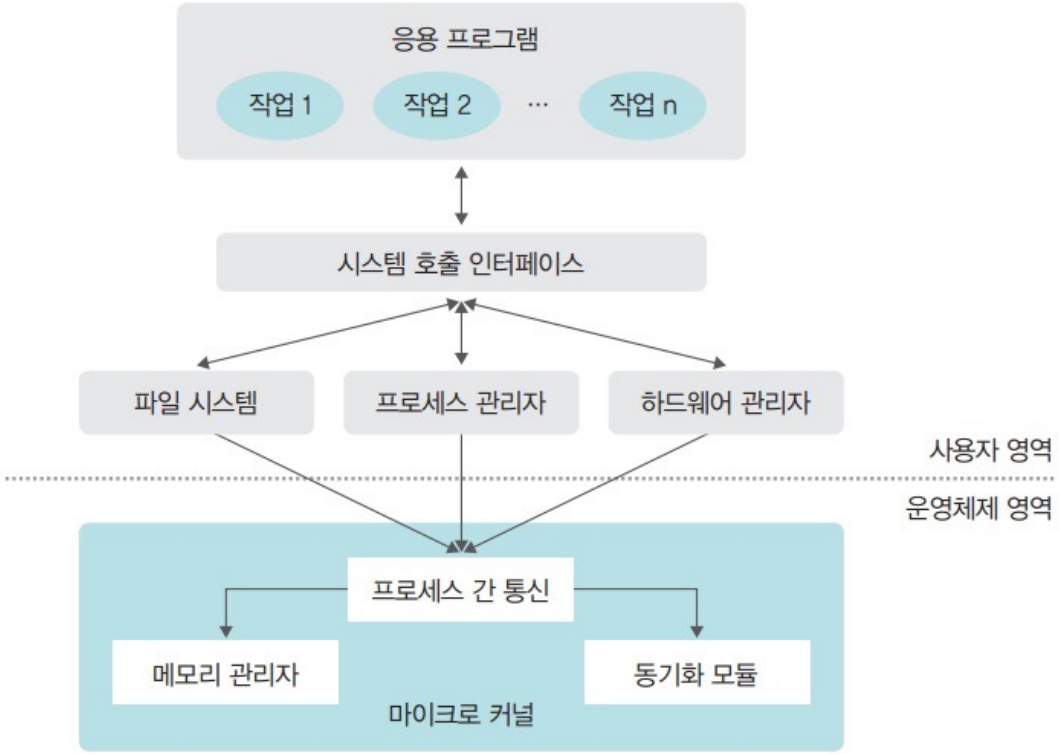


그림 1-26 마이크로 구조 커널

3-4 가상머신

가상머신

- 운영체제와 응용 프로그램 사이에서 작동하는 프로그램
- 가상머신을 설치하면 응용 프로그램이 모두 동일한 환경에서 작동하는 것처럼 보임
- 자바는 유닉스와 윈도우에서 작동하는 다양한 가상머신을 만들어 배포하는데 이를 자바 가상머신Java Virtual Machine, JVM이라고 함

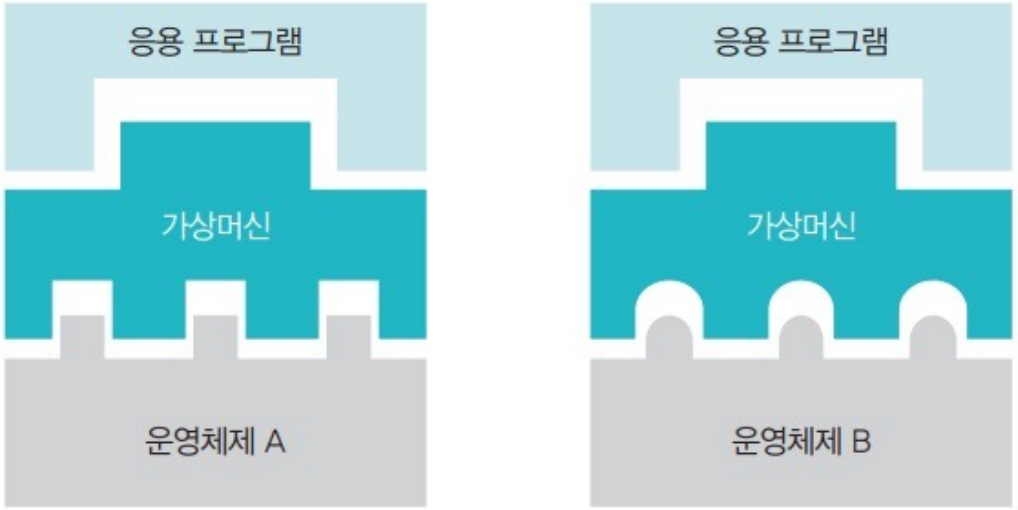


그림 1-27 가상머신의 개념

심화학습 4-1 유닉스와 리눅스

유닉스의 개발과 확산

- 1969년 AT&T의 연구원으로서 멀틱스 프로젝트에 참가 중이던 켄 톰프슨은 사무실에서 안 쓰던 PDP-7 컴퓨터에 멀틱스와 비슷한 개념의 운영체제를 구현하려 함
- 멀틱스 프로젝트가 잘 진행되지 않던 차에 톰프슨의 행동에 흥미를 느낀 데니스 리치와 피터 뉴만도 여기에 함께하게 되었고, 이 운영체제의 이름을 골치 아픈 멀틱스 대신 단순하다는 의미의 '유닉스'로 지음
- 유닉스는 이식하기 쉬웠던 탓에 인기를 얻게 됨
- 개발 후 소스코드가 공개되어 계속 다른 기종의 컴퓨터에 이식되었으며, 여러 기업과 대학에서 이를 이용한 연구가 진행되어 다양한 기능이 추가됨

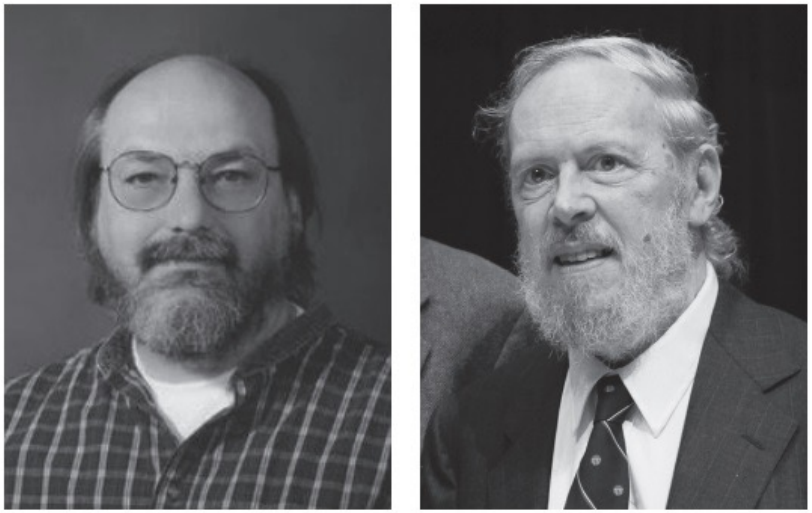


그림 1-28 켄 톰프슨(왼쪽)과 데니스 리치(오른쪽)

4-1 유닉스와 리눅스

리눅스의 개발

- 1991년에 리누스 토르발스가 PC에서 동작하는 유닉스 호환 커널을 작성하여 GPL로 배포하고 이어서 소스코드도 공개한 것이 리눅스의 시작

그림 1-29 리누스 토르발스

4-2 매킨토시와 스티브 잡스

애플 II의 등장

- 1976년 스티브 잡스는 스티브 워즈니악, 로널드 웨인과 함께 애플을 창업하고, 1977년 애플 II라는 개인용 컴퓨터를 대중화함

그림 1-30 애플 II

4-2 매킨토시와 스티브 잡스

스티브 잡스의 업적

- 기술을 발전시키는 것보다 사용자에게 편리함을 제공하는 사용자 인터페이스와 같은 기술 개발에 집중
- 사용자 인터페이스UI보다 더 사용자 친화적이고 사용자 경험에 중심을 두는 UXUser eXperience 기술 개발에 집중
- 2001년 아이팟을 출시하여 음악 산업 전체를 뒤집어엎음

그림 1-31 아이팟

4-2 매킨토시와 스티브 잡스

스티브 잡스의 업적

- 아이팟 다음으로 아이폰을 내놓아 세상을 놀라게 함
- 사용자들이 애플 제품에 열광하는 이유는 다른 제품에서는 경험할 수 없는 독특하고 편리한 인터페이스에 있음

그림 1-32 아이폰을 들고 있는 스티브 잡스

4-3 윈도우 운영체제

윈도우의 출시

- 마이크로소프트는 애플 Mac OS의 그래픽 사용자 인터페이스에 자극을 받아 윈도우 운영체제를 출시하고 계속 업그레이드하여 현재는 버전 10까지 나와 있음

그림 1-33 윈도우의 버전별 로고

4-4 모바일 운영체제

스마트폰의 등장

- 마이크로소프트의 번영은 스마트폰의 보급으로 한풀 꺾임
- 애플의 아이폰이 스마트폰 시장을 장악
- 다른 회사들도 그에 대항할 스마트폰을 개발하는 데 주력
- 구글이 모바일용 운영체제인 안드로이드를 개발
- 현재 아이폰을 제외한 대부분의 스마트폰은 안드로이드 운영체제로 사용하고 있음

4-4 모바일 운영체제

안드로이드의 특징

- GNU의 리눅스 커널을 사용하여 제작되었기 때문에 GPL을 따름
- 누구나 공짜로 사용할 수 있고, 새로운 버전과 동시에 소스코드도 공개하기 때문에 누구나 수정·배포할 수 있음
- 스마트폰 제조사들은 대부분 안드로이드의 소스코드를 자사 제품에 맞게 수정하여 무료로 배포하고 있음

그림 1-34 구글과 안드로이드

Thank You