

# Chapter 2

## Operating System Overview

# Contents

---

- 2.1 운영체제의 목적 및 기능
- 2.2 운영체제의 발전
- 2.3 주요 성과
- 2.4 최근 운영체제로의 발전
- 2.5 결함허용
- 2.6 멀티프로세서와 멀티코어를 위한 운영체제 설계 사항
- 2.7 Microsoft Windows 개요
- 2.8 전통적인 UNIX 시스템
- 2.9 최근의 UNIX 시스템
- 2.10 Linux
- 2.11 Android

## 2.1 운영체제의 목적과 기능

---

### ■ 운영체제란?

- ✓ 응용 프로그램의 실행을 제어하는 프로그램
- ✓ 응용과 하드웨어 간의 인터페이스

### ■ 운영체제의 목적

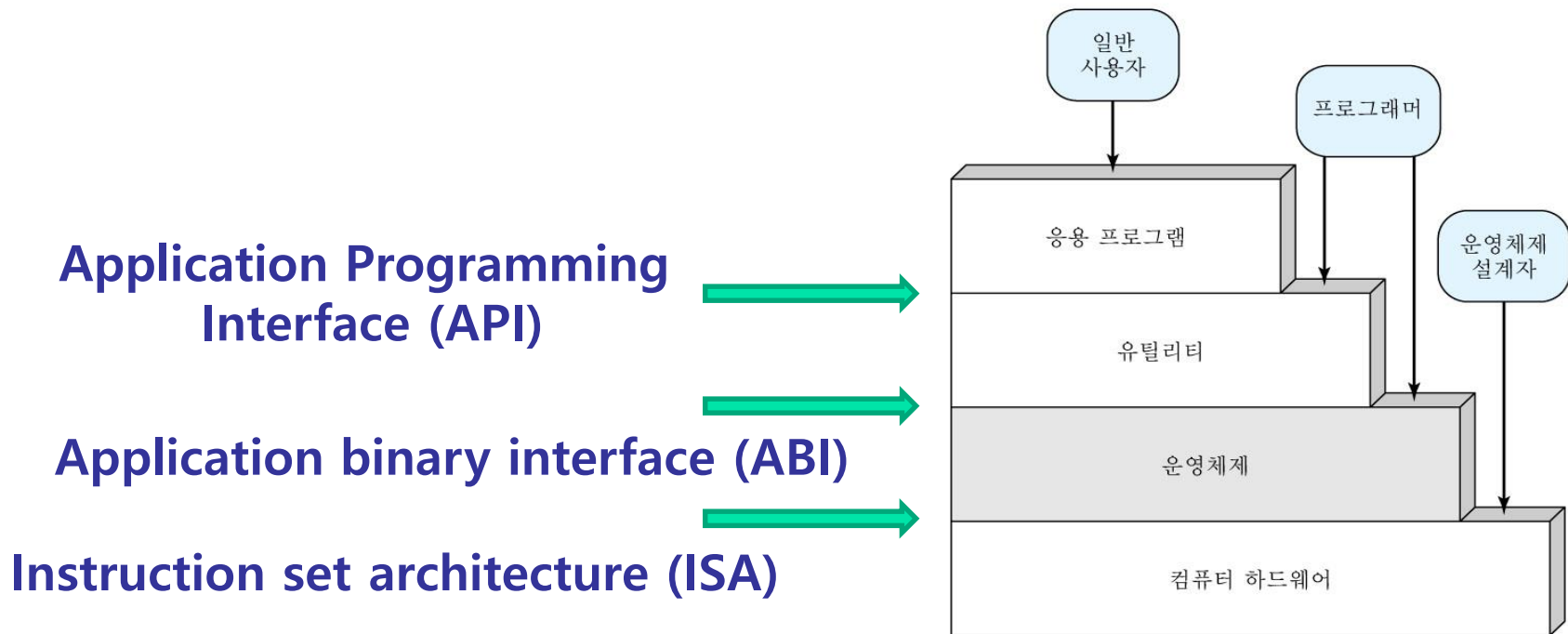
- ✓ 편리성 (convenience)
  - 컴퓨터를 편리하게 사용
- ✓ 효율성 (efficiency)
  - 컴퓨터 시스템 자원을 효율적인 방법으로 사용
- ✓ 발전성 (ability to evolve)
  - 효과적인 개발과 검사 그리고 새로운 시스템 기능도입을 다른 서비스를 방해하지 않고 수용해야 함

# 운영체제

---

- 컴퓨터 사용자와 하드웨어 사이에 중재자(intermediary) 역할을 하는 프로그램
- 사용자/컴퓨터 인터페이스로서의 운영체제(사용자관점)
  - ✓ 사용자가 컴퓨터를 손쉽게 사용할 수 있게 해주고, 컴퓨팅 환경을 제공해준다.
  - ✓ 프로그램 개발, 프로그램 실행, 입출력장치 접근, 파일접근, 시스템접근, 에러발견 및 처리, 어카운팅,
- 자원관리자로서의 운영체제(시스템관점)
  - 시스템 자원을 효율적으로 관리하는 자원 관리자

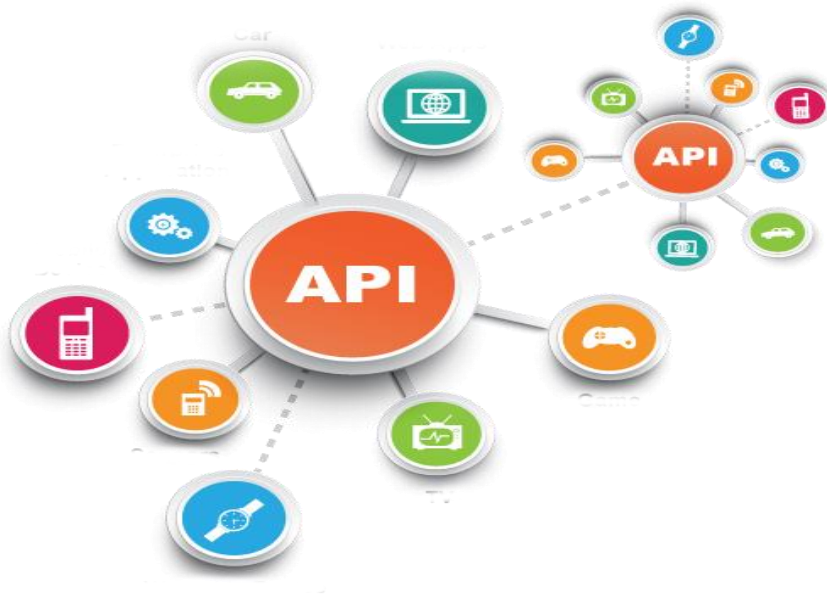
# 컴퓨터 시스템의 계층 구조와 관점



- Operating System as a User/Computer Interface
  - ✓ Program development
  - ✓ Program execution
  - ✓ Access to I/O devices
  - ✓ Controlled access to files
  - ✓ System access
  - ✓ Error detection and response
    - Internal and external hardware errors: Memory error, Device failure
    - Software errors: Arithmetic overflow, Access forbidden memory locations
    - Inability of the Operating system to grant requests of applications
    - Stop, retry, report
  - ✓ Accounting
    - Collect usage statistics
    - Monitor performance
    - Used for future enhancements and tuning
    - Used for billing purposes

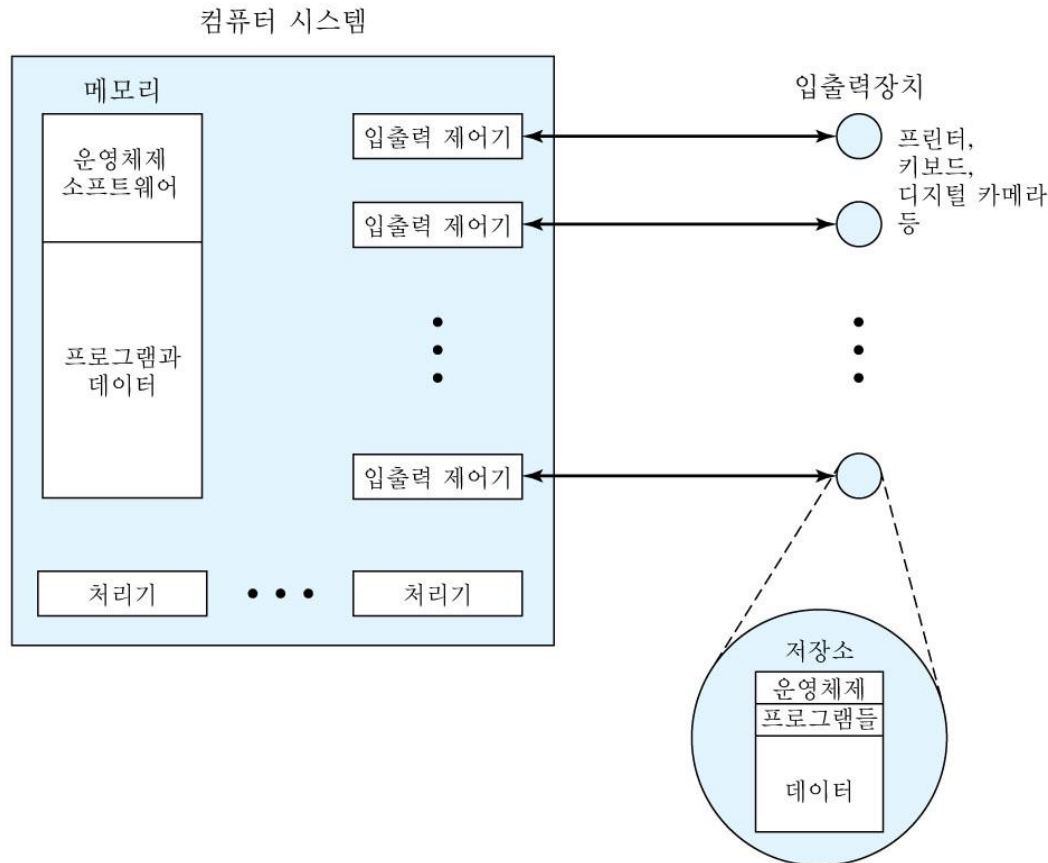
## ■ Operating System as a User/Computer Interface

- ✓ 인스트럭션 셋 구조 (ISA)
  - 컴퓨터가 따르고 있는 기계명령어 집합
- ✓ 응용 실행파일 인터페이스 (ABI)
  - 이기종간 호환성 지원
- ✓ 응용 프로그래밍 인터페이스 (API)
  - 서로 다른 플랫폼(OS)간 호환성 지원



# 자원관리자로서의 운영체제

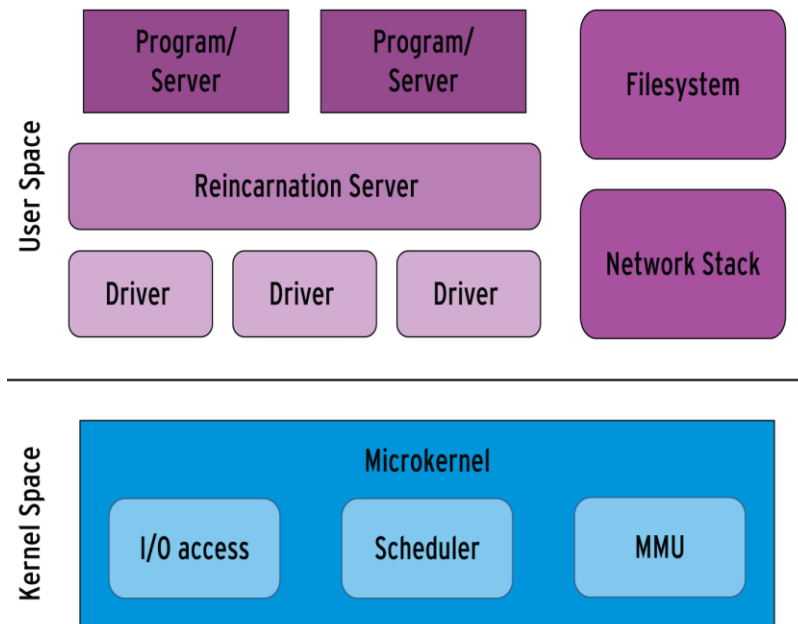
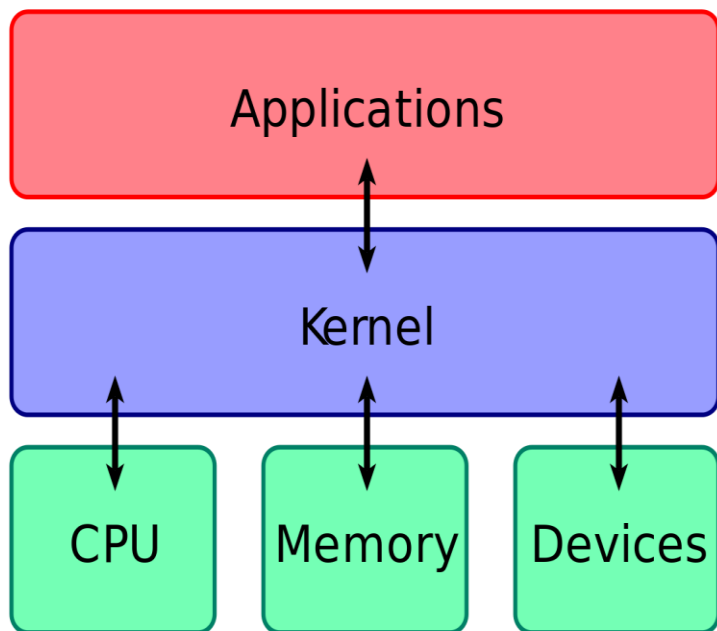
- 일반적인 컴퓨터 소프트웨어와 동일하게 기능
- 처리기에 의해 실행되는 프로그램 (또는 일련의 프로그램들)
- 수시로 응용에게 제어를 양도해가며 실행되고, 특정 CPU 기능을 통해 응용으로부터 제어를 넘겨 받는다





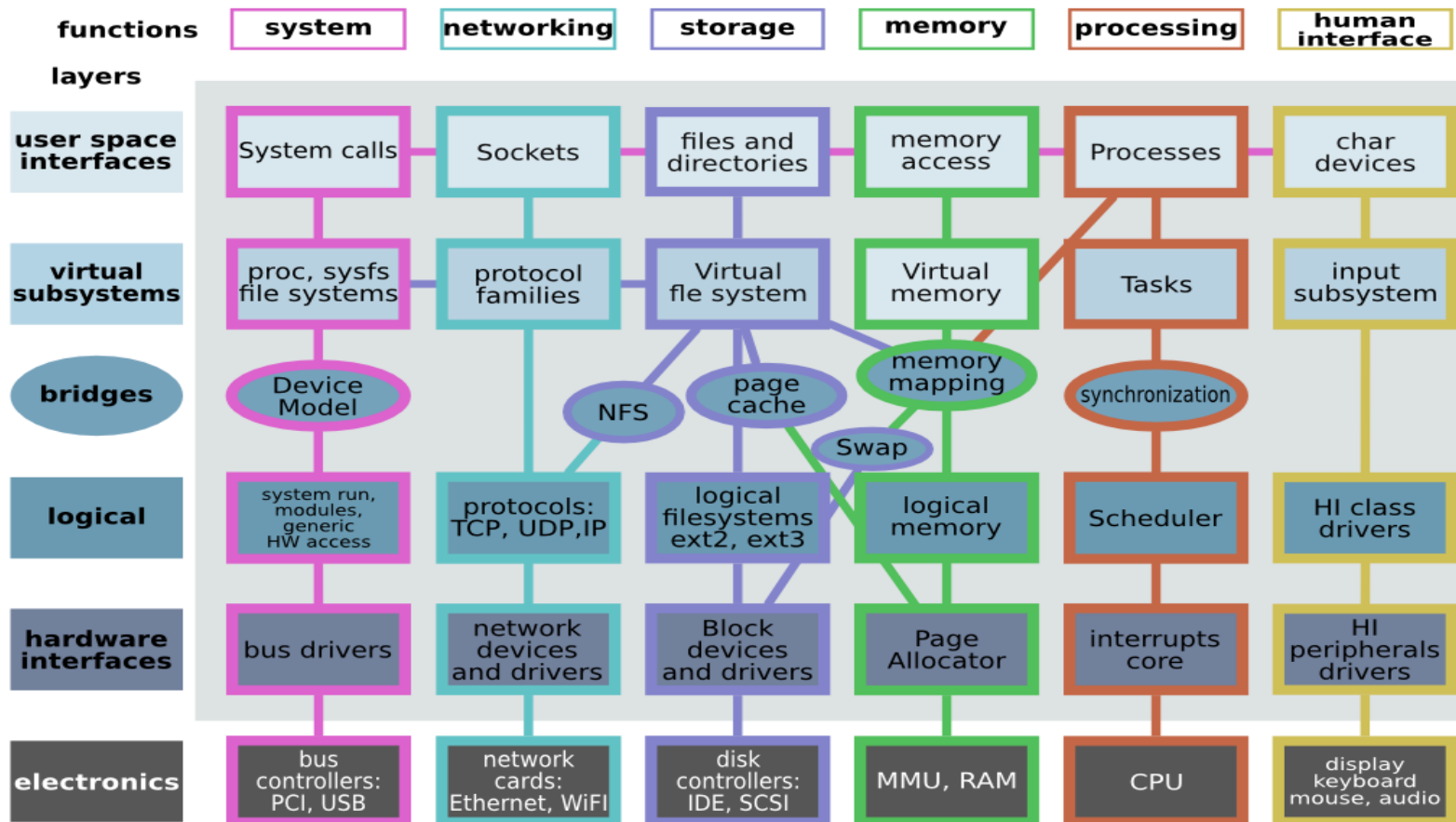
# 커널 (kernel)

- 주 메모리에 상주하는 운영체제 핵심
- 자주 사용되는 기능을 포함
- Nucleus라고도 불림



# 커널 (kernel)

Linux kernel diagram



© 2007-2009 Constantine Shulyupin <http://www.MakeLinux.net/kernel/diagram>

# 운영체제 발전의 용이성

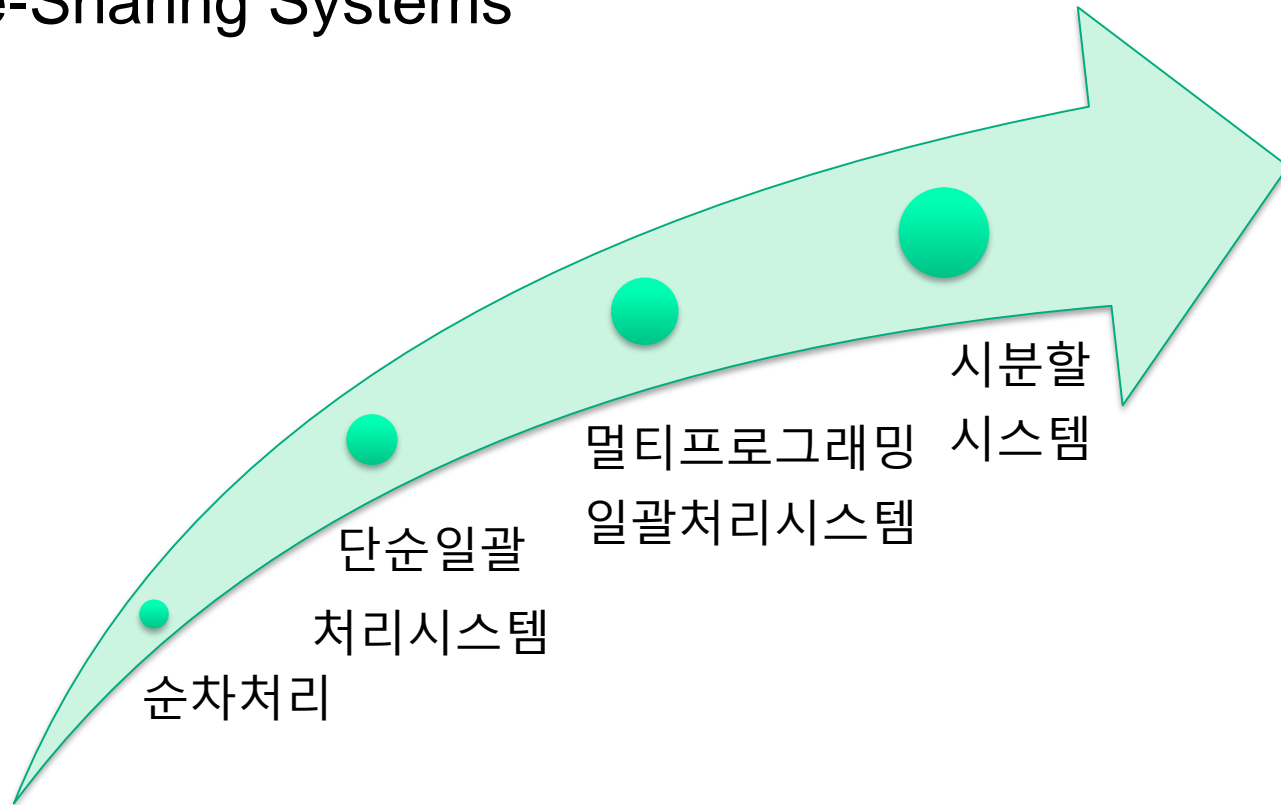
---

주요 운영체제가 계속해서 버전을 업그레이드 하는 이유로는

- 하드웨어 업그레이드와 새로운 형태의 하드웨어의 지원
  - ✓ HW 업그레이드에 따른 SW 기능 향상
  - ✓ Paging, window
- 새로운 서비스의 도입
  - ✓ 사용자 또는 시스템 관리자의 요구에 따라 새로운 서비스를 제공하기 위해 운영체제가 확장되어야 함
- 버그 수정
  - ✓ 어떤 운영체제도 결함을 가지고 있음

## 2.2 운영체제의 발전

- 순차처리(Serial Processing)
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-Sharing Systems



# 운영체제의 발전

## ■ 순차 처리

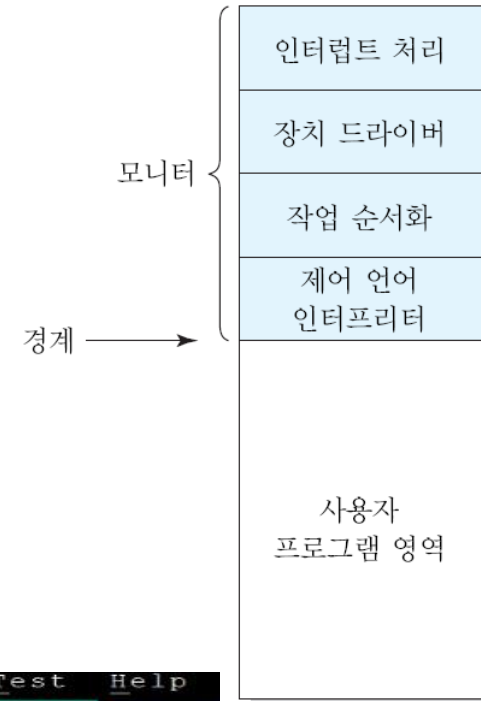
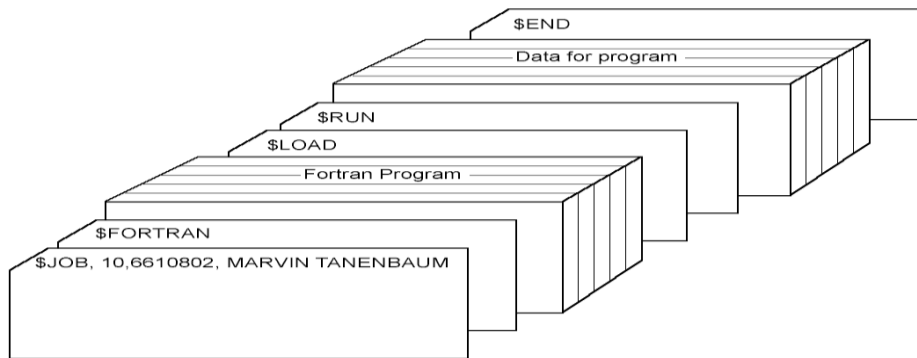
- ✓ 운영체제 없음
- ✓ 이 시대의 컴퓨터들은 디스플레이 발광체(light), 토글(toggle) 스위치, 입력 장치, 그리고 프린터로 구성된 콘솔(console)을 통해 운영
- ✓ 초기 시스템의 문제점
  - 스케줄링(scheduling) 시간
  - 준비시간(setup time)은 컴파일러와 소스 프로그램의 적재, 컴파일된 프로그램의 저장, 그리고 적재와 linking 을 포함함

## ■ 단순 일괄 처리 시스템

- ✓ 모니터
  - 일련의 이벤트를 제어하는 소프트웨어
  - 일괄처리 작업
  - 작업이 완료되면 제어는 다시 모니터로 넘어가고, 모니터는 즉시 다음 작업을 읽음
  - 각 작업의 결과는 출력되어 사용자에게 전달

# Job Control Language (JCL)

- 특수한 형태의 프로그래밍 언어
- 모니터에 명령어 제공
  - ✓ 사용할 컴파일러가 무엇인지
  - ✓ 사용할 데이터가 무엇인지



```
Edit Edit_Settings Menu Utilities Compilers Test Help
MATEKS.JOBLIB(IEBGENER) - 01.07
=====1-----2-----3-----4-----5-----6-----
//MATEKSD JOB (123), 'TRAINOOL', CLASS=A, MSGCLASS=A, MSGLEVEL=(
//          NOTIFY=&SYSUID
//***** Top of Data *****
//** TO COPY PS DATA SET ANOTHER PS DATA SET
//*****
//STEP10 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSOUT  DD SYSOUT=*
//SYSDUMP DD SYSOUT=*
//SYSUT1  DD DSN=MATEKS.TEST.PSFILE1, DISP=SHR
//SYSUT2  DD DSN=MATEKS.TEST.BACKUP,
//          DISP=(NEW,CATLG,DELETE),
//          SPACE=(TRK,(1,2),RLSE),
//          UNIT=SYSDA,
//          DCB=(DSORG=PS, RECFM=FB, LRECL=80, BLKSIZE=800)
//
F2=Split F3=Exit F4=Long F5=Rfind F6=
F8=Down F9=AUTOTYPE F10=Left F11=Right F12=
```

The screenshot shows a JCL job card with several annotations. A red box labeled 'JOB CARD' highlights the top section. A red box labeled 'EXEC Statement' highlights the 'EXEC PGM=IEBGENER' line. A red box labeled 'DD Statement' highlights the 'DD' statements. A red arrow points from the 'JOB CARD' box to the 'EXEC Statement' box. A red arrow points from the 'EXEC Statement' box to the 'DD Statement' box.

# 하드웨어 기능

---

제어를 서로 넘겨가며 수행하는 CPU 역할 이외에도

- 메모리 보호

- ✓ 모니터를 포함하고 있는 메모리 영역이 변경되지 않도록

- 타이머

- ✓ 단일 작업이 시스템을 독점하지 못하도록

- 특권 명령어(privileged instruction)

- ✓ 특정 기계 수준의 명령어는 모니터에 의해서만 수행되게

- 인터럽트

- ✓ 초기의 컴퓨터 모델에서는 사용하지 않았음, 인터럽트로 인하여 제어를 운영체제와 사용자 프로그램 사이에 융통성 있게 전달

# 메모리 보호

---

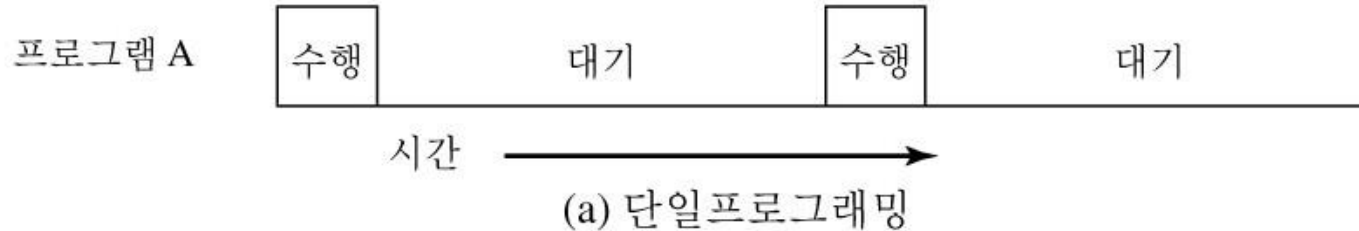
- 사용자 프로그램은 사용자 모드에서 수행
  - ✓ 특정 명령어는 수행되지 않을 수 있음
- 모니터는 시스템 모드에서 수행
  - ✓ 커널 모드 (또는 system mode)
  - ✓ 특권 명령어가 수행
  - ✓ 보호된 메모리 영역을 접근할 수 있음



# 멀티프로그래밍 일괄처리(Batch)시스템

## ■ 단일프로그래밍

- ✓ 처리기는 진행 전에 I/O 명령어를 대기해야 함



## ■ 시스템 이용률 (utilization) 예제

- ✓ I/O Devices are slow compared to the processor
- ✓ IO 장치의 전송완료를 기다리면서 소비하는 CPU시간이 96.8%

파일로부터 한 개의 레코드 판독	$15 \mu s$
100개의 명령어 수행	$1 \mu s$
파일에 한 개의 레코드 기록	$15 \mu s$
합계	$31 \mu s$

$$\text{CPU 이용 백분율} = \frac{1}{31} = 0.032 = 3.2\%$$

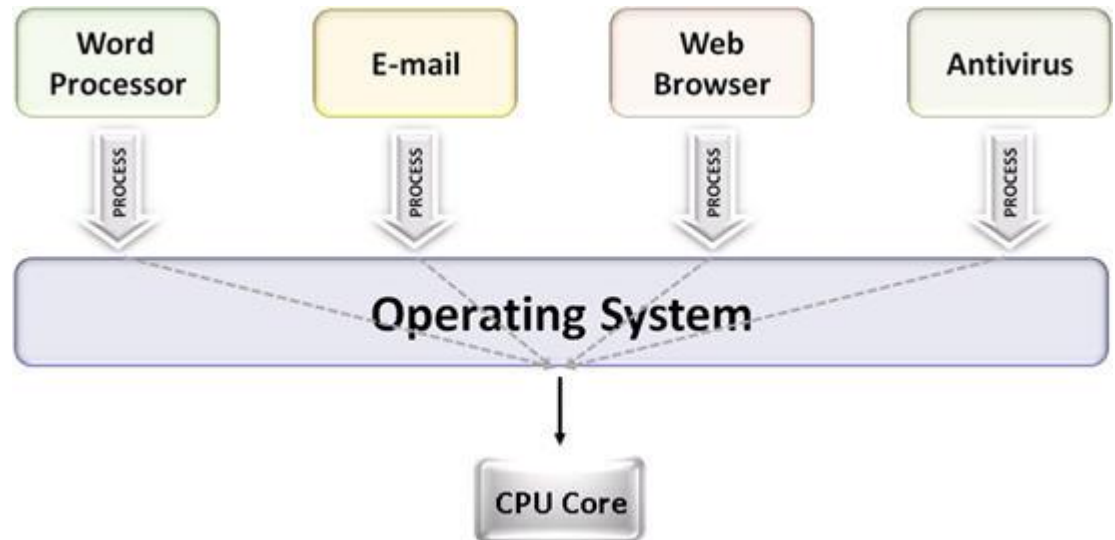
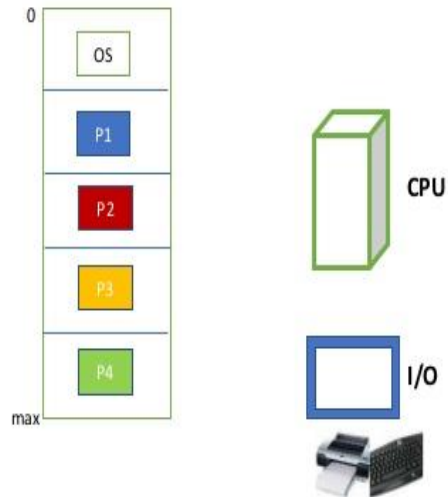
# 멀티 프로그래밍 (multiprogramming)

- Multitasking 이라고도 하며, 현재 OS의 주요 특징
- 여러 개의 프로그램이 메모리에 올라와 CPU를 번갈아가며 사용하며 작업을 수행
  - ✓ 즉, 병렬처리의 초보적인 형태로서, 1 CPU가 여러 프로그램을 동시에 실행
- 멀티프로그래밍을 위한 운영체제 기능
  - ✓ 운영체제에 의해 I/O 작업이 수행
  - ✓ 메모리 관리: 여러 개의 job에 메모리 할당
  - ✓ CPU 스케줄링: 준비 상태에 있는 job 중 하나를 선택해 CPU를 할당
- 멀티프로그래밍의 목적
  - ✓ CPU 이용률(utilization)의 극대화
- 단점
  - ✓ 수행중인 job과 상호작용이 안됨

# 멀티 프로그래밍

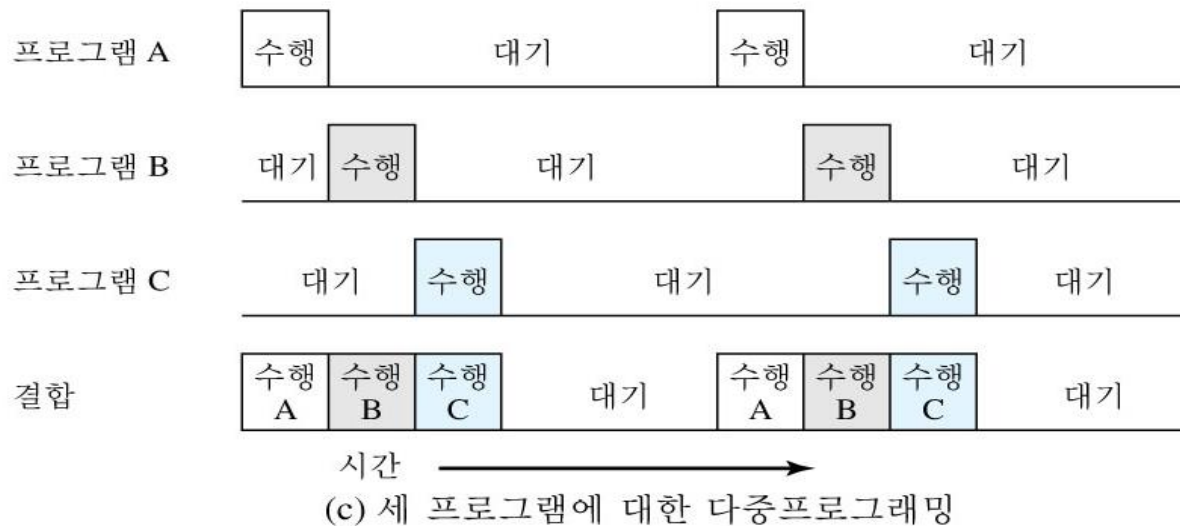
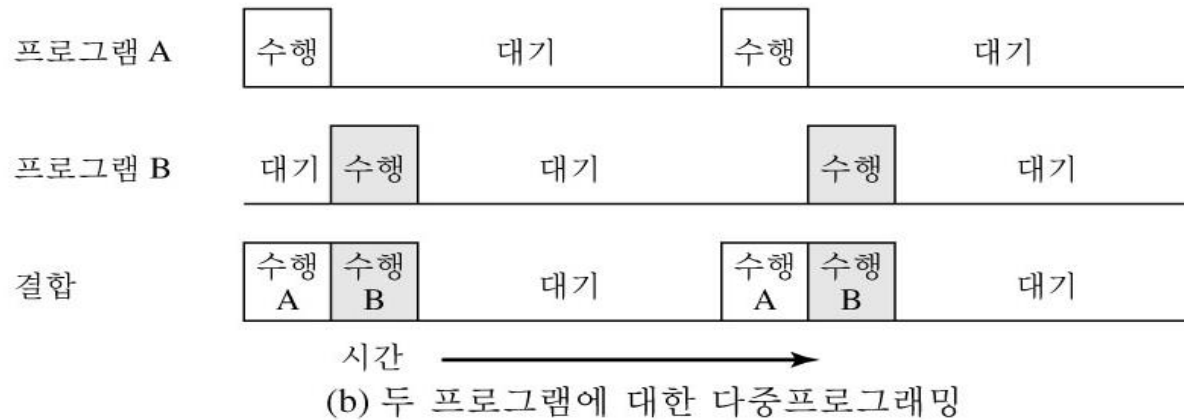
Baljit Singh Saini

## Multi-Programming



# 멀티 프로그래밍

- 한 작업이 입출력을 대기해야 할 때, 처리기는 다른 작업으로 제어를 넘김



# 샘플 프로그램의 수행 특성

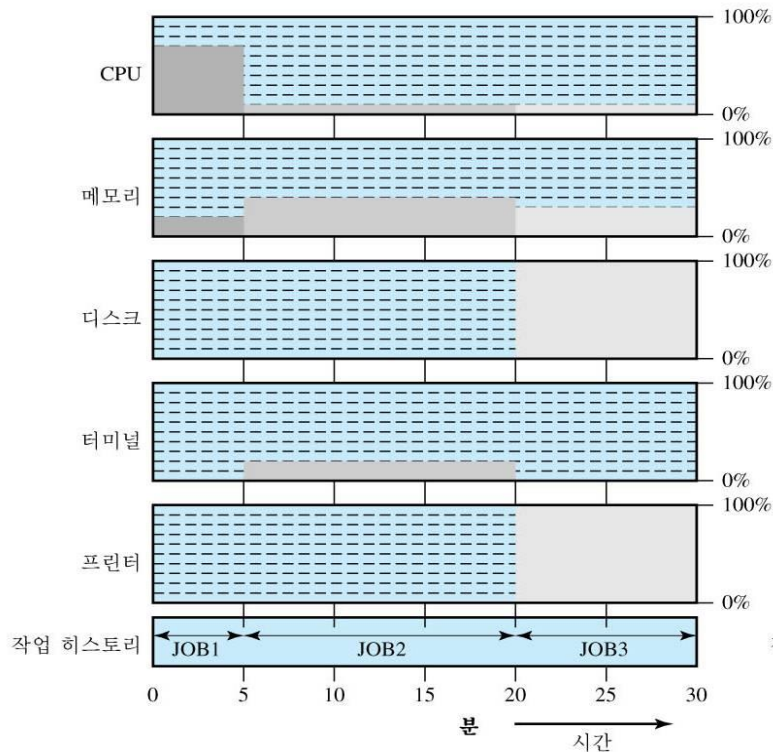
- Effects of Multiprogrammed batch system
  - ✓ System: Processor, disk, terminal, printer, and main memory with 250MB
  - ✓ Application: JOB1, JOB2, JOB3

표 2.1 | 샘플 프로그램의 수행 속성

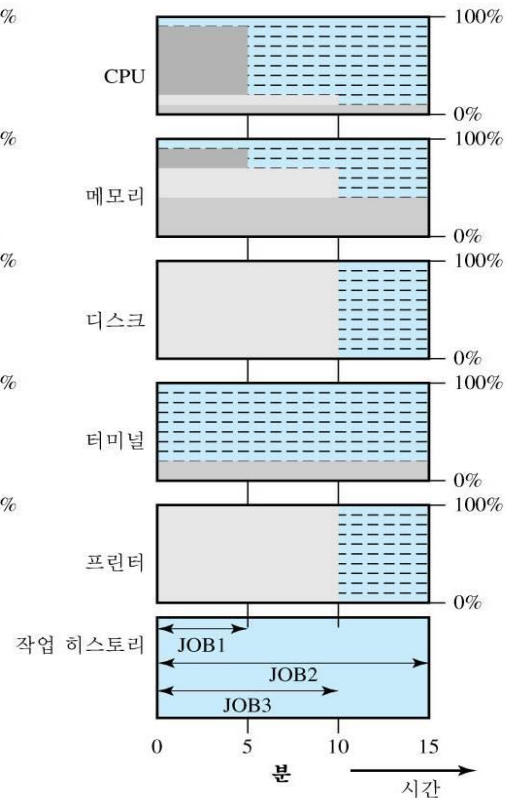
	JOB1	JOB2	JOB3
작업의 형태	대량의 계산	대량의 입출력	대량의 입출력
지속시간	5분	15분	10분
요구되는 메모리	50M	100M	80M
디스크가 필요한가?	아니오	아니오	예
터미널이 필요한가?	아니오	예	아니오
프린터가 필요한가?	아니오	아니오	예

# 이용률 히스토그램

- 멀티프로그래밍 요구 HW 기능
  - ✓ I/O 인터럽트
  - ✓ DMA 지원 HW
- 경과시간 : 30분 .vs. 15분



(a) 단일프로그래밍



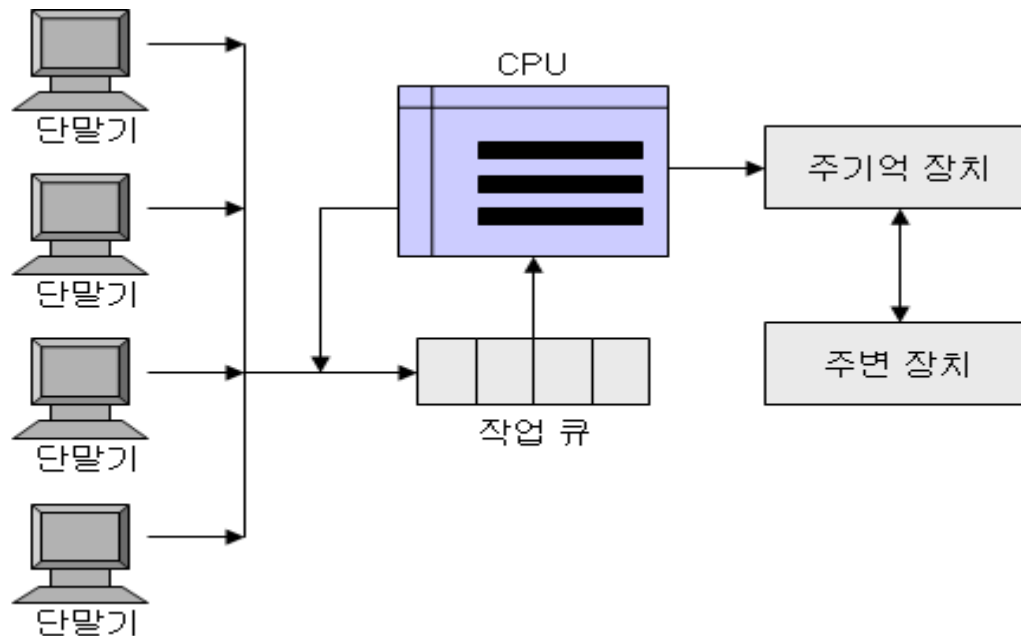
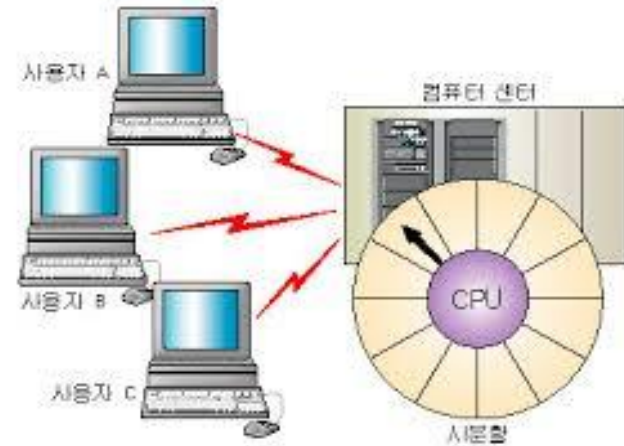
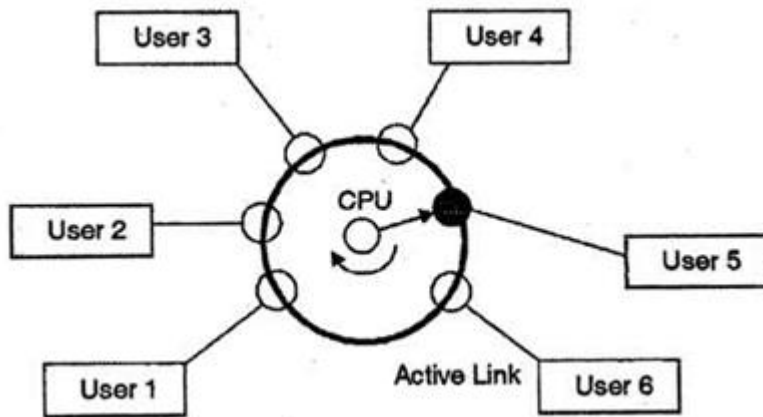
(b) 다중프로그래밍

# 시분할 (time sharing) 시스템

---

- 여러 개의 대화형 작업들을 다루기에 적합
  - ✓ 대화형 멀티프로그래밍
- 처리기 시간은 여러 사용자들이 공유(나눠 사용한다는 의미)
- 다수의 사용자들은 터미널을 통해 동시에 시스템에 접근
  - OS는 CPU 시간을 조금(quantum)씩 쪼개어(time slicing) 각 사용자 프로그램을 실행시킴
    - ✓ 매 클럭 인터럽트마다 제어가 OS에게 넘겨져 CPU가 다른 사용자에게 할당됨
    - ✓ 현재 수행중인 사용자는 선점(preempted) 되고 다른 사용자가 적재(load) 됨

# 시분할 (time sharing) 시스템

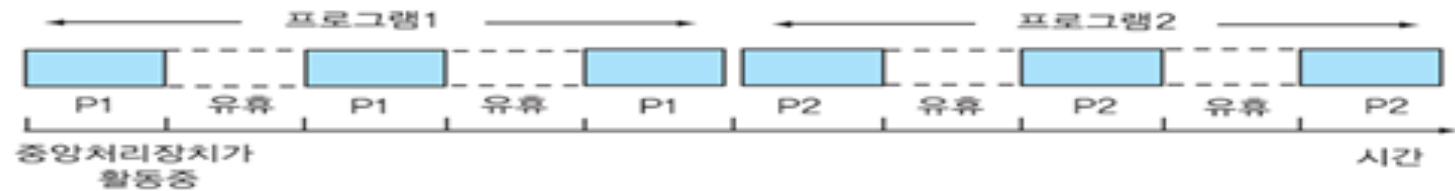




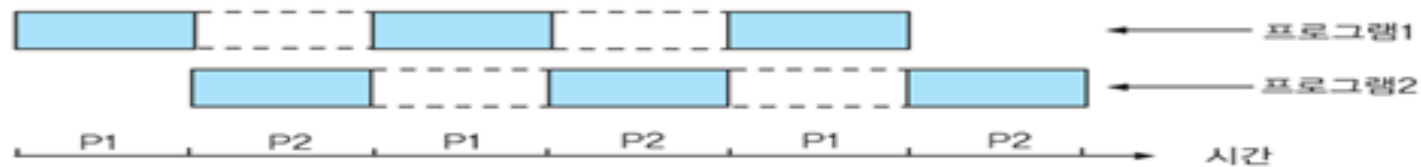
# 일괄처리 멀티프로그래밍 대 시분할 시스템

표 2.3 | 일괄처리 다중프로그래밍과 시분할의 비교

	일괄처리 다중프로그래밍	시분할
주요 목적	처리기 이용률의 최대화	응답시간의 최소화
운영체제에 대한 명령어 소스	작업과 함께 제공되는 작업 제어 언어 명령어	터미널에서 입력되는 명령어



(a) 순차성 실행

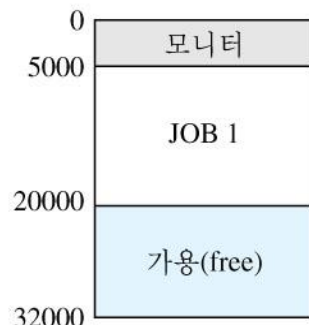


(b) 다중 프로그래밍에서의 실행

# Compatible Time-Sharing System (CTSS)

## ■ Examples of Time Sharing Systems

- ✓ MIT에서 개발한 초기의 시분할 시스템
- ✓ 32000 words of memory, monitor itself consumes 5000
- ✓ 4 jobs (JOB1: 15000, JOB2: 20000, JOB3: 5000, JOB4: 10000)
- ✓ Each start at the location of 5000



(a)



(b)



(c)



(d)



(e)



(f)

# Time Sharing Systems

---

- Usually, multiple jobs are in main memory in Time Sharing System
- Time Sharing Systems raise new problems
  - ✓ **scheduling**, context switch
  - ✓ memory **protection** between jobs
  - ✓ handle the **contention** for resources (such as disks and printers)
  - ✓ **access control** to resources (for example, file)
  - ✓ ...

## 2.3 Major Achievements

---

- 운영체제 발전과정에서 이루어진 5가지 주요 진전
  - ✓ Processes
  - ✓ Memory Management
  - ✓ Information protection and security
  - ✓ Scheduling and resource management
  - ✓ System structure

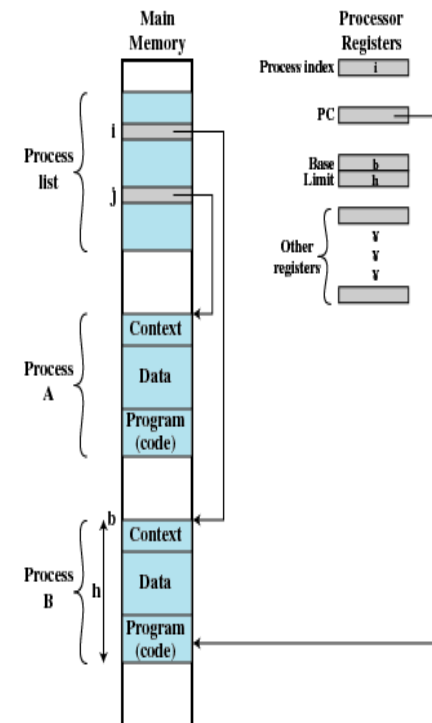
# Processes

## ■ Definitions

- ✓ A program in execution
- ✓ An instance of a program running on a computer
- ✓ The entity that can be assigned to and executed on a processor
- ✓ 단일 순차수행 스레드, 현재 상태, 연계된 시스템 자원 등에 의해 특징지어지는 활동 단위 (a unit of activity)

## ■ Three components

- ✓ An executable program code
- ✓ Associated data needed by the program
- ✓ 프로그램 수행 문맥(Execution context)
  - All information the operating system needs to manage the process
  - registers, priority, signals, state, statistics, ...



- 아래 3가지 컴퓨터 시스템 계열이 개발되면서 **타이밍과 동기화**라는 문제를 야기했고, 타이밍과 동기화가 프로세스 개념 탄생에 큰 기여를 함

## 멀티프로그래밍 일괄처리 동작

- 처리기는 주 메모리에 있는 다양한 여러 프로그램들을 돌아가며 실행시킨다

## time sharing

- 개별 사용자에게 빨리 응답해야 하면서도 동시에 여러 사용자들을 지원할 수 있어야 한다

## real-time transaction systems

- 많은 사용자들이 한 DB에 대해 동시에 질의와 갱신을 입력한다

# 운영체제 오류의 주요 원인

## ■ 부적절한 동기화

- 프로그램은 버퍼에 데이터가 채워질 때까지 기다려야 함
- 잘못 설계된 "신호 주고받기" 메커니즘은 데이터를 잃게 하거나 중복 수신하게 함

## ■ 상호배제 실패

- 여러 사용자나 프로그램이 동시에 공유 자원을 사용하려 시도한 경우
- 한 번에 한 루틴만이 공유 자원에 대한 업데이트를 할 수 있게 허용해야 함

## ■ 비결정적 프로그램 실행

- ✓ 메모리를 나눠 쓰는 여러 프로그램들은 처리기에 의해 번갈아 실행되는데,
- ✓ 이 때 프로그램들의 스케줄링 순서에 따라 실행 결과가 달라지는 현상

## ■ 교착상태(Deadlock)

- 두 개 이상의 프로그램들이 서로 상대방 실행을 기다리면서 무한 대기에서 빠질 수 있음
- 공유 자원 할당과 반환의 우연한 타이밍 미스 때문에 발생하기도 함

# 프로세스의 구성 요소

## ■ 프로세스의 세가지 구성요소:

- ✓ 실행 가능 프로그램
- ✓ 프로그램 수행에 필요한 데이터 (변수, 작업 공간, 버퍼 등)
- ✓ 프로그램의 실행 문맥 (또는 "프로세스 상태")

## ■ 실행 문맥은 매우 중요한 구성요소임:

- ✓ 실행 문맥을 통해 OS는 프로세스를 감시하고 제어하기 때문
- ✓ 다양한 프로세스 레지스터 값들이 포함되어 있음
- ✓ 프로세스 우선순위 정보나 프로세스가 특정 I/O가 끝나기를 기다리고 있는지의 여부 등이 기록되어 있음



# 프로세스 관리

- 실행 중인 프로세스의 모든 상태는 각 프로세스의 문맥에 저장됨
- OS에 새로운 기능을 추가하려면 문맥 구조를 확장하여 새 기능 지원에 필요한 정보를 문맥에 추가해야 한다

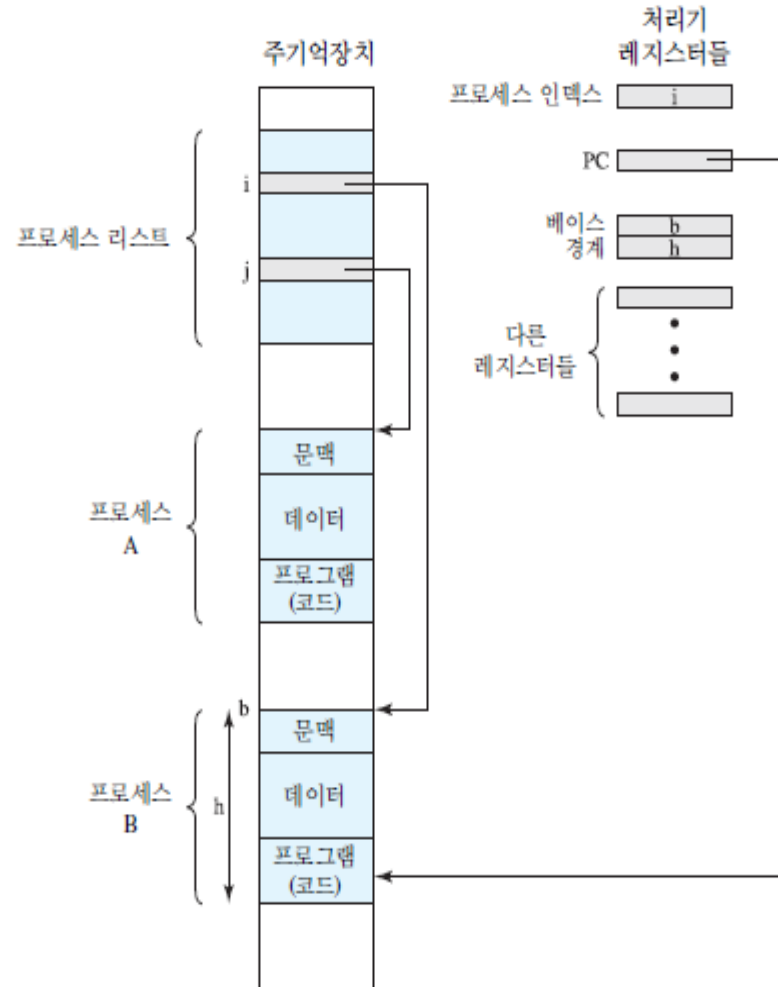
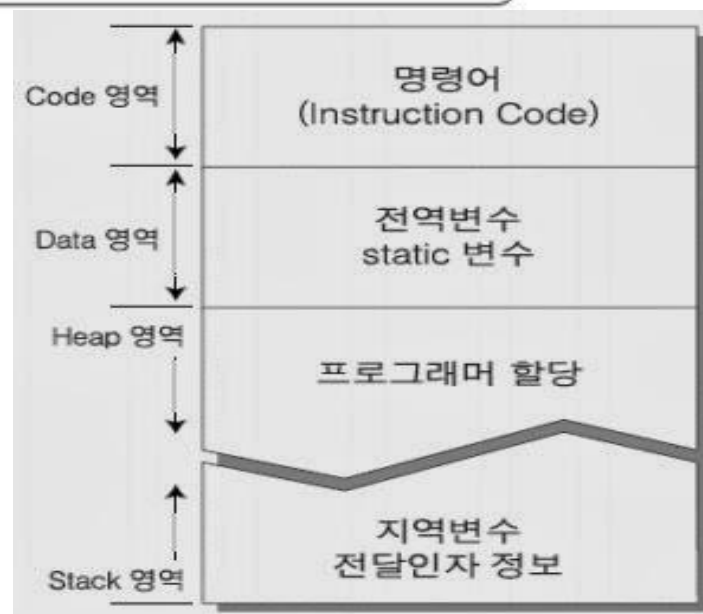
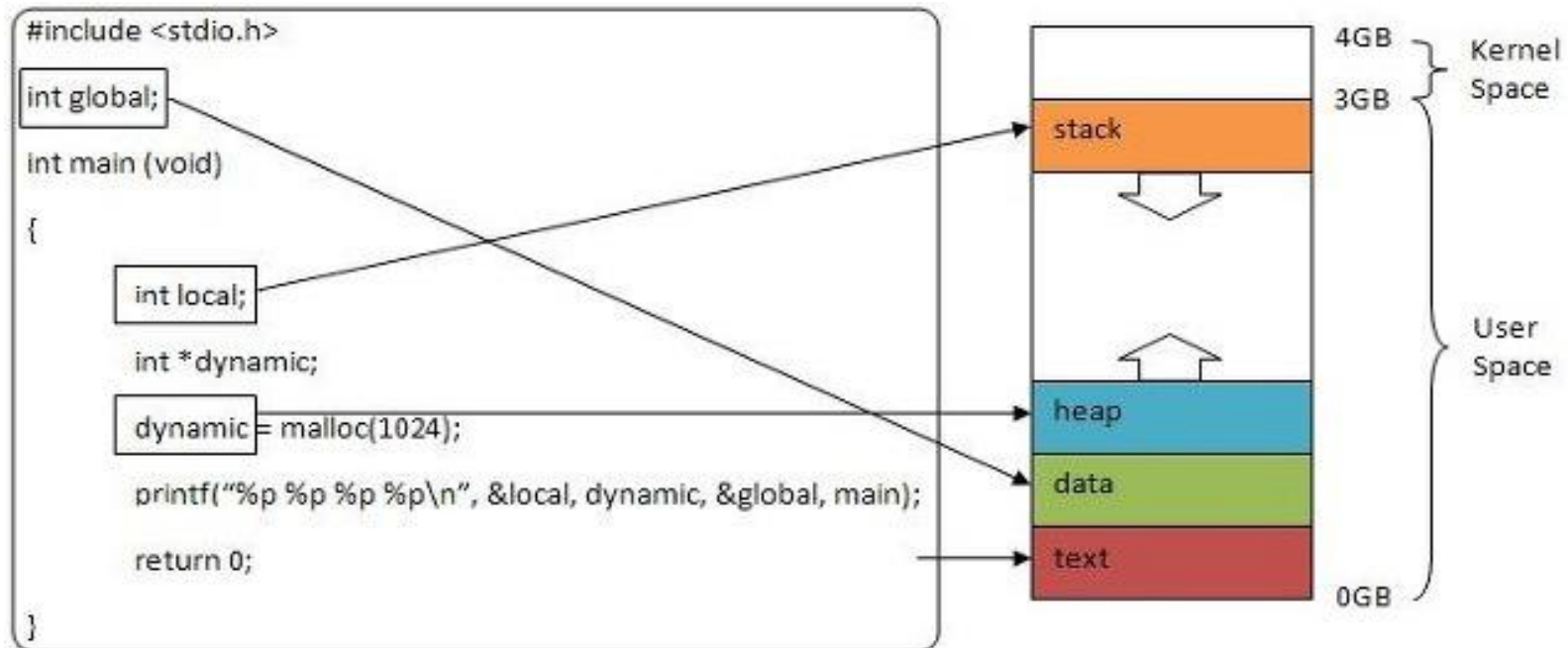


그림 28 전형적인 프로세스 구현

# 프로세스의 메모리할당 구조



# Memory Management

---

## ■ Functions

- ✓ Process isolation
- ✓ Support of modular programming
- ✓ Protection and access control
- ✓ Automatic allocation and management
- ✓ Long-term storage

## ■ Virtual memory

- ✓ 물리적으로 이용 가능한 주기억장치의 크기에 관계없이, 프로그램이 메모리의 주소를 논리적인 관점에서 참조할 수 있도록 하는 기법
- ✓ virtual address(page# + offset), physical memory (real memory)
- ✓ paging, segmentation
- ✓ 여러 개의 사용자 프로그램들이 동시에 주 메모리에 올라와 있어도 프로그램들이 통일된 주소 지정 방식을 사용할 수 있게 해 주는 역할을 한다

# 페이징

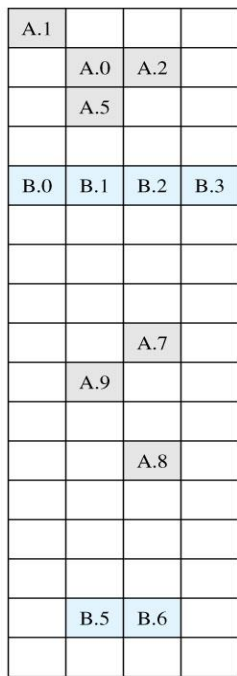
---

- 프로세스 메모리를 페이지라고 하는 고정-크기 블록 단위로 잘라 관리하는 기법
- 프로그램은 가상 주소를 통해 원하는 워드에 접근
  - ✓ 가상주소 구성: 가상 페이지 번호 + 그 페이지 내에서의 오프셋
  - ✓ 가상 페이지에 해당하는 실제 페이지는 주 메모리 내 또는 보조 기억장치 내 어느 위치에도 존재할 수 있다
- 프로그램에서 사용한 가상 주소를 주 메모리 내의 물리 주소로 동적으로 바꾸기 위해 OS가 매핑 정보를 제공

# Memory Management

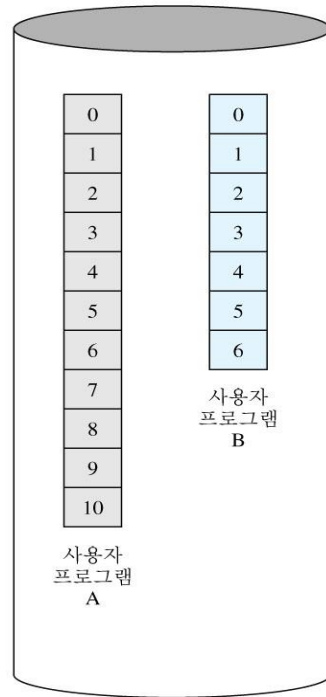
## ■ Virtual memory concepts

- ✓ 페이징 시스템은 프로그램이 사용하는 가상주소와 주기억장치의 실제주소간 동적 매핑을 제공



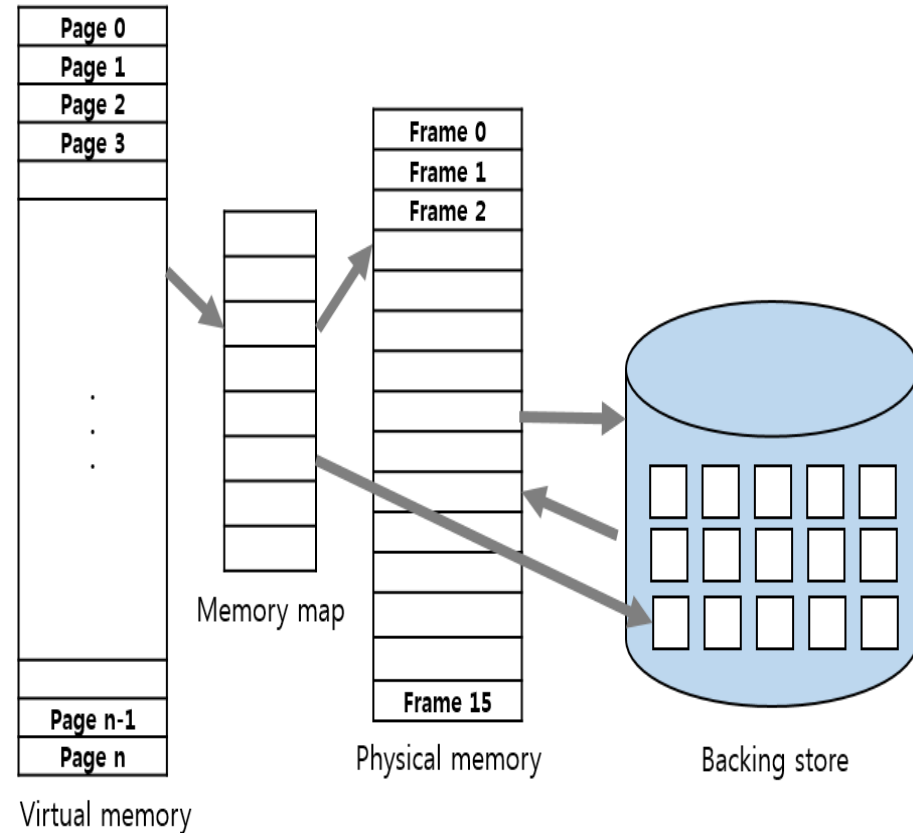
주기억장치

주기억장치는 페이지와 동일한 크기의 고정길이 프레임들로 구성된다. 프로그램 수행을 위해, 그 페이지들 중 일부 혹은 전부가 주기억장치에 있어야 한다.



디스크

보조기억장치(디스크)는 고정길이 페이지들을 보유한다. 사용자 프로그램은 몇 개의 페이지들로 구성된다. 모든 프로그램과 운영체제의 페이지들은 파일들처럼 디스크에 있다.



# Memory Management

## ■ Virtual memory addressing

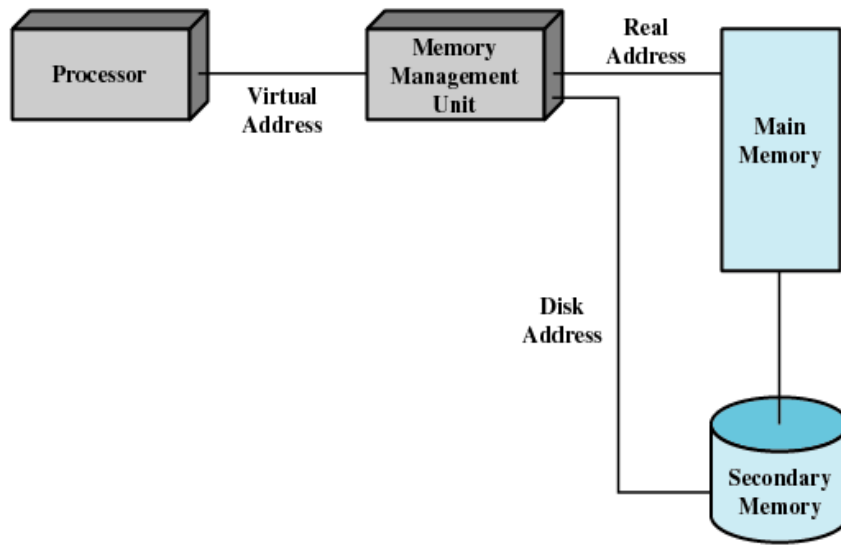


Figure 2.10 Virtual Memory Addressing

The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. A red box highlights the 'Private Bytes' column, which shows the memory usage for each process. The processes listed include Google Chrome, Windows 탐색기, LINE, MySQL Notifier, taskmgr.exe, and Windows 작업 관리자.

이미지 이름	사용자 이름	...	메...	설명
chrome.exe...	mooneegee	00	87,792 KB	Google Chrome
chrome.exe...	mooneegee	00	75,740 KB	Google Chrome
chrome.exe...	mooneegee	01	55,244 KB	Google Chrome
chrome.exe...	mooneegee	00	40,404 KB	Google Chrome
chrome.exe...	mooneegee	00	25,380 KB	Google Chrome
chrome.exe...	mooneegee	00	24,492 KB	Google Chrome
explorer.exe	mooneegee	00	20,060 KB	Windows 탐색기
chrome.exe...	mooneegee	00	16,580 KB	Google Chrome
Line.exe *32	mooneegee	00	6,080 KB	LINE
csrss.exe		01	5,236 KB	
MySQLNotifi...	mooneegee	00	4,784 KB	MySQL Notifier
chrome.exe...	mooneegee	00	4,492 KB	Google Chrome
taskmgr.exe	mooneegee	00	3,580 KB	Windows 작업 관리자
Mini_Monito...	mooneegee	00	2,308 KB	CleanMem Mini M
TSVNCache...	mooneegee	00	1,728 KB	TortoiseSVN status
calc.exe	mooneegee	00	1,616 KB	Windows 계산기

## ✓ file system

- as long-term storage
- Information stored in named objects called files

# Information Protection and Security

---

- 가용성(Availability)
  - ✓ Concerned with protecting the system against interruption
- 기밀성(Confidentiality)
  - ✓ Assuring that users cannot read data for which access is unauthorized
- 데이터 무결성(Data Integrity)
  - ✓ Protection of data from unauthorized modification
- 신빙성(Authenticity)
  - ✓ 사용자 신원과 메시지나 데이터의 유효성에 대한 적절한 검증과 관련됨

# Scheduling and Resource Management

---

## ■ Scheduling

- ✓ Resource(CPU, 메모리, IO장치들) allocation

## ■ Scheduling factors

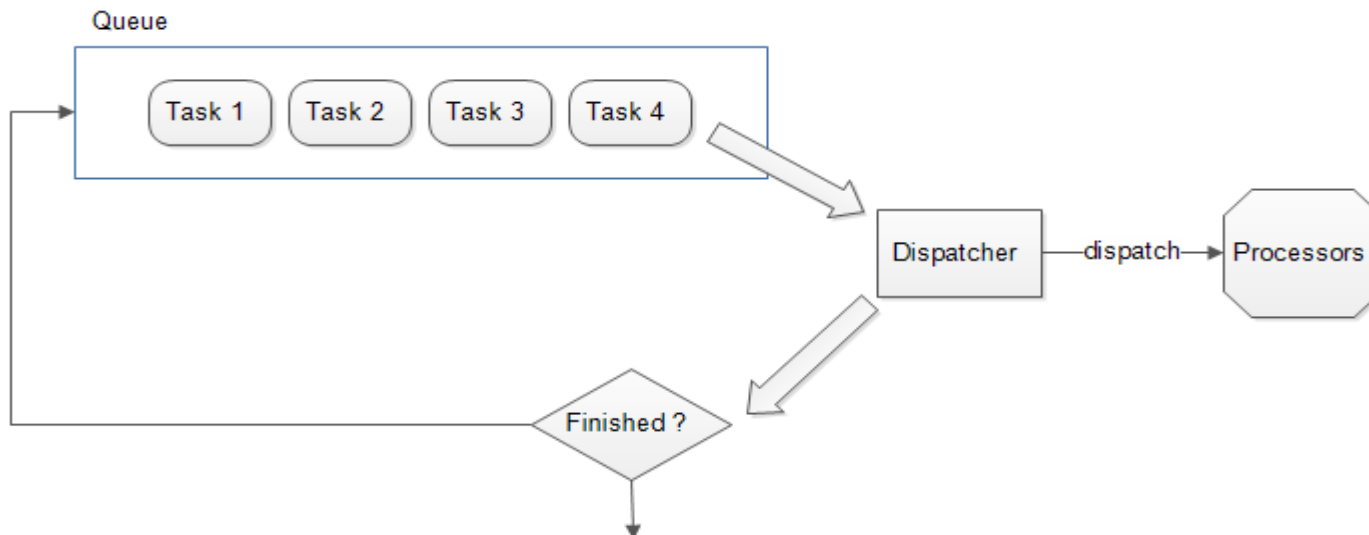
- ✓ Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible
- ✓ Fairness
  - Give equal and fair access to resources
- ✓ 반응시간 차등화(Differential responsiveness)
  - Discriminate among different classes of jobs
  - Dynamic adaptation
  - Deadline



# Scheduling and Resource Management

## ■ Scheduling model

- ✓ scheduling of processes and allocation of resources
- ✓ implementation: diverse kind of queues, kernel entry points
- ✓ 디스패처: round-robin 방식으로 큐에 있는 프로세스에게 정해진 시간 할당
  - 라운드로빈 방식은 Q에 있는 각 프로세스에게 CPU를 차례대로 일정시간 할당해주는 방식.
  - Dispatch는 멀티태스킹 환경에서 우선순위가 높은 작업이 수행되도록 CPU 할당을 받아 해당 프로세스가 실행되도록 하는 작업



# Scheduling and Resource Management

## ■ Scheduling model

### ✓ 단기 큐(short-term Q)

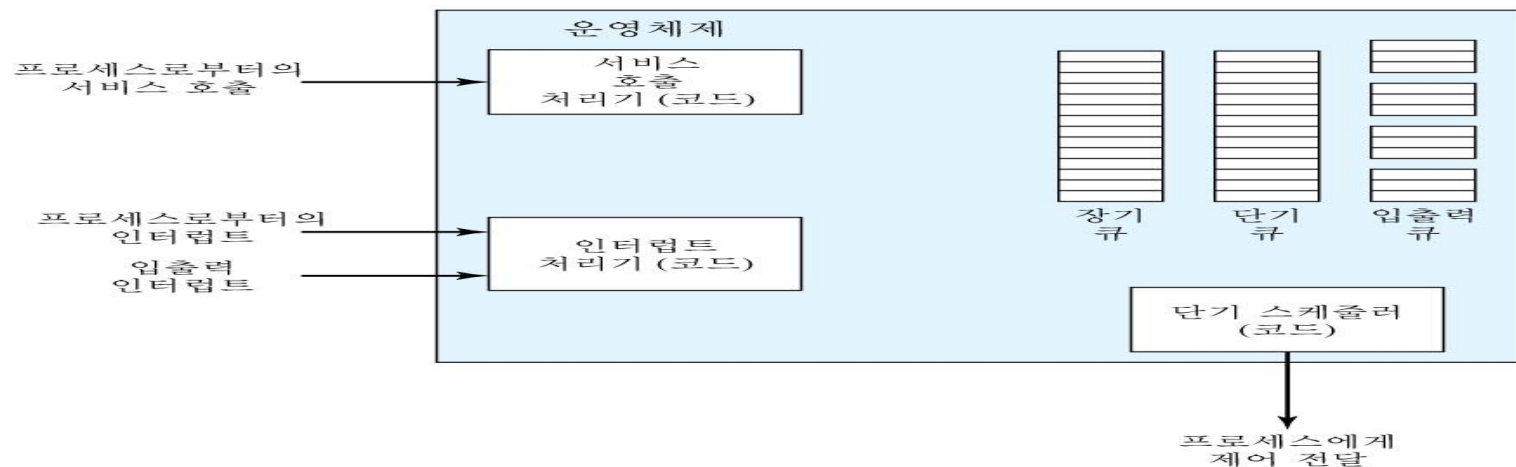
- 주기억장치에 로드되어 있는 수행 준비상태의 프로세스들로 구성
- 이들 프로세스들 중 하나가 단기 스케줄러 (또는 디스패처) 에 의해 선정된 다음 CPU에 할당됨

### ✓ 장기 큐(long-term Q)

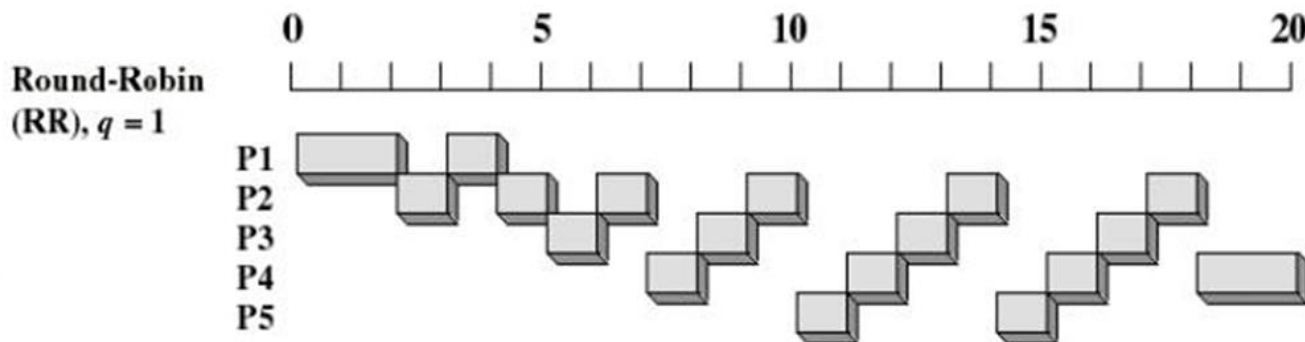
- CPU를 사용하기 위해 대기하고 있는 새로운 작업들의 리스트
- OS는 프로세스를 장기 큐에서 단기 큐로 이동시킨후 시스템에서 처리

### ✓ 입출력 큐(IO Q)

- 각 입출력장치를 사용하기 위해 대기중인 프로세스



## 라운드-로빈(Round-Robin) 스케줄링



- 선택함수: **FCFS**와 같다
- 결정모드: **preemptive**(선점)
  - ◆ 프로세스는 하나의 **time slice (quantum, typically from 10 to 100 ms)** 동안에 실행된다
  - ◆ **clock interrupt**가 발생하면 현재 실행중인 프로세스를 **ready queue**로 보내고 **FCFS** 방식으로 새로운 프로세스를 실행한다

## 2.4 Modern Operating Systems

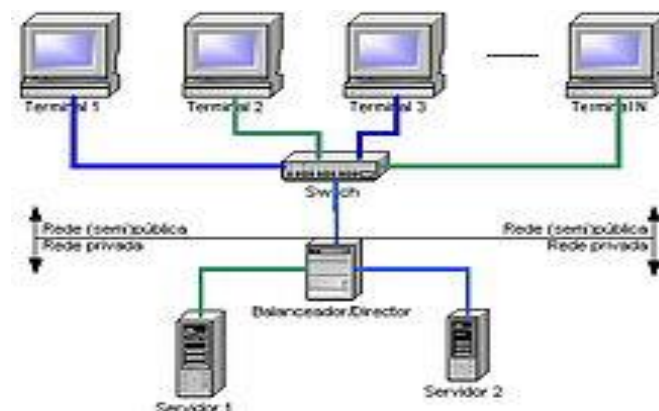
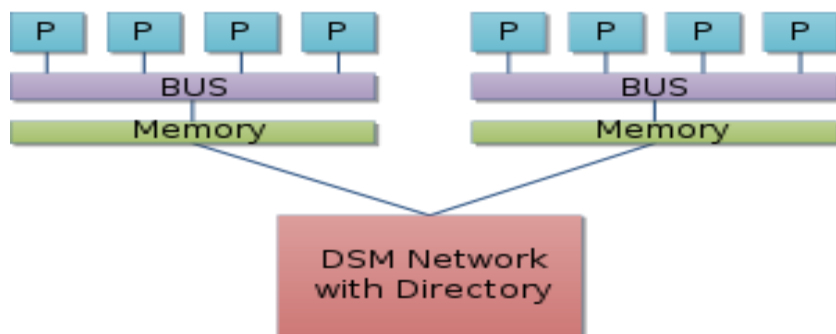
### ■ Triggering

#### ✓ new hardware

- 64-bit CPU

- SMP, NUMA(Non-Uniform Memory Access), Clustering

- NUMA는 CPU와 memory가 1 set로 구성되어 있어서 Bus를 통해 각각의 노드를 연결하는 구조로서, 버스에 노드를 추가하는 방식으로 확장성이 매우 좋은 장점



- High-speed bus and network

- Increasing size and variety of memory storage

#### ✓ new application

- Multimedia, internet, security

#### ✓ embedded system

## 2.4 Modern Operating Systems

---

- New approach – 새로운 OS 구조를 위한 다양한 접근방법 및 설계 요소들로는
  - ✓ Microkernel architecture
  - ✓ Multithreading
  - ✓ Symmetric multiprocessing
  - ✓ Distributed operating system
  - ✓ Object-oriented design

# Modern Operating Systems

## ■ Microkernel architecture

- ✓ Assigns only a few essential functions to the kernel
  - Address spaces
  - Interprocess communication (IPC)
  - Basic scheduling
- ✓ 나머지 OS 서비스들은 서버라고 하는 프로세스에 의해 제공되며, 마이크로 커널에 의해 일반 응용과 동일하게 취급됨

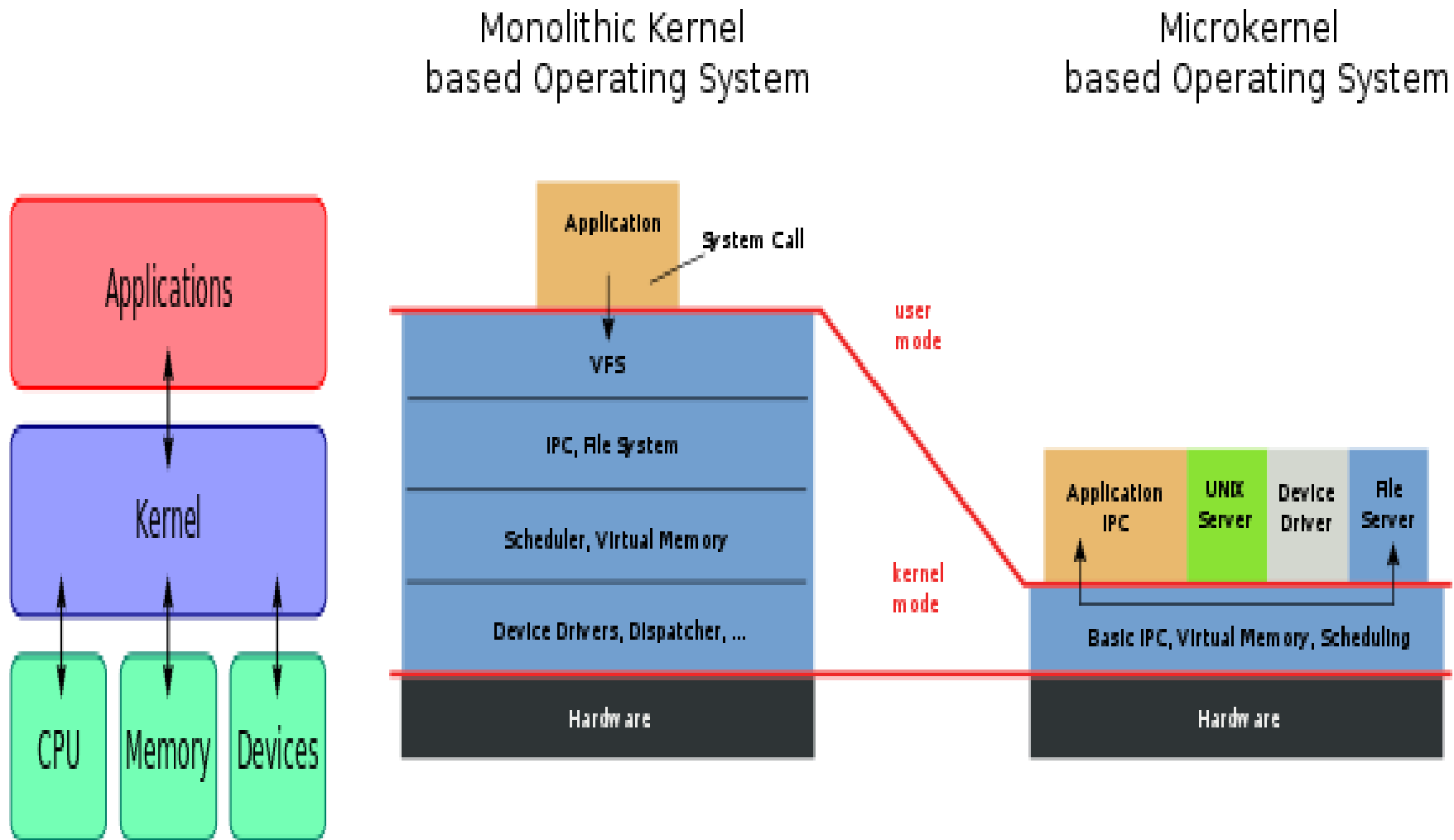
cf) Monolithic kernel

- ✓ The Approach



# Modern Operating Systems

## ■ Microkernel architecture



# Modern Operating Systems

## ■ Multithreading

☞ light-weight process

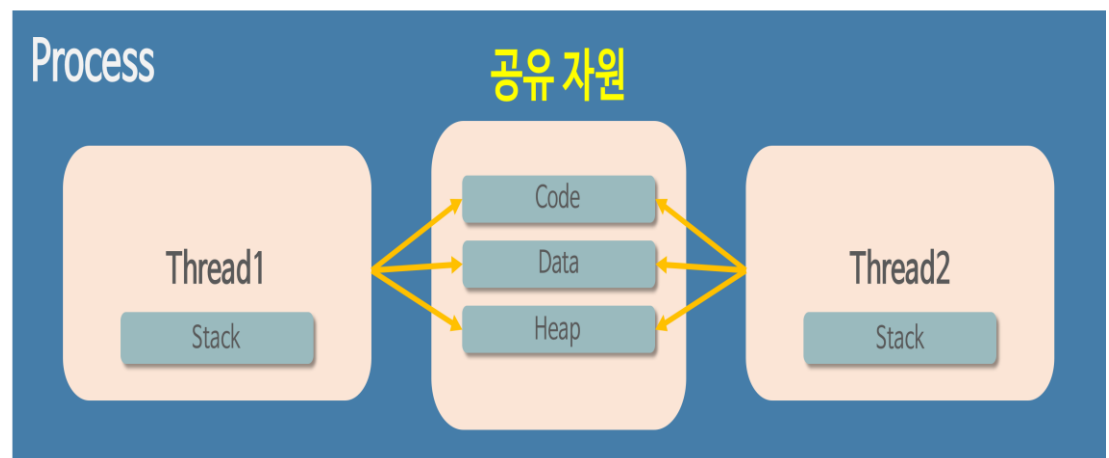
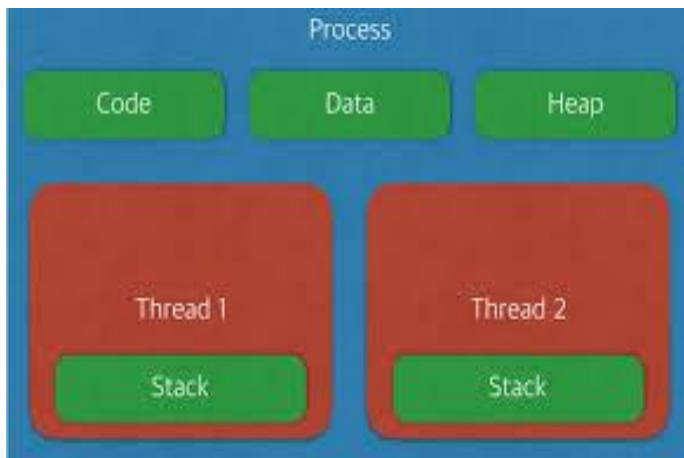
- ✓ 한개의 프로세스를 동시에 수행될 수 있는 여러 개의 쓰레드들로 나누어 병행적으로(concurrently) 실행시키는 기법

### 쓰레드(Thread)

- CPU에 작업을 할당하는 디스패칭 단위
- 서브루틴 분기/복귀를 가능하게 하기 위한 처리기 문맥과 쓰레드만의 데이터 영역을 포함하고 있음
- 순차적으로 실행되며 인터럽트 가능(interruptible)

### 프로세스(Process)

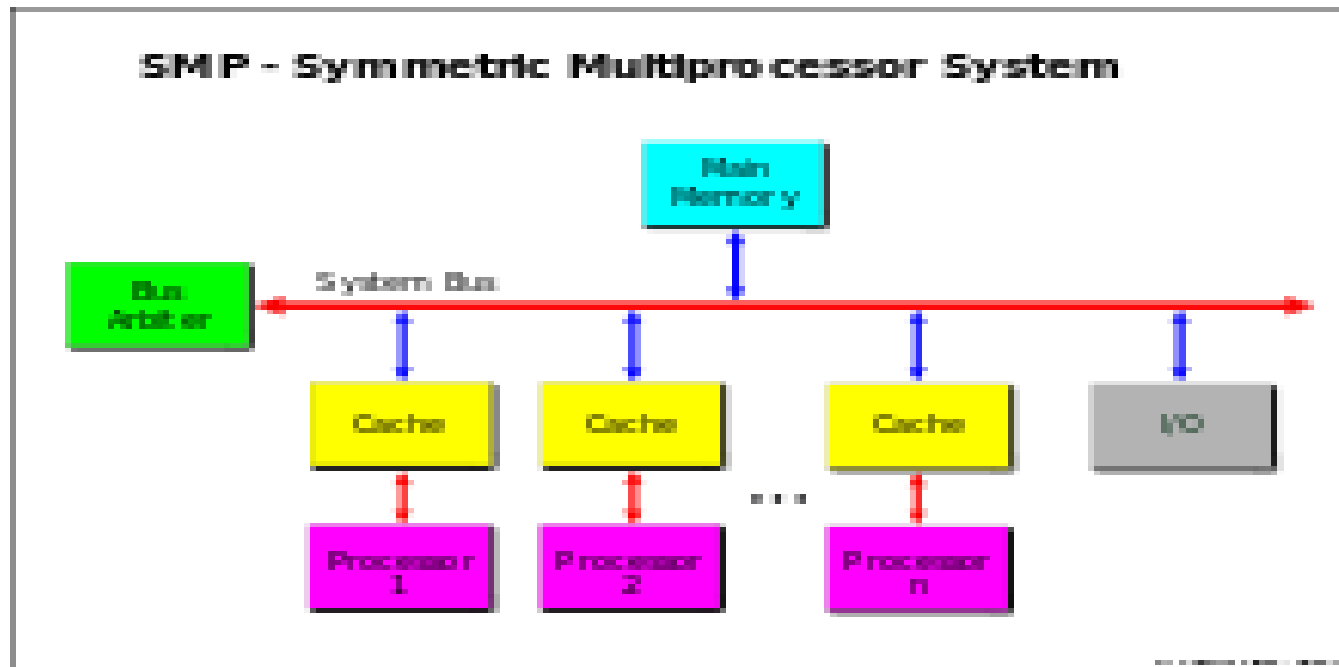
- 하나 이상의 쓰레드와 관련 시스템 자원들의 컨테이너 역할
- 프로세스를 통해 프로그래머는 응용의 모듈화 수준과 응용 관련 사건들의 타이밍을 조절할 수 있다





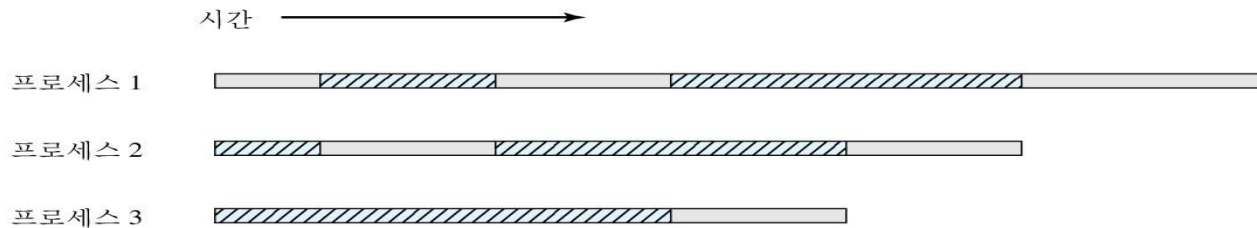
# Modern Operating Systems

- SMP라는 컴퓨터 하드웨어 구조를 일컫는 용어이지만, SMP 구조를 활용하는 OS 기능을 의미하기도 함
- 여러 프로세스들이 동시에 실행될 수 있는 구조
- OS는 스레드나 프로세스들을 각 처리기로 스케줄링 하고, 처리기들 간의 동기화도 책임진다

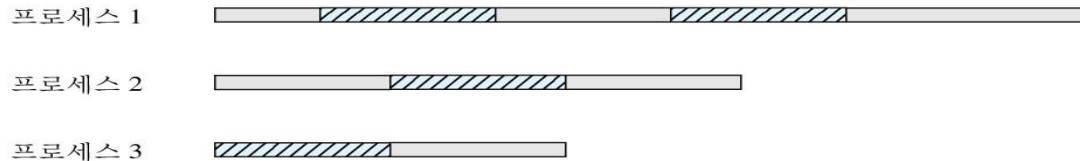


# Modern Operating Systems

- 대칭형 다중처리(Symmetric multiprocessing :SMP)
  - ✓ There are multiple processors
  - ✓ These processors share same main memory and I/O facilities
  - ✓ All processors can perform the same functions



(a) 인터리빙(다중프로그래밍, 단일처리기)



👉 many-core

(b) 인터리빙과 오버래핑(다중처리, 두 처리기)

▨ 블록 상태    □ 수행 상태

멀티프로그래밍 .vs. 멀티프로세싱


# Modern Operating Systems

---

## ■ Distributed operating systems

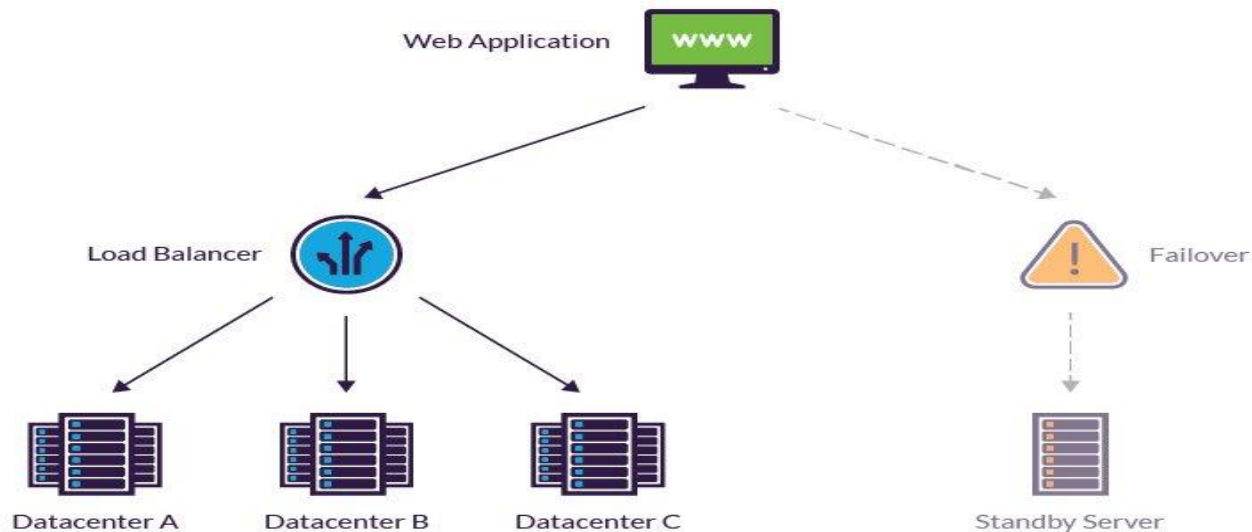
- ✓ For a cluster of separated nodes (computers)
- ✓ Each nodes owns its main memory, secondary storage, and I/O modules (possible asymmetric)
- ✓ 사용자가 하나의 주기억장치 공간과 하나의 보조기억장치 공간을 사용하는 것처럼 느끼게 해주고, 분산 파일시스템과 같은 통합된 접근 기능 제공
- ✓ RPC, Load balancing

## ■ Object-oriented design

- ✓ 소형화된 커널(a small kernel) 에 모듈식 확장 기능을 추가할 때 유용
- ✓ Enables programmers to customize an operating system without disrupting system integrity  decomposable
- ✓ Encapsulation, Inheritance, ...

## 2.5 결함허용(fault tolerance)

- HW, SW 고장에도 불구하고 어떤 시스템 또는 컴포넌트가 계속 정상 작동할 수 있는 능력을 일컫는 용어
  - ✓ 대개 어느 정도의 여분 설비(redundancy)가 동반됨
- 시스템의 신뢰성을 높이기 위해 고안됨
  - ✓ 결함 허용을 지원하려면 비용과 성능에 손해가 발생하는 것이 보통임
  - ✓ (예) 은행등 금융사, 항공사, 정부통합전산센터
- 결함 허용 척도를 시스템에 도입할 것이냐 아니냐는 그 자원이 얼마나 중요한 자원이냐에 따라 결정됨

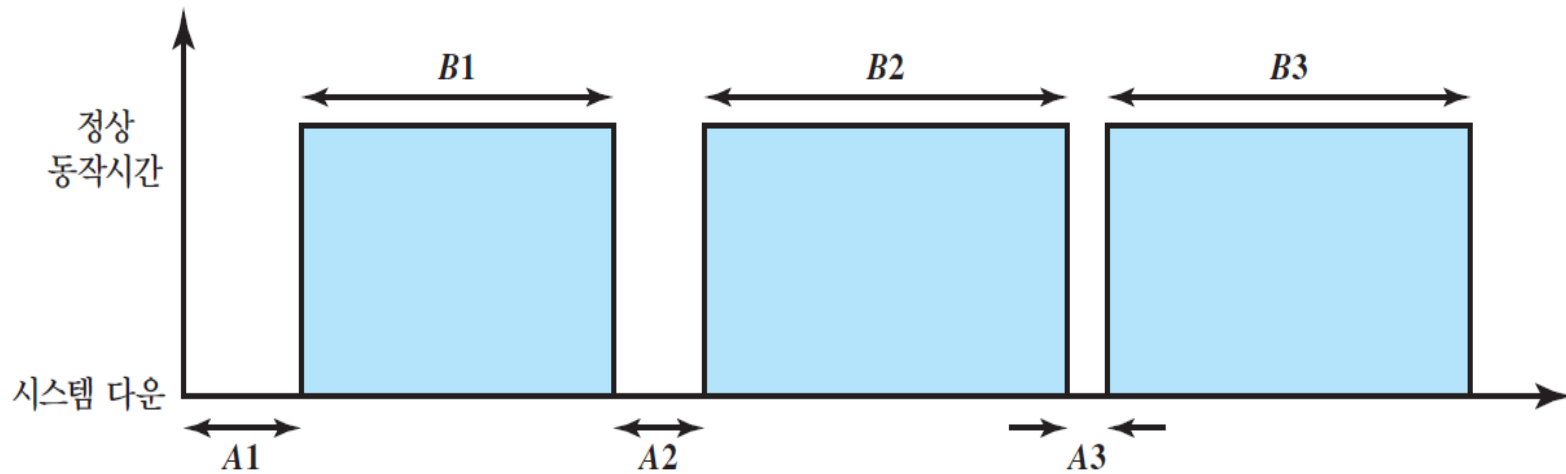


# 결함허용의 기본 개념

- 결함 허용 수준을 측정하는 단위들:
  - 신뢰성(Reliability):  $R(t)$ 
    - $t = 0$  시간에 정상작동하고 있던 시스템이 임의의 시간  $t$  까지 계속 정상적으로 작동할 확률로 정의
  - Mean Time To Failure (MTTF): 결함이 발생하기까지의 평균 시간
  - Mean Time To Repair (MTTR): 결함이 발생한 부품이나 소프트웨어 모듈을 수리완료 하거나 새것으로 교체하기까지 걸리는 평균 시간
  - 가용성(A: Availability) : 시스템이나 서비스가 기동된 후 어느 시점까지의 총 시간 중 사용자의 요청을 서비스할 수 있었던 시간이 얼마나 되는지의 비율

# MTTF와 MTTR의 관계

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$



$$MTTF = \frac{B1 + B2 + B3}{3} \quad MTTR = \frac{A1 + A2 + A3}{3}$$

# 가용성 등급

표 2.4 가용성 등급

등급	가용성	연간 다운타임
항상 가용	1.0	0
결함허용	0.99999	5분
결함감내	0.9999	53분
고가용성	0.999	8.3시간
보통 가용성	0.99-0.995	44-87시간

- SLA(Service Level Agreement:서비스레벨협약) :  
Cloud Service

# 결함 및 여분 설비 유형

## ■ 영구적 유형

- 한 번 발생하면 영구적으로 존재한다.
- 새 것으로 교체되거나 수리될 때까지 계속 결함이 발생한 채로 있다

### 공간적(물리적) Redundancy

같은 기능을 동시에 수행할 수 있는 여분의 부품 여러 개를 설치하는 기법. 한 부품에 결함이 발생하면 여분으로 준비한 부품들이 백업의 역할을 하도록 설정

### 시간적(Temporal) redundancy

오류가 발생했을 때 동일한 기능이나 동작을 계속 반복할 수 있도록 하는 방식. 영구적 결함에는 효과가 없지만 일시적 결함에는 큰 효과가 있음

## ■ 일시적 유형

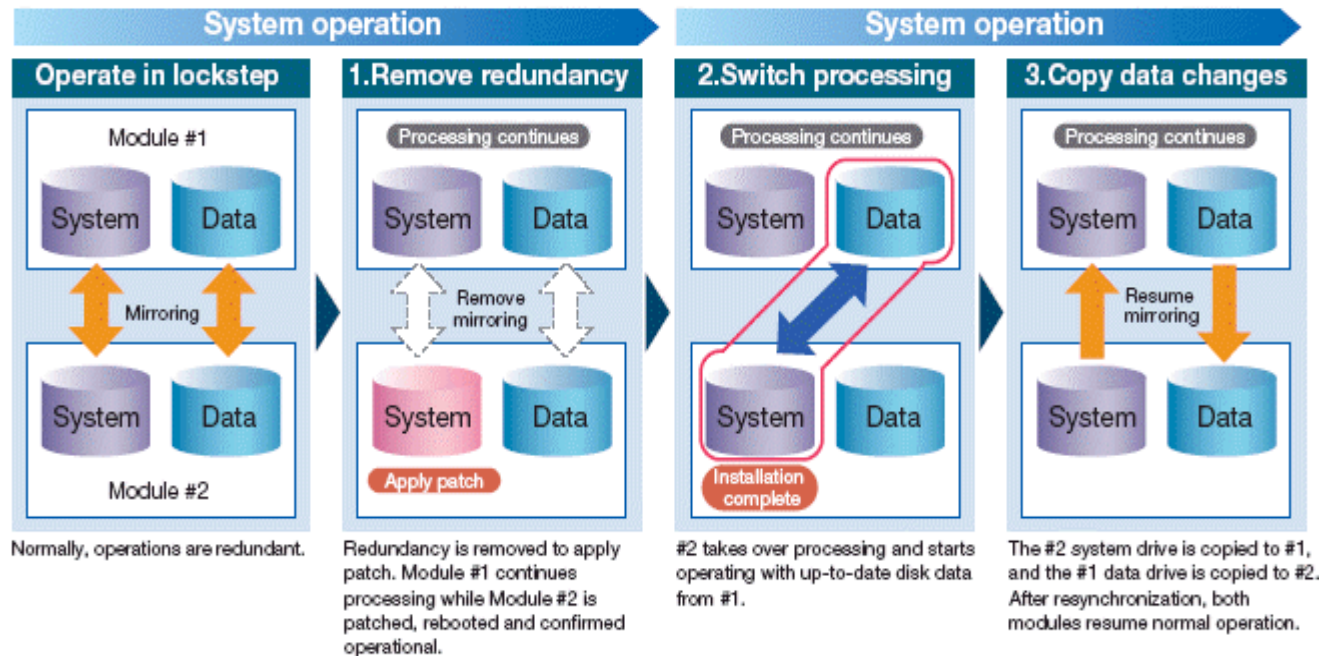
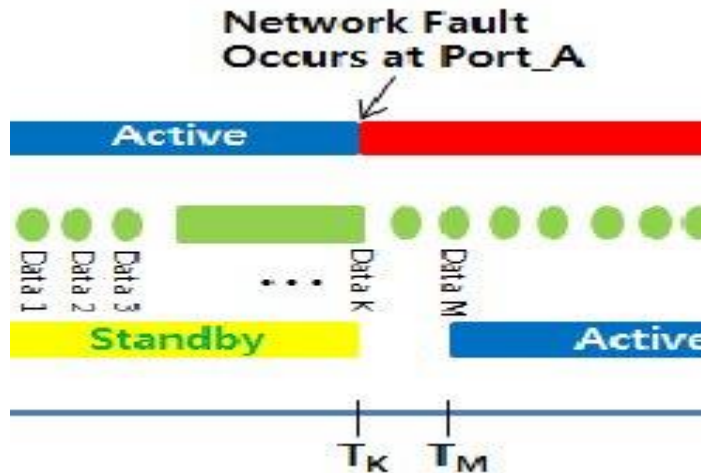
- 모든 동작 환경에서 항상 나타나지는 않음
- 일시적 결함의 유형
  - 단발적 - 한 번 발생하고 마는 유형
  - 간헐적 - 예상할 수 없는 시점에 수차례 발생하는 유형

### 정보(Information) redundancy

데이터를 복사해두거나, 비트 오류를 감지하고 복구할 수 있는 코드를 데이터에 덧붙이는 방식



# 결함 및 여분 설비 유형



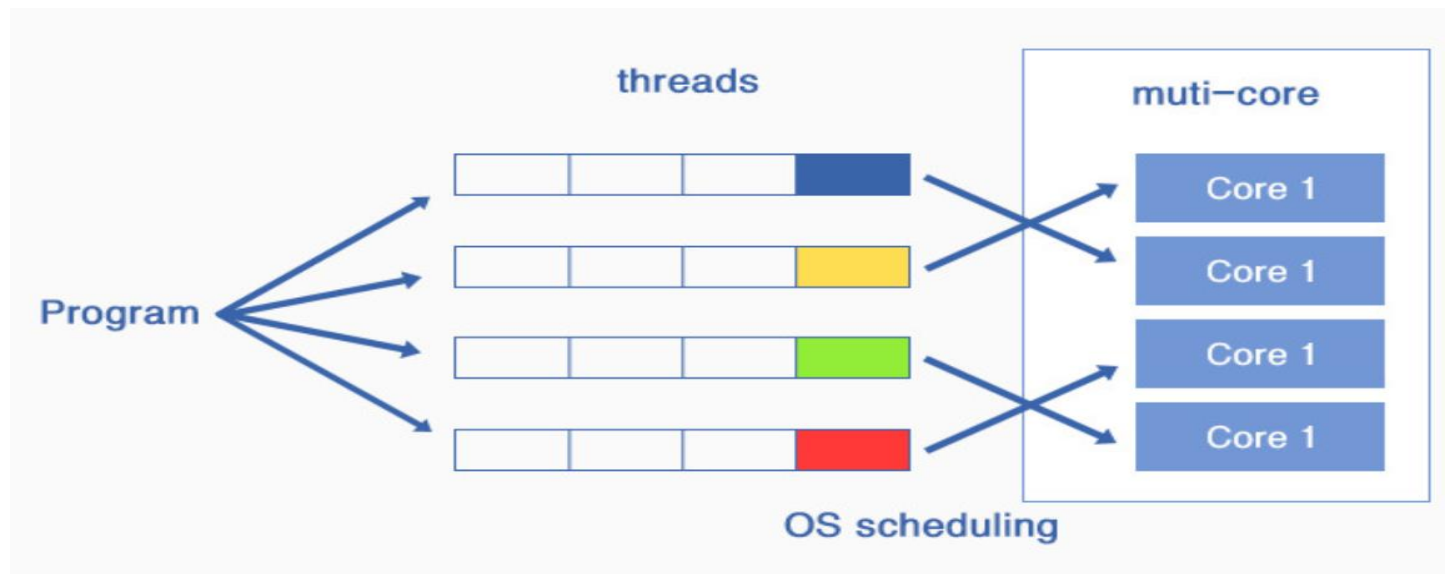
# 운영체제의 결함허용 기법

---

- 운영체제 내에는 결함허용을 위한 여러 가지의 기법들이 구현될 수 있다:
  - ✓ 프로세스 분리(process isolation)
  - ✓ 병행성 제어(concurrency)
  - ✓ 가상 기계(virtual machines)
    - 응용간 분리를 통해 다른 응용으로의 결함 전파 차단
    - 컴퓨터를 에뮬레이트(emulate)하는 SW로서, VM상에서 OS나 응용 프로그램을 설치하고 실행가능
    - VDI(Virtual Desktop Infrastructure)
  - ✓ 체크 포인트와 롤백(checkpoints and rollbacks)
    - 체크포인트로 롤백하면 그 지점부터 실행 재개 가능

## 2.6 멀티코어를 위한 운영체제 고려사항

- Multi-core (~16) -> many-core(~256)
- 매니코어 멀티코어 시스템을 위한 운영체제가 해결해야 할 난제
  - ✓ 멀티코어 처리 용량을 어떻게 하면 최대한 활용할 것인가?
  - ✓ 멀티코어를 담고 있는 단일 CPU 칩 내에 존재하는 상당 용량의 자원들을 어떻게 지능적으로 관리할 것인가?
- 멀티코어 시스템에서 지원하는 병렬성은 3가지 단계에 걸쳐 존재:



# 응용내에서의 병렬성 지원

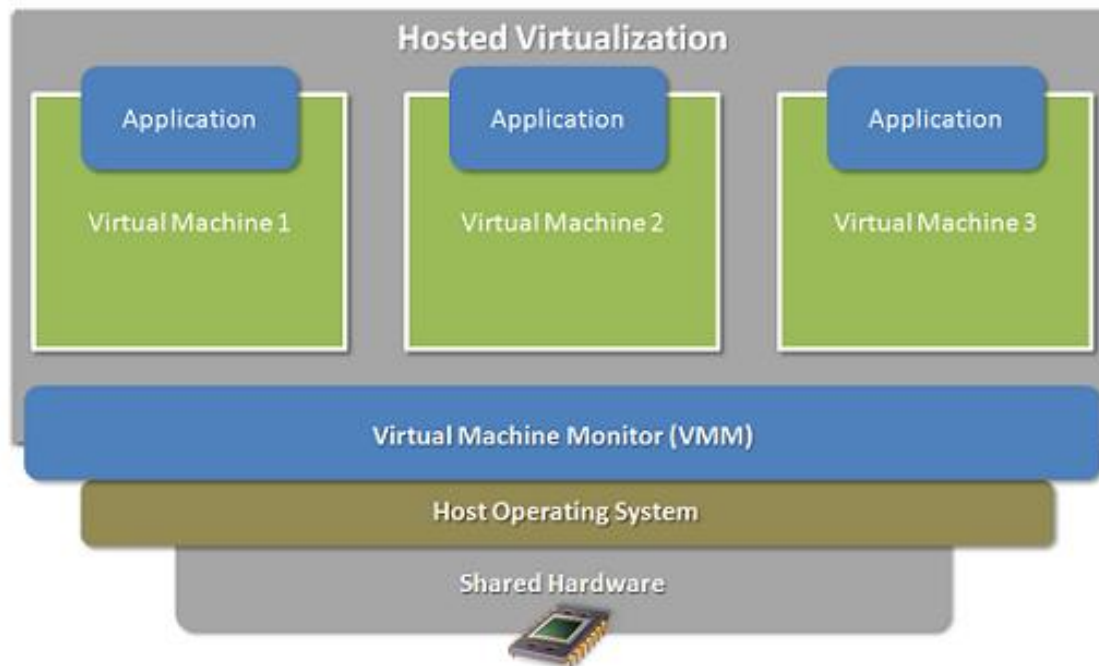
- 응용이 병렬성을 활용하려면 개발자가 자신의 코드 중 어느 부분들을 서로 독립적으로 병렬 실행시킬 수 있는지 반드시 결정해야만 함

## 응용 내 병렬성 지원도구: Grand Central Dispatch (GCD)

- Mac Os X 10.6부터 지원됨
- 일단 개발자에 의해 한 태스크가 여러 독립적 영역으로 분할되고 나면 GCD는 이들 분할된 태스크들이 서로 충돌을 일으키지 않도록 최대한 독립적으로 돌아갈 수 있도록 도와준다
- thread pool mechanism을 사용
- 함수 정의로 묶이지 않은 임의의 코드 블록(anonymous function)을 병행 실행 태스크로 지정할 수 있게 허용

# 가상 기계(VM) 방식

- 하나 또는 몇몇 코어들이 어떤 특정 프로세스를 위해서만 실행 되도록 고정시켜놓고, 그 처리기는 자신의 모든 성능을 그 프로세스를 위해서만 사용하도록 하는 방식
  - ✓ 태스크 문맥교환이나 프로세스 스케줄링 오버헤드 감소
- 멀티코어용 운영체제는 코어들을 응용에 할당하는 고차원 결정만 담당하면 된다.
  - ✓ 응용이 실행될 코어가 고정되기 때문에 세세한 자원 할당에 대한 개입을 많이 할 필요가 없기 때문



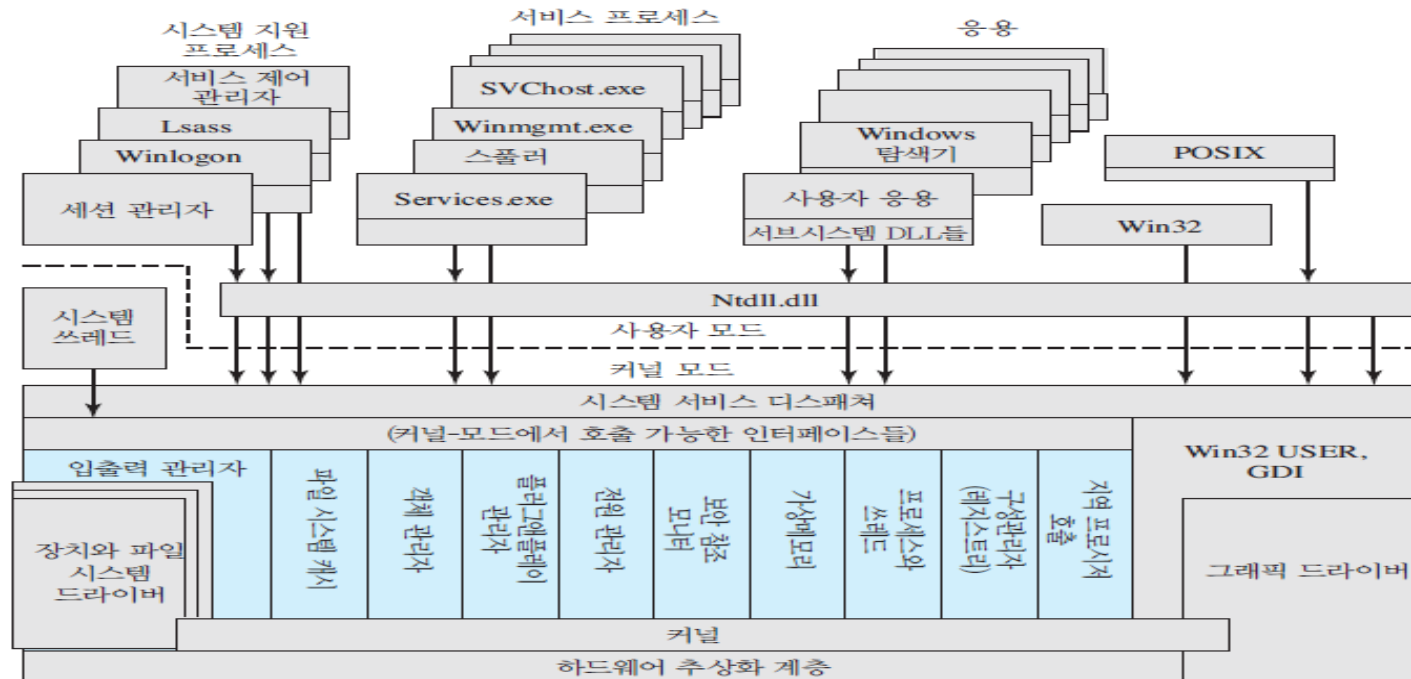
# Examples of Operating system

---

- A variety of operating systems
  - ✓ Microsoft Windows
  - ✓ UNIX
  - ✓ Linux
  - ✓ Android

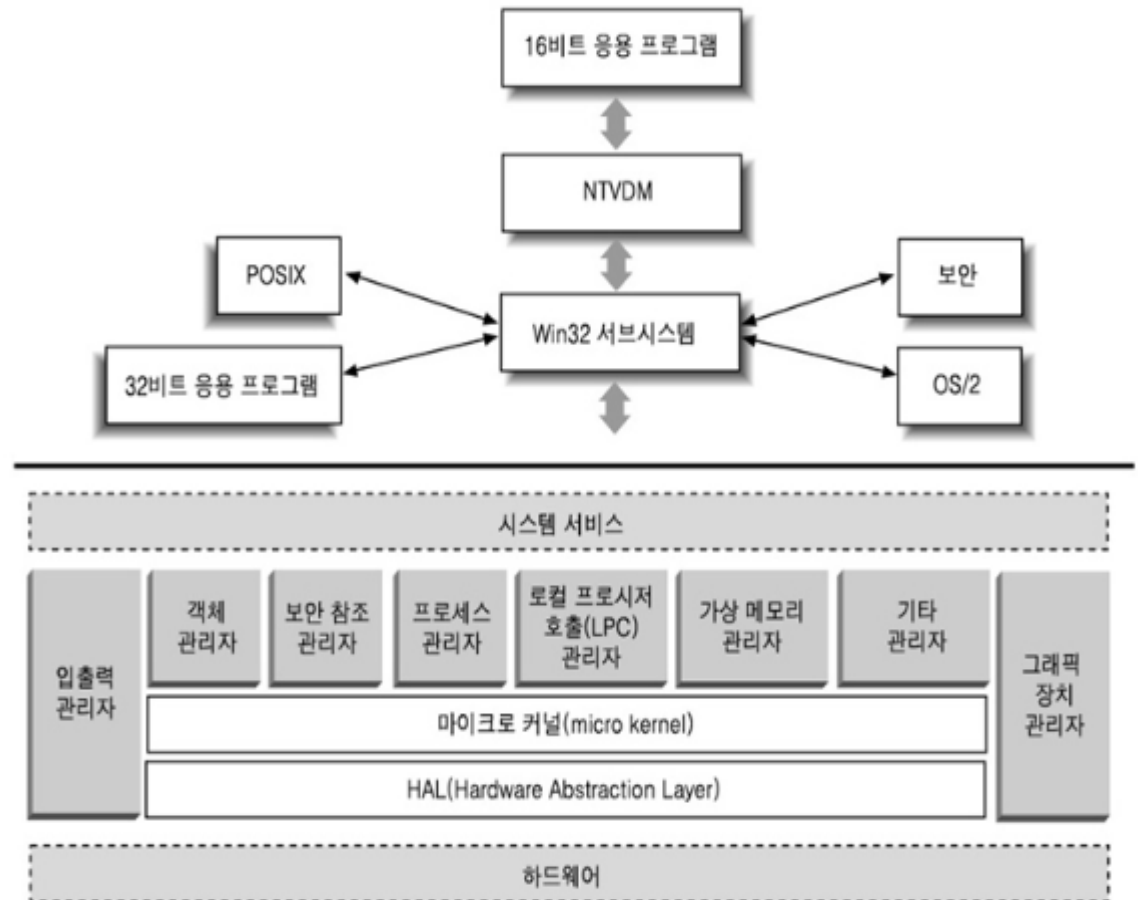
## 2.7 Windows 개요

- 모듈화는 상당한 융통성을 제공
- 다양한 형태의 하드웨어 플랫폼에서 수행 가능
- 다른 다양한 운영체제용으로 개발된 응용도 지원



Lsass = 지역 보안 인증 서버  
POSIX = 이식가능한 운영체제 인터페이스  
GDI = 그래픽 디바이스 인터페이스  
DLL = 동적 링크 라이브러리

## 2.7 Windows 개요





# 커널 모드 요소

---

- 수행부(Executive)
  - 핵심 운영체제 서비스를 포함하고 있다
  - 프로세스, 메모리관리, I/O, IPC
- 커널(Kernel)
  - 처리기의 실행을 제어한다
  - 스레드 스케줄링, 프로세스 교환, 인터럽트 처리
- 하드웨어 추상화 계층(HAL: Hardware Abstraction Layer)
  - 일반적인 하드웨어 명령 및 이에 대한 응답을 특정 플랫폼에 적합한 것으로 사상시켜, 플랫폼들의 하드웨어적 차이로부터 운영체제를 독립시킨다
- 장치 구동기(Device Drivers)
  - 수행부 기능 확장하는 역할을 동적 라이브러리(dynamic libraries)
- 창 다루기 및 그래픽스 시스템
  - 사용자 그래픽 인터페이스(GUI) 기능을 구현한다

# Windows의 사용자모드 프로세스들

- 사용자-모드 프로세스의 4가지 유형:

## 특별 시스템 프로세스

- 시스템을 관리하기 위해 필요한 사용자 모드 서비스

## 서비스 프로세스

- 프린터 스피일러, 사건 기록기(event logger), 장치 드라이버와 협동하는 사용자 모드 컴포넌트, 다양한 네트워크 서비스 등

## 프로그래밍 환경 서브시스템

- 다른 운영체제의 프로그램 개발 환경을 지원한다. 현재는 Win32(Windows API)와 POSIX(Portable OS Interface)를 지원
- 다른 OS에서 작성된 프로그램은 윈도우에서 re-compile만 하면 소스 변경없이 재사용

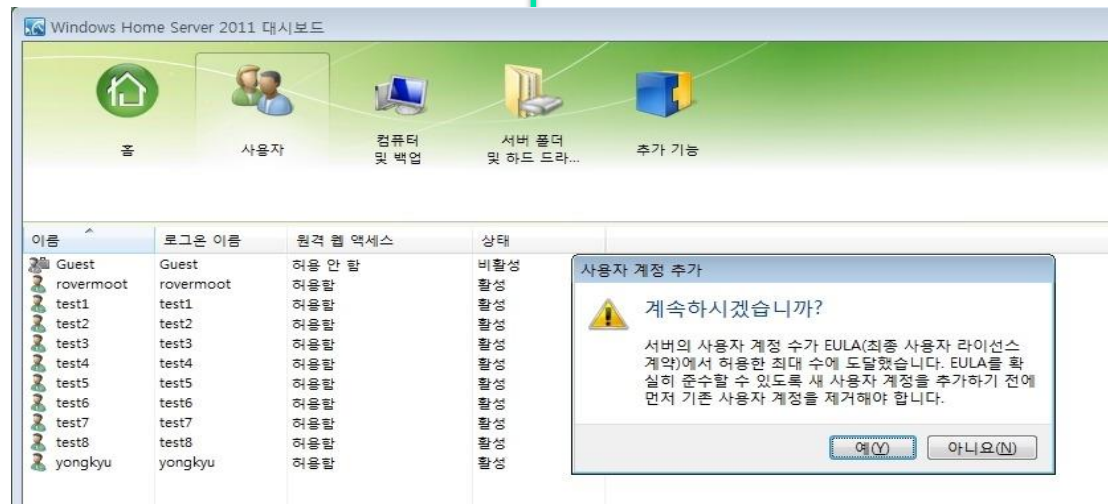
## 사용자 응용

- 실행 파일(EXE 파일)과 DLL이 이 유형에 속하는데, EXE 파일과 DLL 파일은 사용자가 시스템을 사용할 수 있도록 해준다

# Windows의 클라이언트/서버 모델

- Windows OS 서비스와 환경 서비스 시스템, 그리고 응용들은 클라이언트/서버 모델을 사용하여 구축됨
- 분산 시스템에서는 매우 흔한 구조이지만, Windows 같은 단일 시스템 내부에서도 사용될 수 있음
- 프로세스들은 RPC를 통해 서로 연결된다
- 서비스 요청 메시지를 보내면, 그 메시지는 수행부를 통해 적절한 서버로 전달된 후, 결과를 클라이언트로 전달

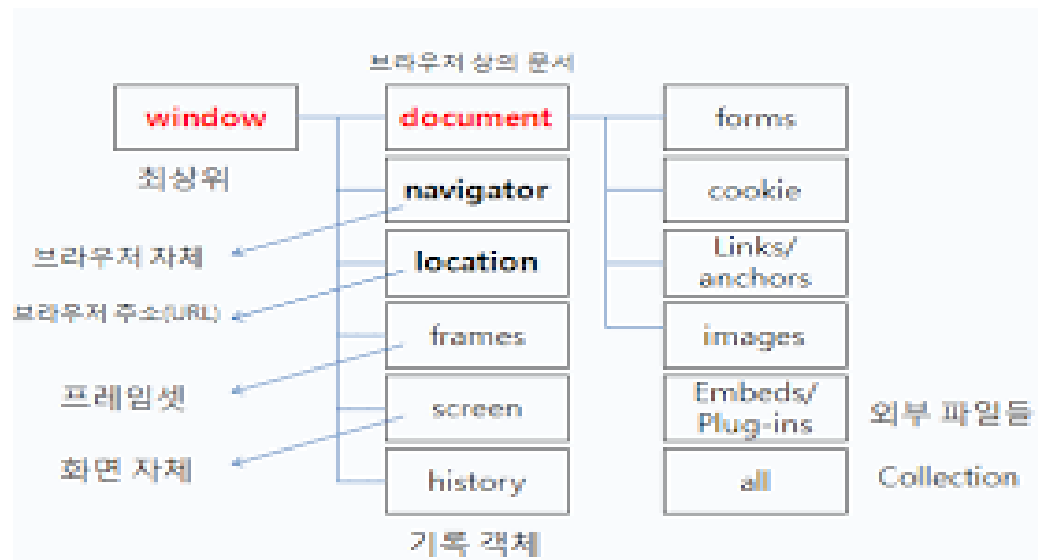
- Advantages:
  - 수행부 단순화
  - 신뢰성 향상
  - 응용성을 제약하지 않으면서 RPC를 통해 응용이 서비스들과 통신을 할 수 있도록 하는 일관된 수단을 제공
  - 분산 컴퓨팅에 적합한 기반을 제공



- Windows의 두 가지 중요한 특징은 쓰레드 및 SMP를 지원한다는 점
  - ✓ 운영체제 루틴들은 이용 가능한 어느 처리기 상에서도 수행될 수 있으며, 서로 다른 루틴들은 서로 다른 처리기 상에서 동시에 수행될 수 있다
  - ✓ Microsoft Windows 에서는 단일 프로세스 안에서 다수의 쓰레드를 사용할 수 있다. 동일 프로세스 내의 다수의 쓰레드들은 다른 처리기 상에서 동시에 수행 가능하다
  - ✓ 서버 프로세스는 하나 이상의 클라이언트로부터 도착한 요청을 동시에 처리하기 위해 다수의 쓰레드를 사용할 수 있다
  - ✓ Windows는 프로세스들 간의 데이터와 자원의 공유, 유연한 프로세스 간 통신(IPC) 기능 등을 위한 기법을 제공한다

# Windows 객체

- 캡슐화
  - ✓ 객체는 하나 이상의 데이터 아이템과 그 데이터를 액세스할 수 있는 프로시저(서비스)를 포함
- 객체 클래스와 인스턴스
  - ✓ 객체 클래스는 객체의 특성을 정의하는 템플릿(template)
  - ✓ OS는 객체의 특정 인스턴스를 생성 가능
- 상속
  - ✓ 수행부의 객체 확장을 지원
- 다형화
  - ✓ 윈도우는 임의 타입의 객체를 처리하기 위해 공통의 API 집합을 사용



## 2.8 UNIX

---

- 운영체제 소프트웨어의 의해 하드웨어는 감추어짐
- 다양한 사용자 서비스와 인터페이스 제공
  - ✓ Shell
  - ✓ 대부분의 UNIX/LINUX는 C-language 로 구현

# Traditional UNIX

- Hardware is surrounded by the operating system
- Operating system is called the system kernel
- Comes with a number of user services and interfaces
  - ✓ Shell
  - ✓ Components of the C compiler
- 커널
  - ✓ 사용자 프로그램은 직접 또는 라이브러리 프로그램을 통해 OS 서비스 호출
  - ✓ 시스템 호출은 사용자와 OS간의 인터페이스
  - ✓ OS는 HW를 제어하는 기본 루틴들을 포함하며, 시스템은 프로세스 제어 파트와 파일 관리 및 입출력 파트로 구분됨

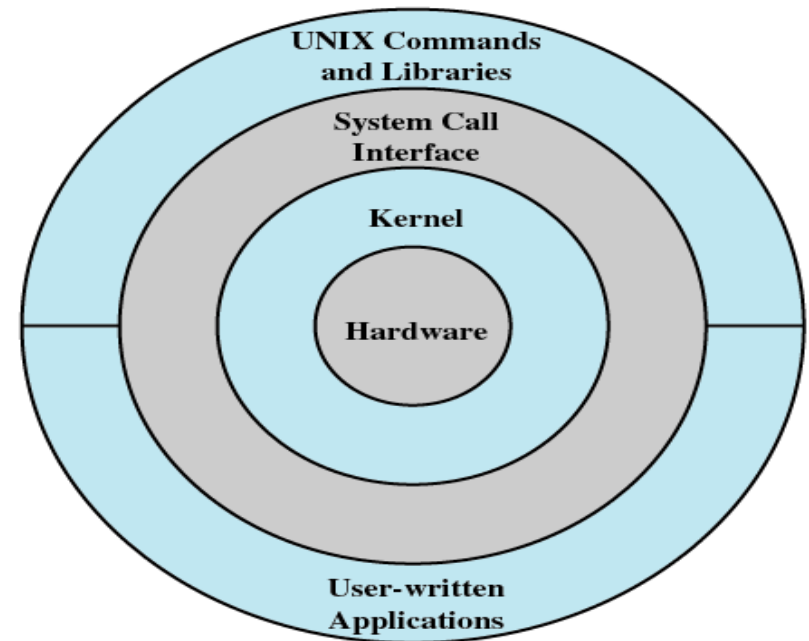
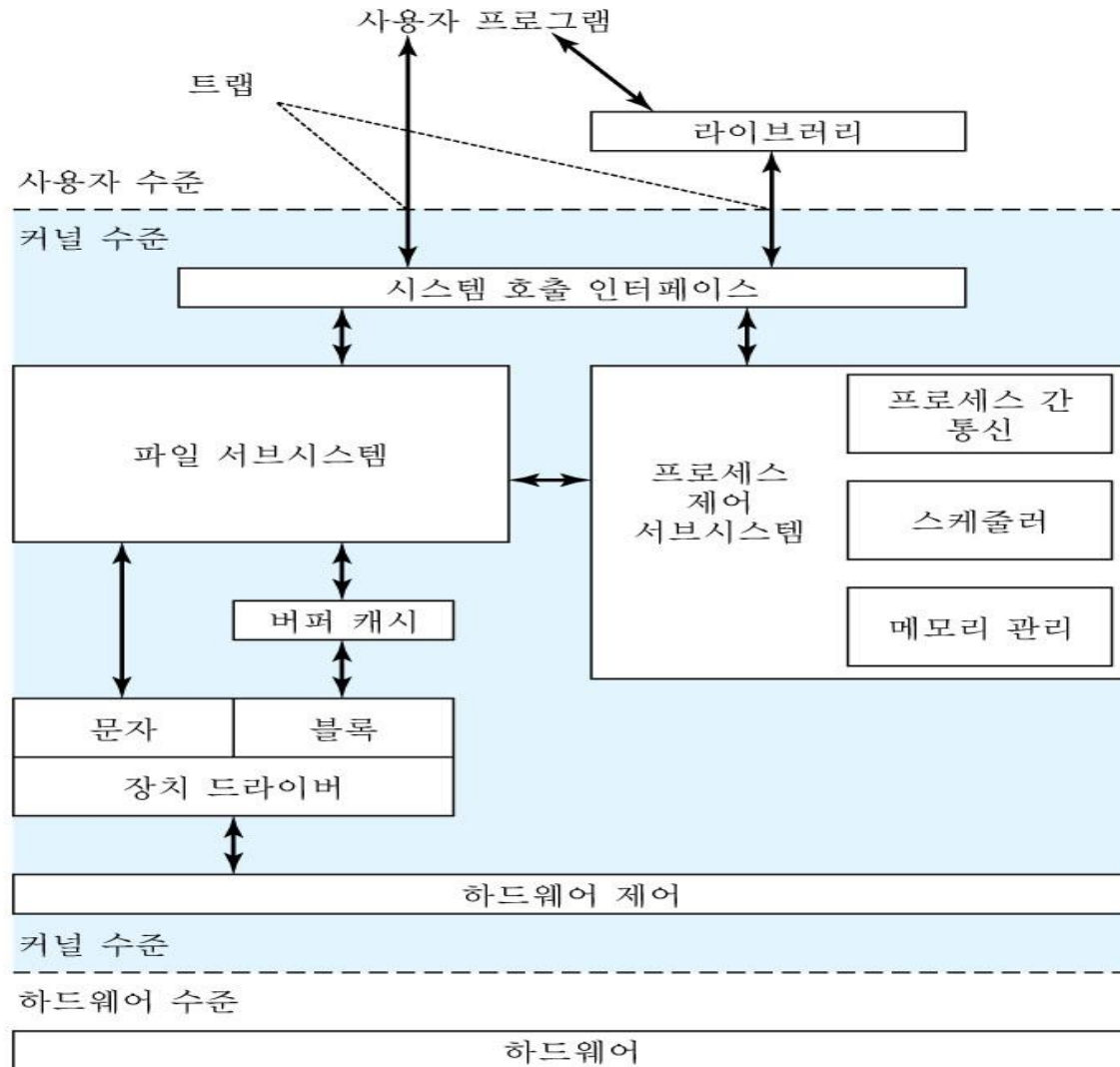


Figure 2.14 General UNIX Architecture

# 전통적인 UNIX 커널





# Modern UNIX Kernel

## ■ Examples

- ✓ System V Release 4 (SVR4), Solaris 9, 4.4BSD, Linux

## ■ UNIX architecture

- ✓ 모듈화된 기능을 이용하여, OS 프로세스에 필요한 기능과 서비스 제공

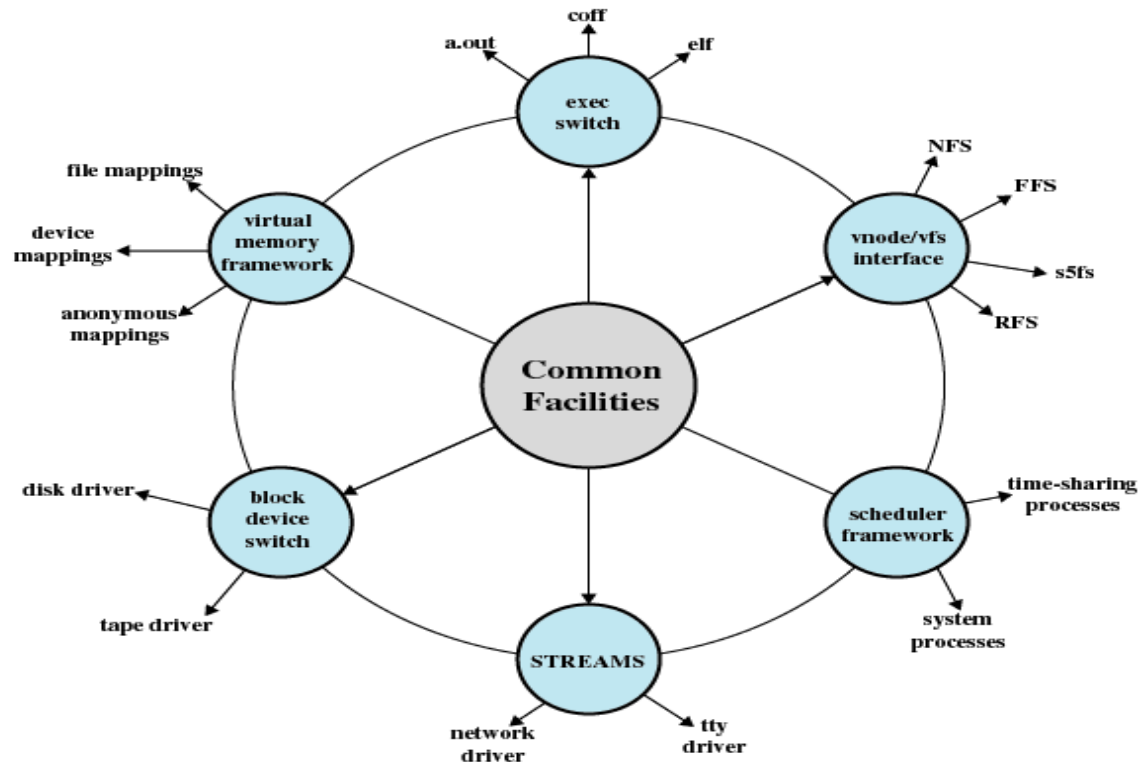


Figure 2.16 Modern UNIX Kernel [VAHA96]

# System V Release 4(SVR4)

---

- AT&T와 Sun Microsystems가 공동으로 개발
- SVR3, 4.3BSD, Microsoft Xenix System V, SunOS 등의 특징들을 통합
- 새로 추가된 특징들:
  - 실시간 처리(real-time processing) 지원
  - 프로세스 스케줄링 클래스(process scheduling classes)
  - 동적 할당 자료구조(dynamically allocated data structures)
  - 가상메모리 관리(virtual memory management)
  - 가상 파일시스템(virtual file system)
  - 선점형 커널(preemptive kernel )

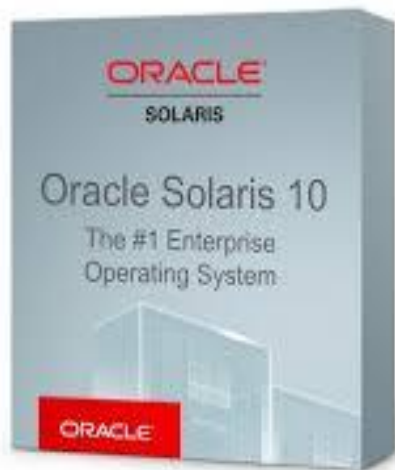
# BSD(Berkeley Software Distribution)

---

- 4.xBSD는 교육기관에서 널리 사용되었으며, 많은 상업용 UNIX 제품의 기반이 됨
- BSD의 설계와 구현을 담당하였던 조직은 4.4BSD를 최종 버전으로 배포한 후 해체
  - 다음과 같은 수많은 기능 향상을 통해 4.3BSD의 기능을 크게 개선
    - 새로운 가상메모리 시스템
    - 커널 구조의 변경
- FreeBSD
  - BSD의 여러 버전 중 가장 많이 사용되고 있고 문서화가 잘된 버전
  - 인터넷-기반 서버와 방화벽 구축 시 애용되었었음
  - 수많은 임베디드 시스템에서도 사용됨
  - MAC OS X도 FreeBSD 5.0과 Mach 3.0 마이크로 커널에 기반함

# Solaris 10

- Sun사에서 개발한 SVR4 기반 UNIX로서 버전 10이 가장 최신
- SVR4의 모든 기능을 제공하며, 이에 덧붙여 다음과 같은 기능도 추가됨
  - 완전 선점 가능한 멀티쓰레드 커널
  - SMP의 완전한 지원
  - 파일 시스템에 대한 객체 지향 인터페이스
- 현재 가장 널리 사용되며 가장 성공적인 상업용 UNIX라는 평가



## 2.10 Linux

---

- UNIX 의 변형으로 IBM PC(Intel 80386) 구조에 맞추어짐
  - ✓ 무료(Free) SW 패키지 이용가능
  - ✓ Linus Torvalds, 1991년 인터넷배포
- 마이크로커널 방식을 사용하지 않음
- 동적 적재 가능한 모듈(loadable module)의 집합
  - ✓ Linux 모듈들은 요구에 따라 자동으로 적재되고 해제됨
  - ✓ 동적 링킹 (dynamic linking)
    - 커널 모듈은 커널이 이미 메모리에 올라와 수행되는 동안에도 적재되어 커널에 링크될 수 있다
  - ✓ 스택 가능한 모듈 (Stackable module)
    - 모듈은 계층적으로 정렬된다

# Linux 커널 모듈

- 아래 리눅스 구조도는 단지 2개의 모듈(FAT, VFAT) 이 적재된 후에 형성된 커널 모듈의 리스트를 보여줌
  - ✓ **VFAT(Virtual File Allocation Table)** 은 기존의 **FAT** 가 긴 이름을 처리하지 못했던 (~8 char) 단점을 보완하기 위해 **Win95** 이후에 추가된 **OS 기능 (-255 char)**
  - ✓ **FAT**는 하드디스크에 있는 파일 위치 정보를 가지고 있는 **table**

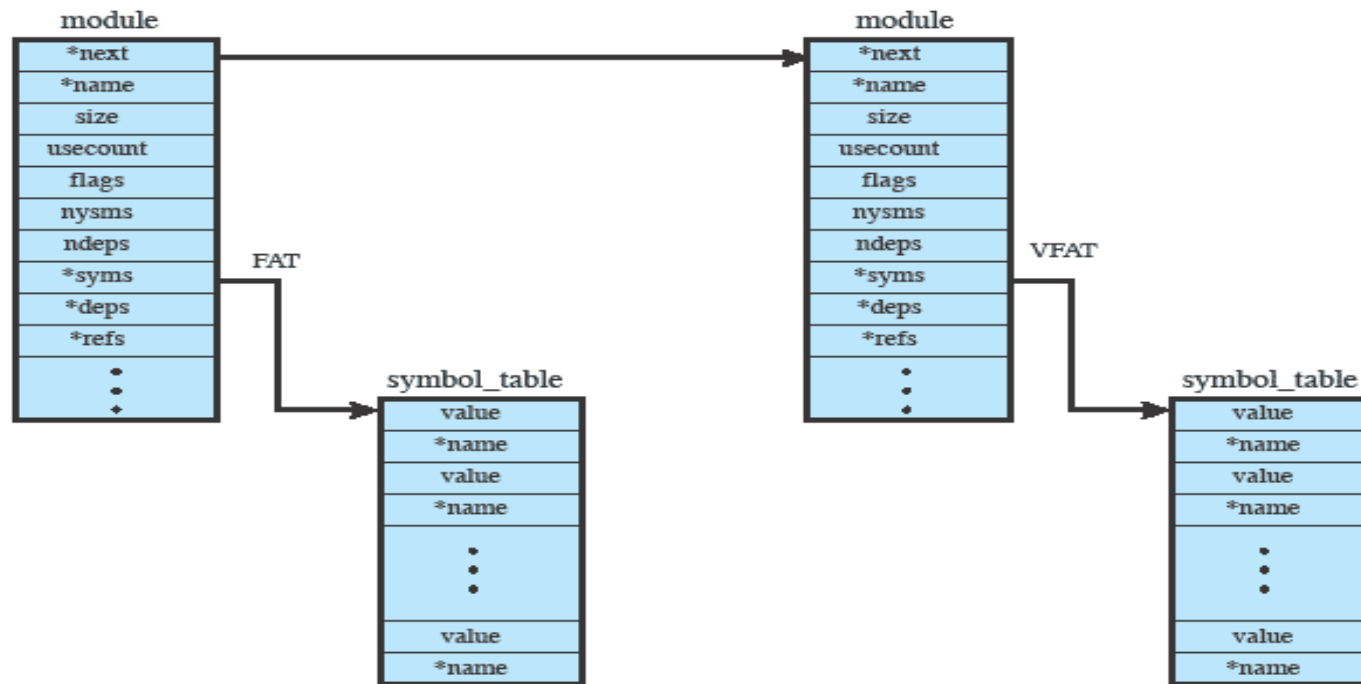
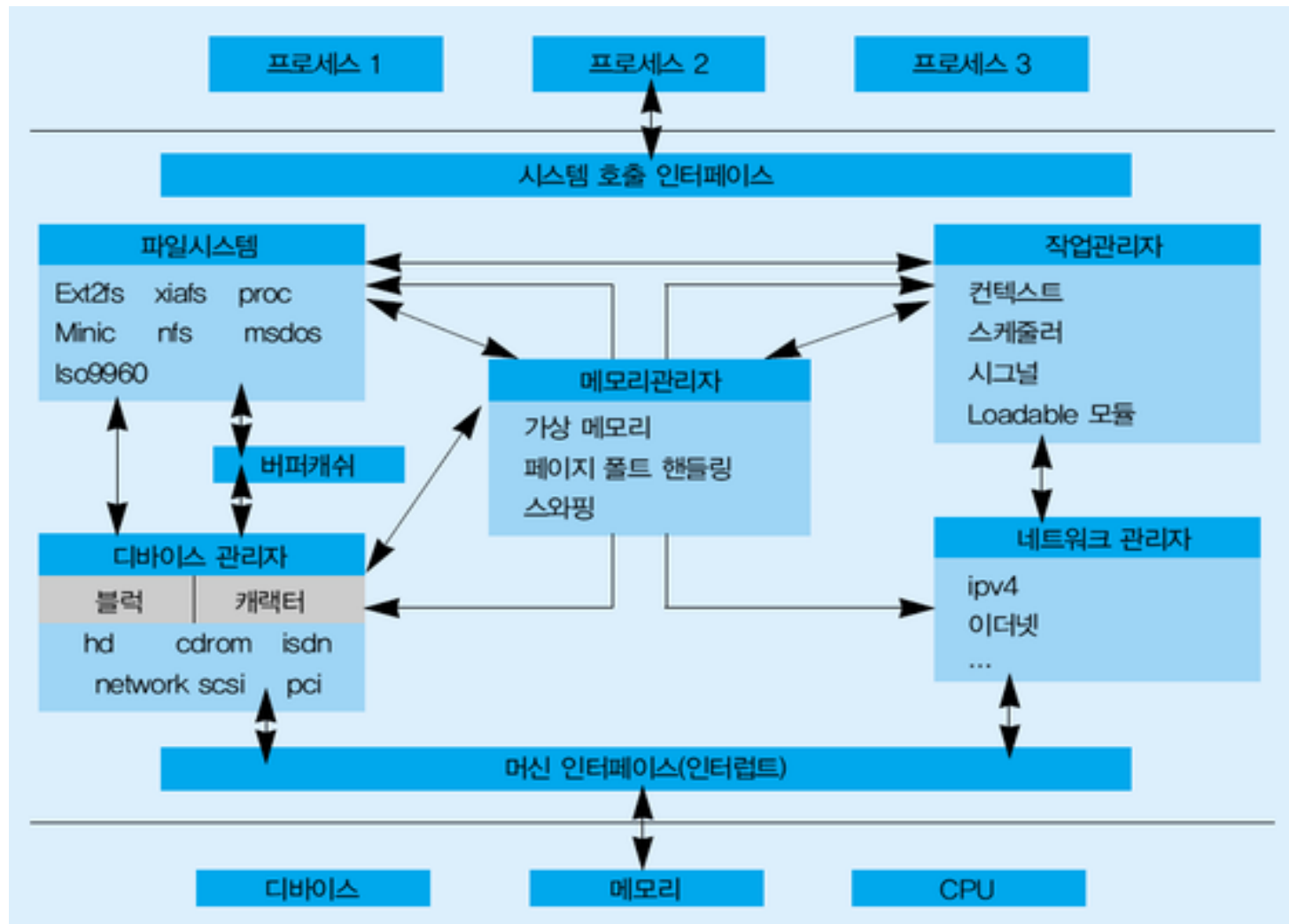


Figure 2.17 Example List of Linux Kernel Modules

# Linux 커널 구성요소



# Linux 커널 구성요소

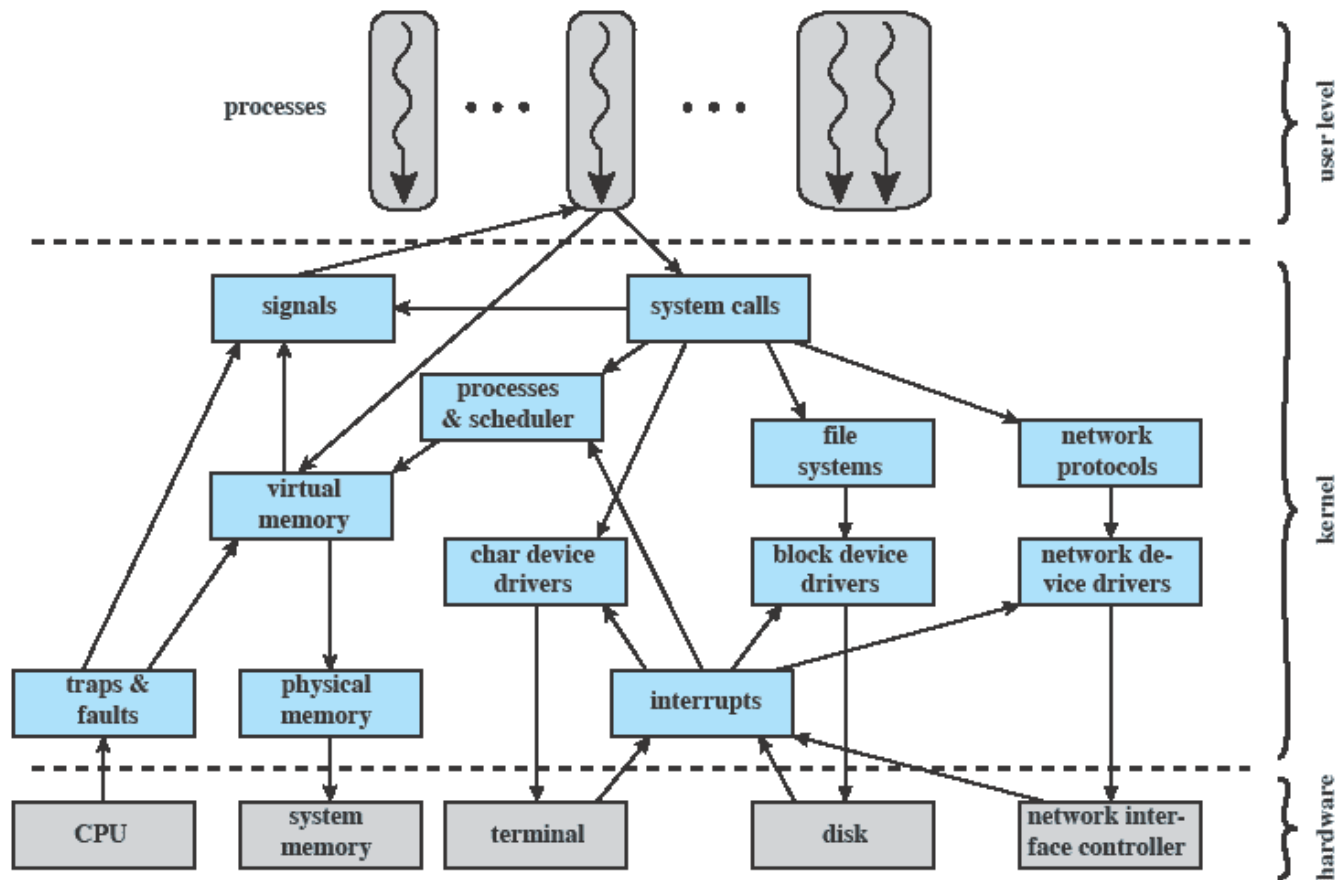


Figure 2.18 Linux Kernel Components



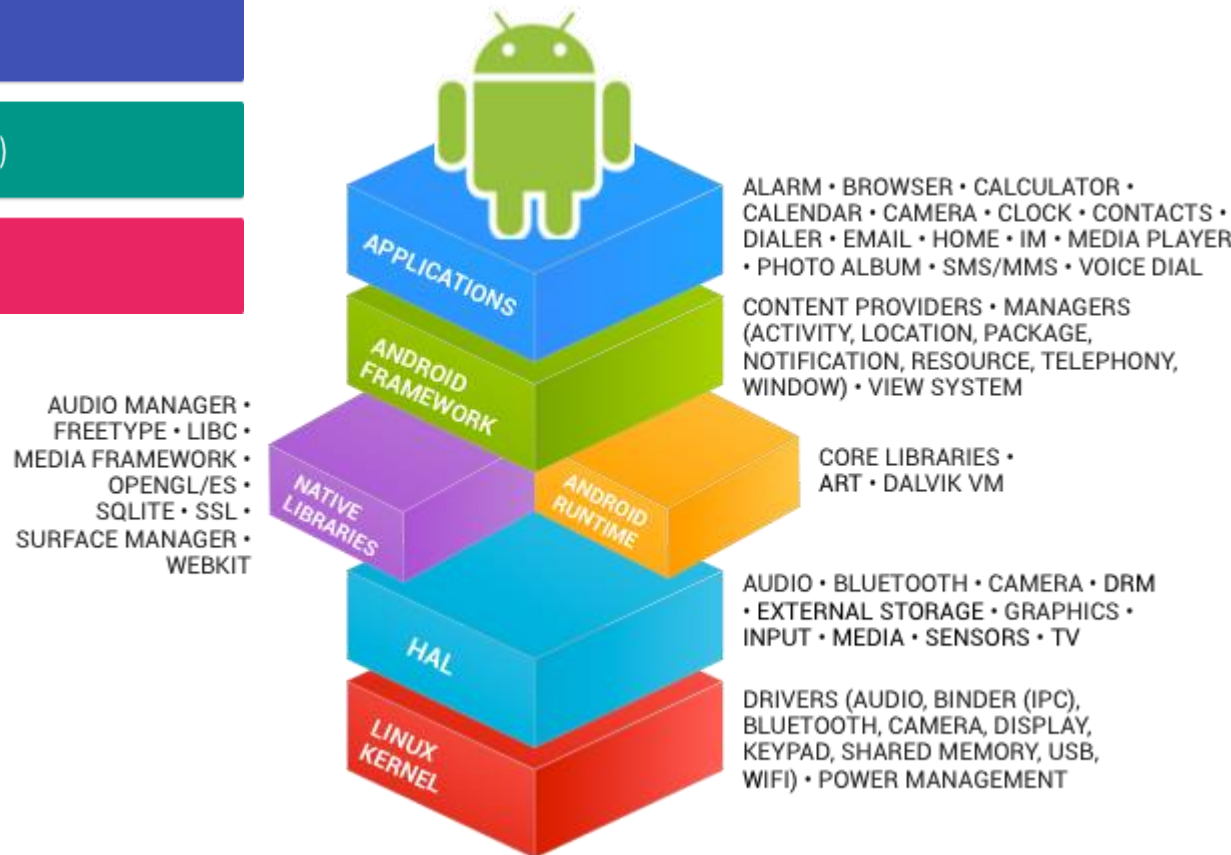
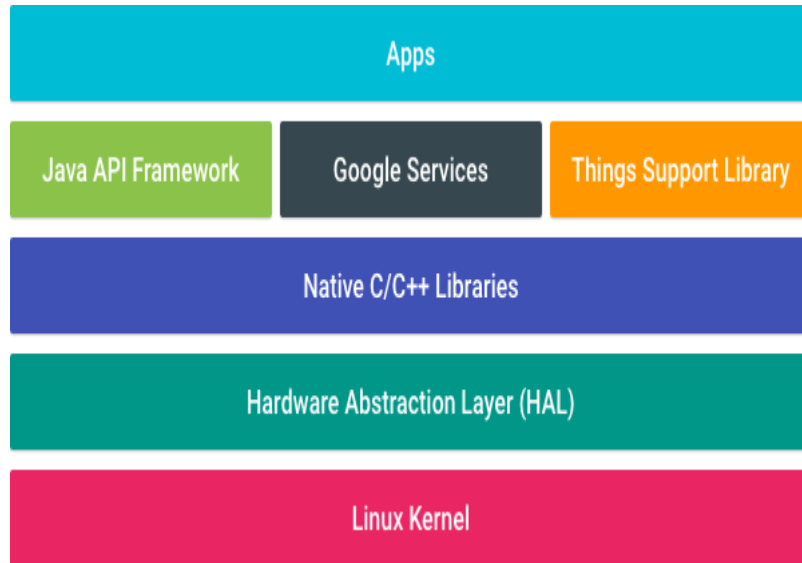
## 2.11 Android 운영체제

---

- 스마트폰이나 태블릿 컴퓨터 같은 터치스크린 모바일 장치 전용으로 설계된, Linux-기반 시스템
- 가장 인기 있는 모바일 OS
- Android 사에 의해 개발되었지만, 이후 2005년에 구글이 Android 사를 인수
- 첫 번째 상업용 버전인 Android 1.0은 2008년에 출시
- 현재 사물인터넷(IoT) OS로 널리 활용
- 최신 버전은 Android 8.1 (오레오)
- OHA(Open Handset Alliance)가 개방 플랫폼의 하나인 Android 운영체제 출시를 책임지고 있다
- Android의 소스가 공개되어 있다는 점은 Android가 성공할 수 있었던 요인 중에 매우 중요한 비중을 차지

# Android 소프트웨어 구조

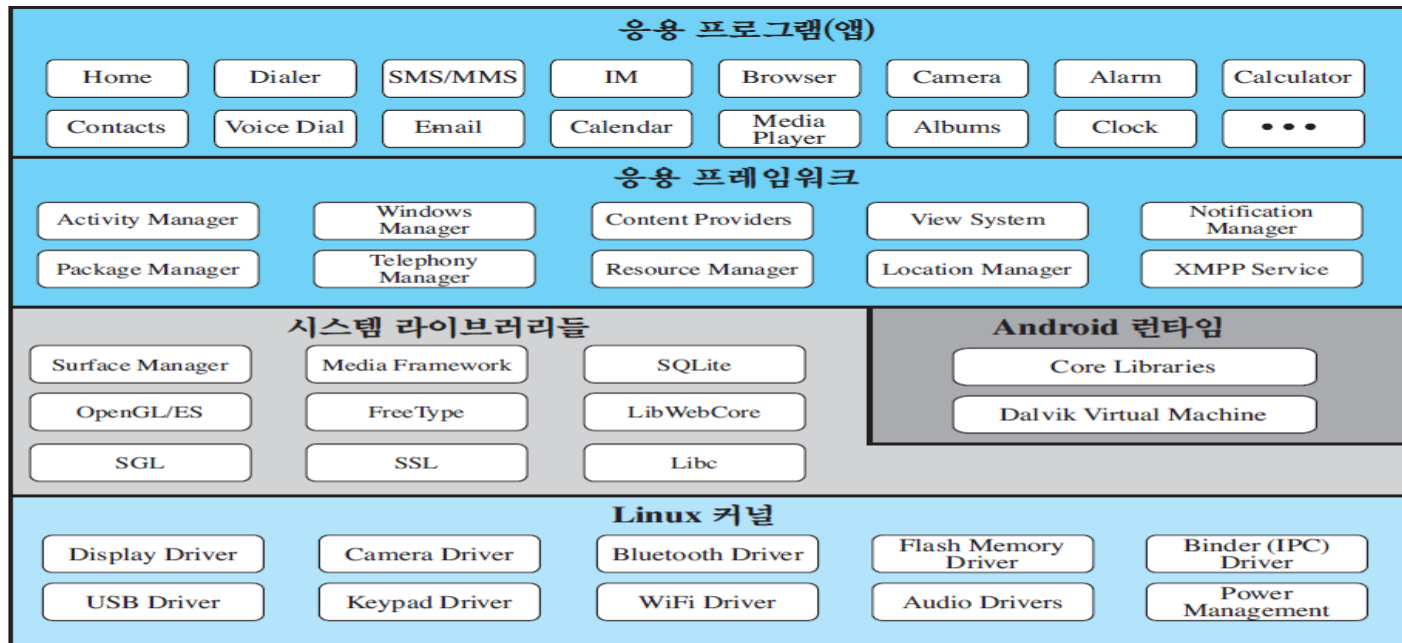
- 안드로이드는 OS 라기보다 SW Stack
  - ✓ OS 커널, 미들웨어, 핵심 응용들로 구성



# Android 소프트웨어 구조

## ■ 안드로이드는 OS 라기보다 SW Stack

- ✓ OS 커널, 미들웨어, 핵심 응용들로 구성
- ✓ 응용프레임워크는 앱 개발시 표준 API를 통해 접근할 수 있는 고수준 빌딩 블록들을 제공



구현 언어:

- Applications, Application Framework: Java
- System Libraries, Android Runtime: C and C+
- Linux Kernel: C

# Summary

---

- Operating System Objectives
- Operating System Definitions
- Evolution of Operating Systems
  - ✓ serial processing, batch system, time sharing system
- Major Achievements and New Technologies
- Existing Operating Systems
  - ✓ Microsoft Windows
  - ✓ Traditional UNIX Systems
  - ✓ Modern UNIX Systems
  - ✓ Linux
  - ✓ Android