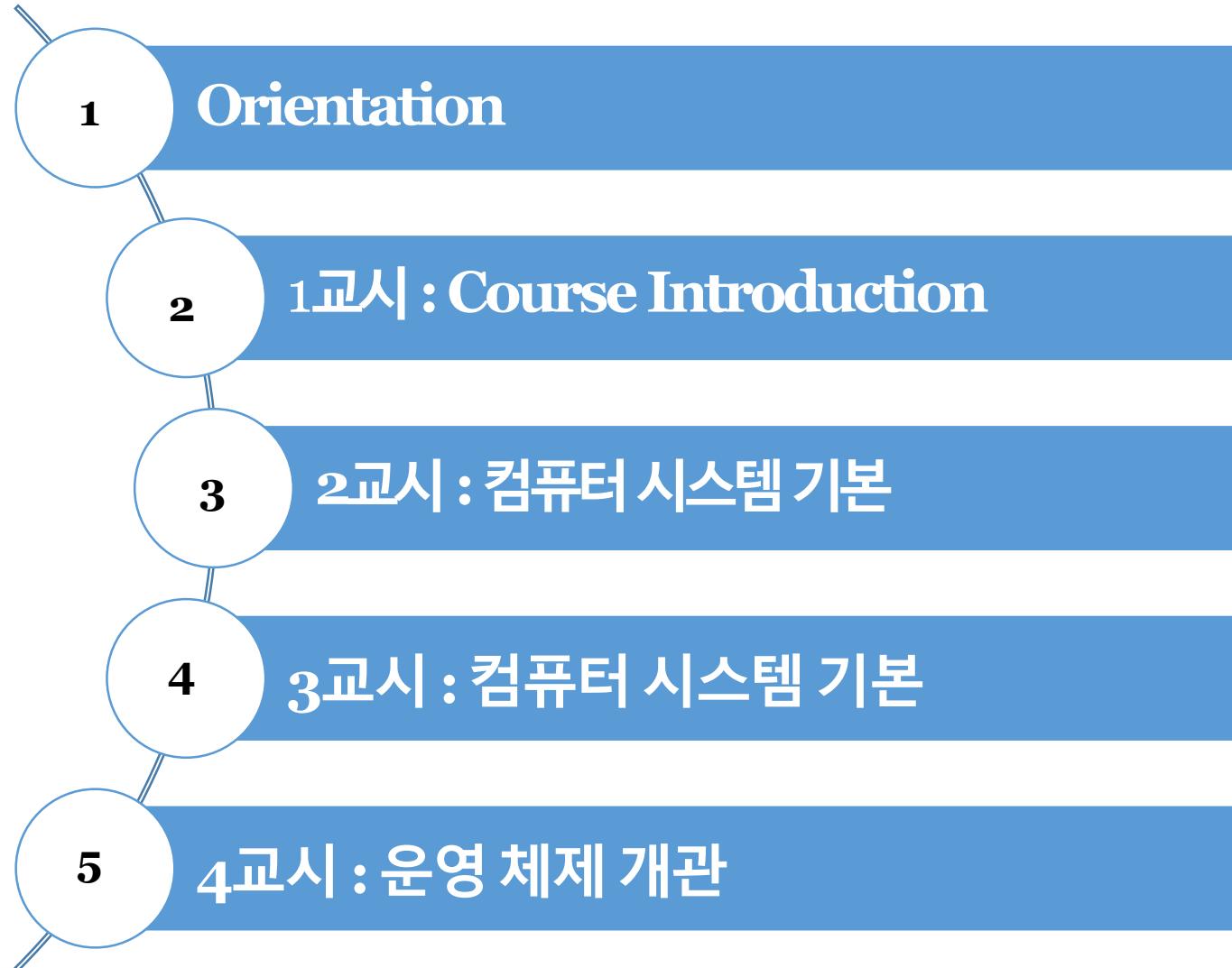


『 1일차 』: 오전

- ◆ 훈련과정명 : OS 기본
- ◆ 훈련기간 : 2023.05.30 ~ 2023.06.02



목차



Orientation

1교시 :

Course Introduction





학습목표

- 이 워크샵에서는 강사 소개, Course Introduction, Course Structure, Pre-Requisites에 대해 알 수 있습니다.

눈높이 체크

- 직무와 능력단위를 알고 계신가요?



1. a lecturer

강사 소개

- 주)엘케이랩 이사 신사업본부장 | 전 인천대학교/가천대학교/ 컴퓨터공학과 겸임교수 | Solutions Architect | Software Architect | NCS 기업 컨설팅 컨설턴트 전문위원 | Smart Factory 공급, HR 솔루션 컨설팅·공급, 산업처리공정 제어장비 제조, Kdata 데이터 가공 등록, IoT Integration, AI 딥러닝 기반 빅 데이터 컨설팅 및 응용SW개발
- E-mail: onlooker2zip@naver.com/dkkim@lklab.org
- Tel: 010-9591-1401

Trainer

- SE | 프로젝트관리기법(ITPM) | Agile & TDD | IoT | Big Data | Machine Learning | AI | Cloud Computing Services | SaaS | PaaS | Docker | Kubernetes



2. Course Introduction

운영체제 약사

- 세계의 운영체제(OS) 시장은 2022년 451억 7000만 달러에서 2023년에는 461억 2000만 달러로, 연평균 성장률(CAGR) 2.1%로 확대할 것으로 예상된다. 러시아·우크라이나 전쟁은 적어도 단기적으로는 COVID-19 팬데믹으로부터의 세계 경제 회복의 가능성을 혼란시켰다. 이 두 국가간 전쟁은 복수의 국가에 대한 경제 제재, 상품 가격의 상승, 공급망 혼란으로 이어지며 상품 및 서비스 전체에 인플레이션을 일으키며, 전 세계의 많은 시장에 영향을 미치고 있으며, 운영체제 시장은 2027년에 CAGR 1.7%로 493억 4000만 달러로 성장할 것으로 예측된다.



2. Course Introduction

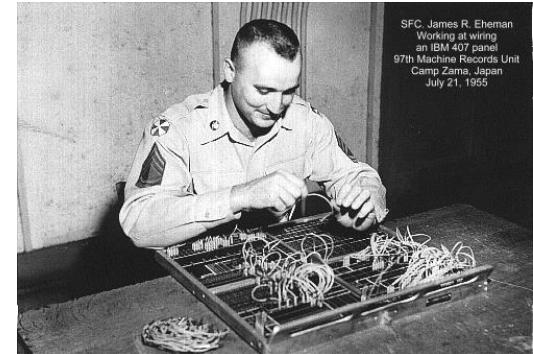
운영체제 약사

- '운영체제란? 컴퓨터 자원을 관리하고, 작업을 스케줄링하고, 사용자에게 편리한 인터페이스를 제공하는 시스템 소프트웨어이다.'
- 그런데 에니악 같은 초창기 컴퓨터는 손으로 스위치를 올렸다 내렸다 해가며 프로그램을 입력했다는데, 그런 컴퓨터도 운영체제를 가졌었을까?
- 운영체제를 가지지 않은 그런 초창기 기계를 벌거벗은 컴퓨터, 영어로는 **bare computer**라 한다. 프로그래머는 속을 봐야만 했다. 누산기가 몇 개인지, 어떤 스위치와 전선이 무슨 역할을 하는 지 등을 알아야만 했고, 컴퓨터를 물리적으로 조작해야만 하였다.

2. Course Introduction

운영체제 약사

- 1940년대 말과 1950년대에 걸쳐 큰 변화가 온다. 컴퓨터의 아키텍처가 한 단계 진화한 것이다. 이전 컴퓨터에서는 연산장치와 메모리가 한데 붙어 있었던 반면, 이제는 연산을 담당하는 'CPU(중앙처리장치)'와 데이터를 저장하는 '메모리'가 구분된다. 게다가 메모리에 데이터뿐 아니라 프로그램도 함께 저장할 수 있게 된다. 이제 프로그램을 외부 장치가 아니라 메모리 내부에 적재하는 '저장 프로그램 stored program' 방식이 컴퓨터 아키텍처의 표준이 된 것이다. 또한 프로그램을 기록하는 매체에 큰 변혁이 온다. 플러그보드를 펀치 카드가 punch card 대체한 것이다.





2. Course Introduction

운영체제 약사

- 이제 프로그래머는 빈 카드 묶음을 구입한 후, 카드 천공기를 **card puncher** 이용하여 자신이 짠 프로그램을 카드에 기록한 후, 카드 묶음을 들고 컴퓨터 센터에 가면 된다. 컴퓨터는 이 카드 묶음을 카드 판독기로 읽어 메모리에 적재한 후, 프로그램 실행을 시작한다. 이제 기호와 수를 사용하여 프로그램을 짜고 그것을 카드에 기록하는 '추상화된' 프로그래밍 방식으로 발전한 셈이다.
- 이런 컴퓨터는 여기저기에서 제작된다. 영국 캠브리지 대학에서 에드삭, 미국 펜실베이니아 대학에서 에니악의 후속 모델인 에드박이 제작된다. 또한 유니백이나 IBM 701과 같은 상업용 컴퓨터가 시장에 나온다. 이러한 컴퓨터는 메인프레임이라는 **mainframe** 이름을 얻는다.



2. Course Introduction

운영체제 약사

- 원시적인 운영체제가 태동한다. 사람들은 거의 모든 프로그램이 공통적으로 필요로 하는 부분에 눈길을 준다. 가장 대표적인 부분은 입출력장치를 구동하는 것이었다. 컴퓨터 센터는 이런 부분을 서브루틴으로 작성하여 메모리의 특정 영역에 상주시켜 둔다. 이러한 서브루틴 집합을 입출력 제어 시스템이라 IOCS 불렀다. 프로그래머는 필요한 기능을 IOCS가 제공하면, 그 부분은 직접 작성하지 않고 서브루틴을 호출하여 사용할 수 있게 되었다.
- 결과적으로 프로그래밍에 걸리는 시간이 크게 줄었다. 또한 기계 내부 상태를 외부로 알린다거나, 프로그램 실행 도중 오류가 났을 때 그 상태의 메모리 내용을 저장하여, 오류(버그)를 찾는데 도움을 주는 기능, 비용 산출을 위해 프로그램이 사용한 장비와 사용 시간을 기록하는 기능 등이 추가된다. 이런 원시적인 운영체제를 모니터라 monitor 불렀다. 모니터 덕분에 프로그래머는 기계 내부를 모두 볼 필요는 없어졌다. 하지만 여전히 봐야 할 곳이 많이 남아 있었다.



2. Course Introduction

운영체제 약사

- 프로그램 처리 과정에서 사람을 들어내고, 그 자리에 어떤 프로그램을 배치해야만 한다. 이 프로그램이 '운영체제' operating system(OS)이다. 최초의 운영체제는 당시 가장 큰 컴퓨터 회사인 IBM이 아니라, 자동차 회사인 제너럴모터스가 NASA와 함께 공동 개발한다. 제너럴모터스는 보유하고 있던 IBM 704 컴퓨터를 위해 1956년에 **GM-NAA I/O**라는 운영체제를 자체 개발한다. 1959년에는, IBM 컴퓨터 사용자 그룹인 SHARE가 GM-NAAI/O를 개선한 **SOSSHARE Operating System**을 발표한다.
- 프로그래머는 운영체제로 인해 하루에도 여러 번 실행 결과를 받아 볼 수 있게 되었고 디버깅에 드는 시간도 크게 줄어들었다. 컴퓨터 센터는 시간당 처리할 수 있는 작업의 수가 대폭 늘었다.
- 그리고 1950년대 후반에는 운영체제가 난립한다.



2. Course Introduction

운영체제 약사

- 운영체제의 표준화가 중요한 문제로 대두된 것이다. 이 표준화 작업의 선두에 IBM이 선다. IBM은 1964년에 야심적으로 System/360이라는 컴퓨터를 시판하기 시작한다. 이후 이 모델은 많은 회사가 따르는 표준으로 군림하는데, 컴퓨터 아키텍처에 한 획을 그은 것으로 평가될 정도이다. IBM은 운영체제에서도 야심작을 모색한다. 360에서 사용할 운영체제 OS/360의 개발을 시작한 것이다. 야심 차게 출범한 개발 팀은 얼마 지나지 않아 어리둥절한 상황에 빠진다. 일정이 걱정스러울 정도로 지체된 것이다. 그들은 중요한 사실을 깨닫는다. 그것은 운영체제라는 프로그램은 예상 밖으로 복잡하고 거대하다는 사실이었다. IBM은 당황한다.



2. Course Introduction

운영체제 약사

- 소프트웨어 분석가, 프로그래머, 행정 보조원 등을 포함하여 1000명 가량이 OS/360에 매달렸다. 하지만 일은 더욱 더디게 진행되었다. 프로젝트에 참여하였던 브룩스는 통찰력을 가장 먼저 그리고 가장 깊이 얻은 사람이었다. 그는 1975년에 <신화적 맨먼스The Mythical Man-Month>라는 유명한 책을 출판한다.
- 그는 '아홉 달이 차야 아기가 나온다. 여인을 더 투입해도 기간은 줄어들지 않는다'라는 말로 지연된 프로젝트에 프로그래머를 더 투입하는 짓은 어리석다고 주장하였다. 사람을 더 투입하면, 일을 더 세분화해야 하고, 의사 소통에 따르는 부차적인 일이 부풀고, 개발된 프로그램을 하나로 합치는 일이 복잡해지기 때문이다.



2. Course Introduction

운영체제 약사

- 이후, 1950년대 들어, 영국은 페란티라는 회사와 맨체스터 대학이 손잡고 아트라스라는 컴퓨터를 개발한다. 이 컴퓨터가 운영체제에서 획기적인 공헌을 했는데, 그것은 '가상 메모리' virtual memor 가 그것이다.
- 1957년에는 시분할을 처음 고안한 사람으로 알려져 있는 비머가 <자동제어>라는 잡지에 글을 썼는데, 실물 제작은 MIT의 전기공학과 교수인 매카시, 파노, 코바토는 1961년에 세계 최초의 시분할 운영체제 CTSS(Compatible Time-Sharing System)를 시연하면서 가능해졌다. 이후, MIT는 CTSS의 성공에 힘입어, AT&T의 벨연구소와 제너럴일렉트릭과 손잡고 1964년에 맥MAC 프로젝트를 시작한다. 이 프로젝트의 주요 관심사는 멀티스라는 MULTICS(MULTiplexed Information and Computing Service) 시분할 운영체제를 개발하는 일이었다.



2. Course Introduction

운영체제 약사

- 그러나, 벨연구소의 경영진뿐 아니라 연구원들이 느끼기에, 멀티스는 너무 많은 비용이 들고 너무 지체되고 있었다. 이러한 혼란스런 상황에서도, 톰슨과 리치는 운영체제의 핵심인 파일 시스템의 설계에 몰두한다. 새로운 운영체제에 대한 갈망이 무척 강하였던 톰슨은 멀티스 프로젝트에서 사용하던 GE-645 컴퓨터에서, 나중에 유닉스라는 이름을 얻게 되는 새로운 운영체제를 위한 파일 시스템을 완성한다. 이렇게 1969년에 유닉스가 탄생한다. 1970년이 되자, 연구소의 다른 사람들이 유닉스를 기웃거리기 시작한다.
- 동료인 커니간은 멀티스의 다중을 뜻하는 Multi를 하나를 뜻하는 Uni로 바꾸어 Unics라고 부르자고 제안한다. 나중에 시분할이 완성되어 다중 사용자를 지원하게 되자 유닉스라는 Unix 이름으로 굳어진다. 그리고 유닉스는 부산물로서 또 다른 발명품을 인류에게 선물한다. 그것은 다름 아닌 C 언어이다.



2. Course Introduction

본 강좌의 의의

- 본 과정은 컴퓨터의 내부활동을 조정하고 외부세계와의 통신을 관리하며, 컴퓨터의 전반적인 운영을 제어하는 운영체제에 대한 이론적인 원리와 개념을 이해하고자 하는 인프라 엔지니어를 위한 과정으로서 다음 내용을 학습 목표로 한다.
- 1) 운영체제의 기능과 역할 및 구성 등 기본개념을 알 수 있다.
 - 2) 프로세서의 개념과 종류 스케줄링과의 연관관계 및 변환에 대하여 알 수 있다.
 - 3) 기억장치의 내부구조 및 알고리즘에 대하여 알 수 있다.
 - 4) 디스크공간의 할당 및 디스크 스케줄링을 알 수 있다.
 - 5) 파일관리시스템을 이해한다. 분산처리시스템 및 시스템보안에 대하여 알 수 있다.

동작

이용



3. Course Structure

학습모듈의 내용체계

교육 내용

세부 학습 내용

『1과목』 컴퓨터 시스템 기본

『2과목』 운영 체제 개관

『3과목』 Processes

『4과목』 Threads & Concurrency

『5과목』 CPU Scheduling

『6과목』 Synchronization Tools

『7과목』 Deadlocks

『8과목』 Main Memory

『9과목』 Virtual Memory

『10과목』 Mass-Storage Systems

『11과목』 I/O Systems

『12과목』 File-System

『13과목』 File System Implementation

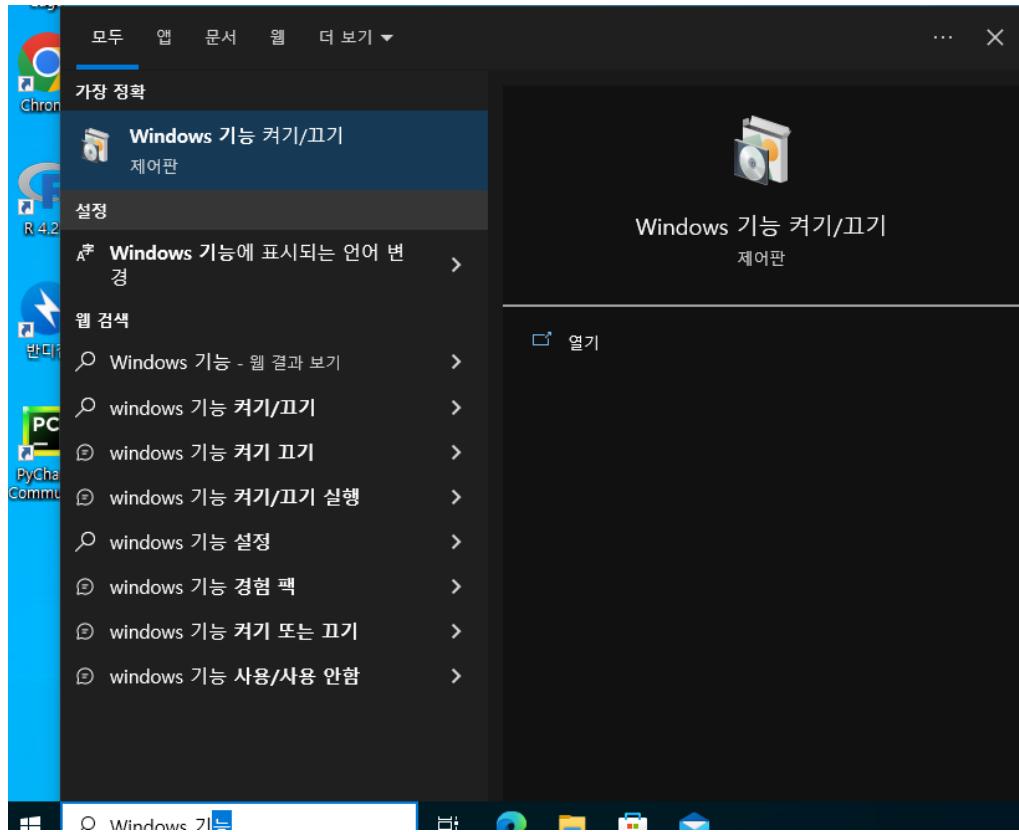
『14과목』 Security



4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

- 원도 시작 버튼 > 검색
 - Windows 기능 > Windows 기능 켜기 /끄기 선택

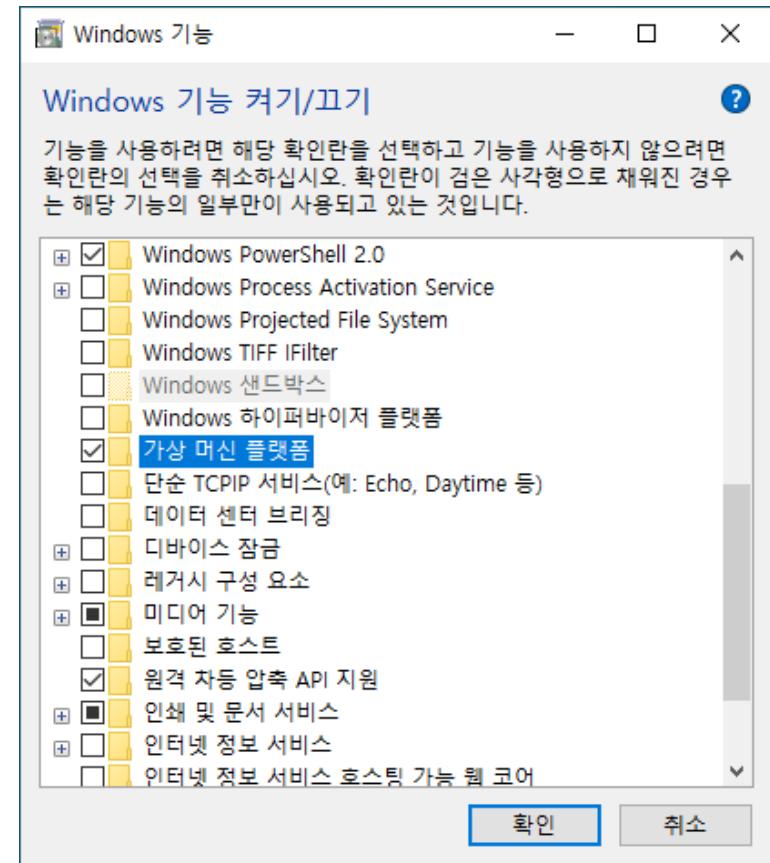
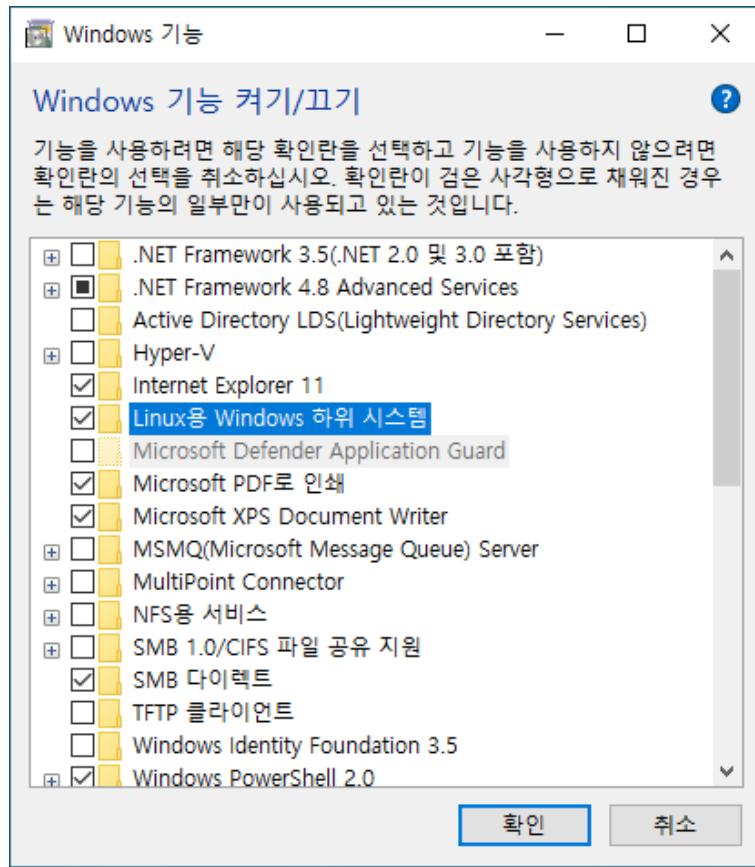




4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

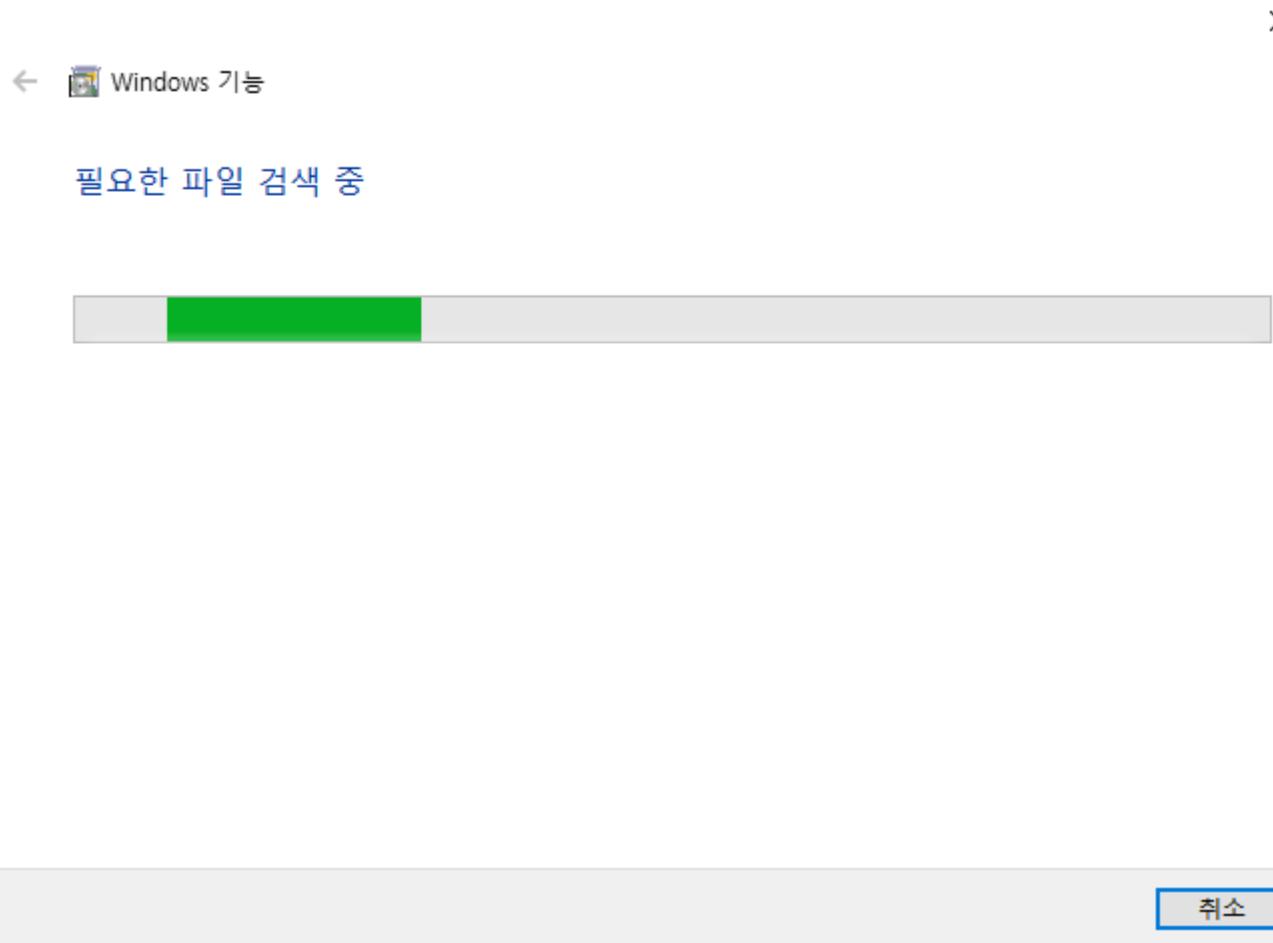
● Linux용 Windows 하위 시스템과 가상 머신 플랫폼 선택



4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

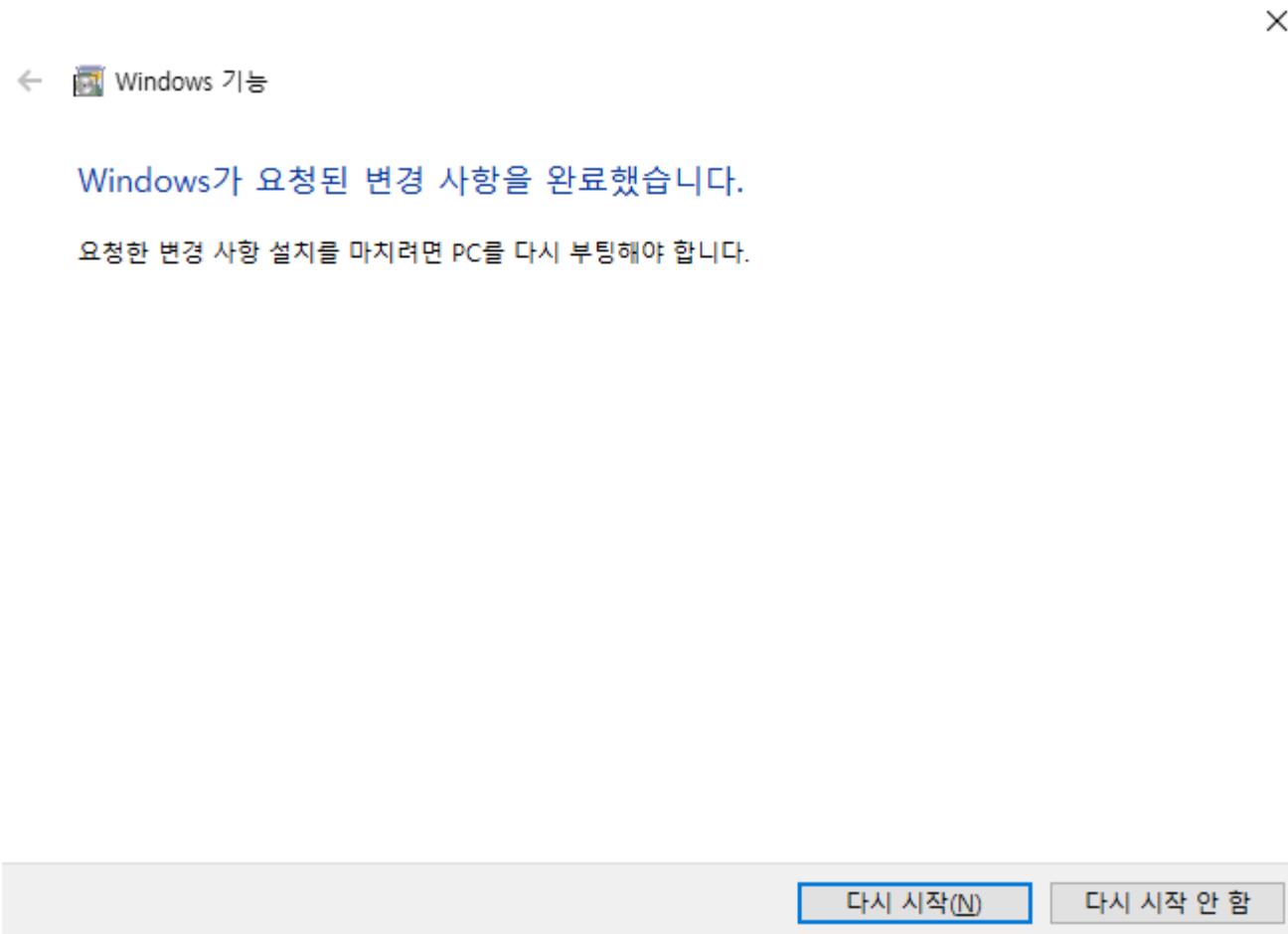
- 필요한 파일 설치



4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

- 원도 다시 시작

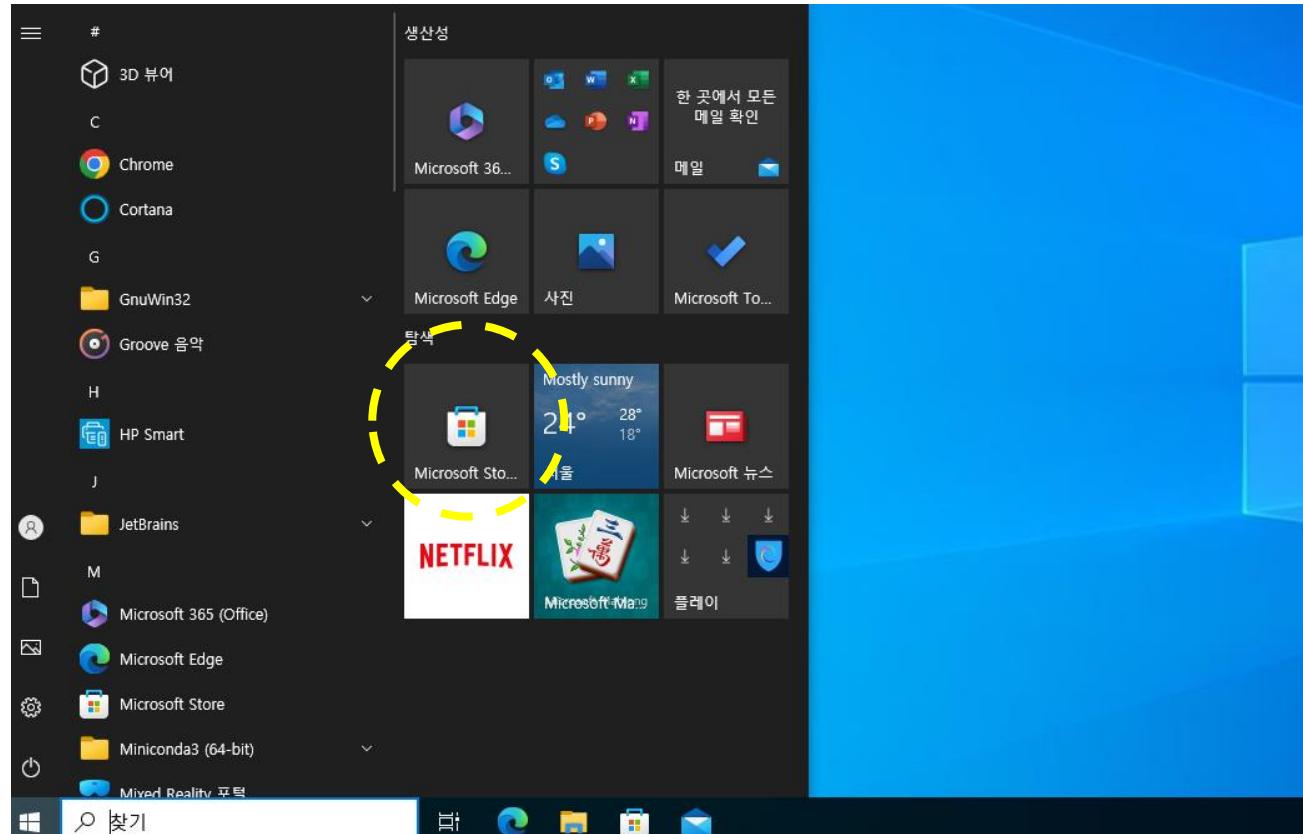




4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

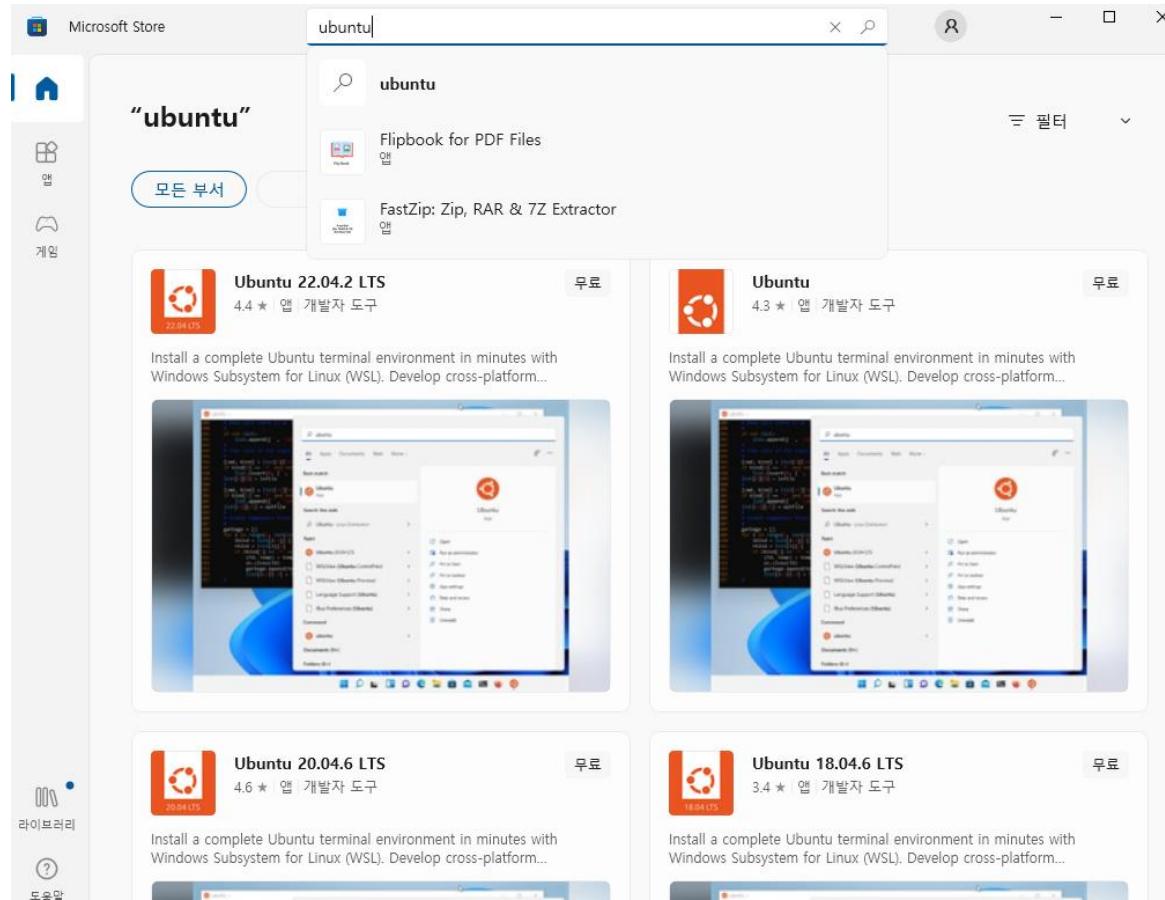
- 시작 > 마이크로소프트 스토어 선택



4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

● Ubuntu 검색

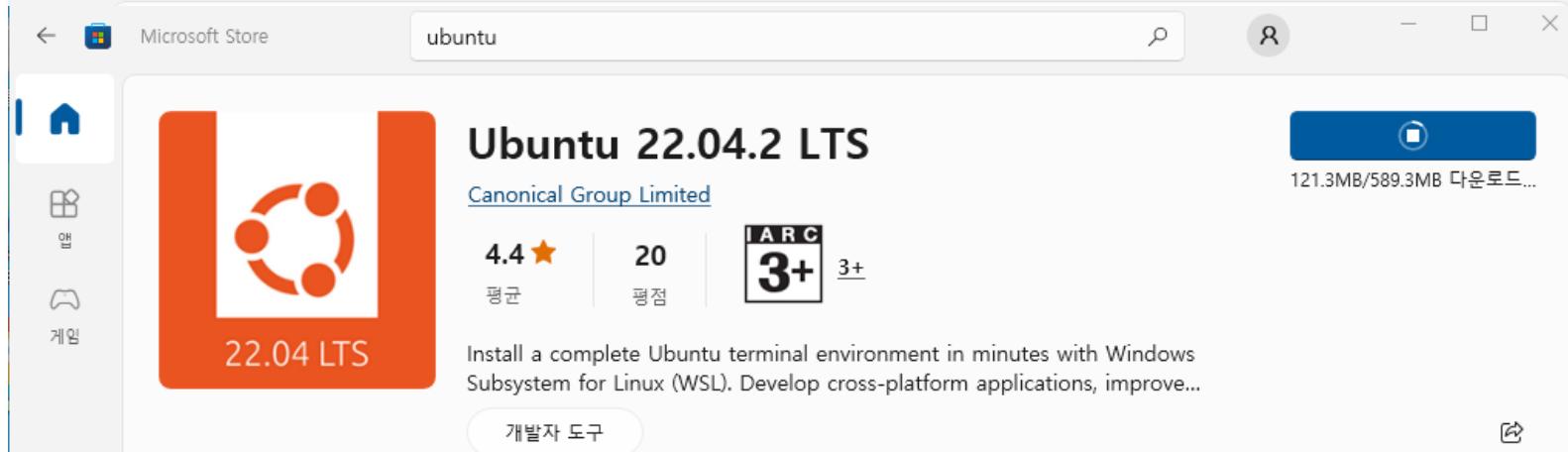
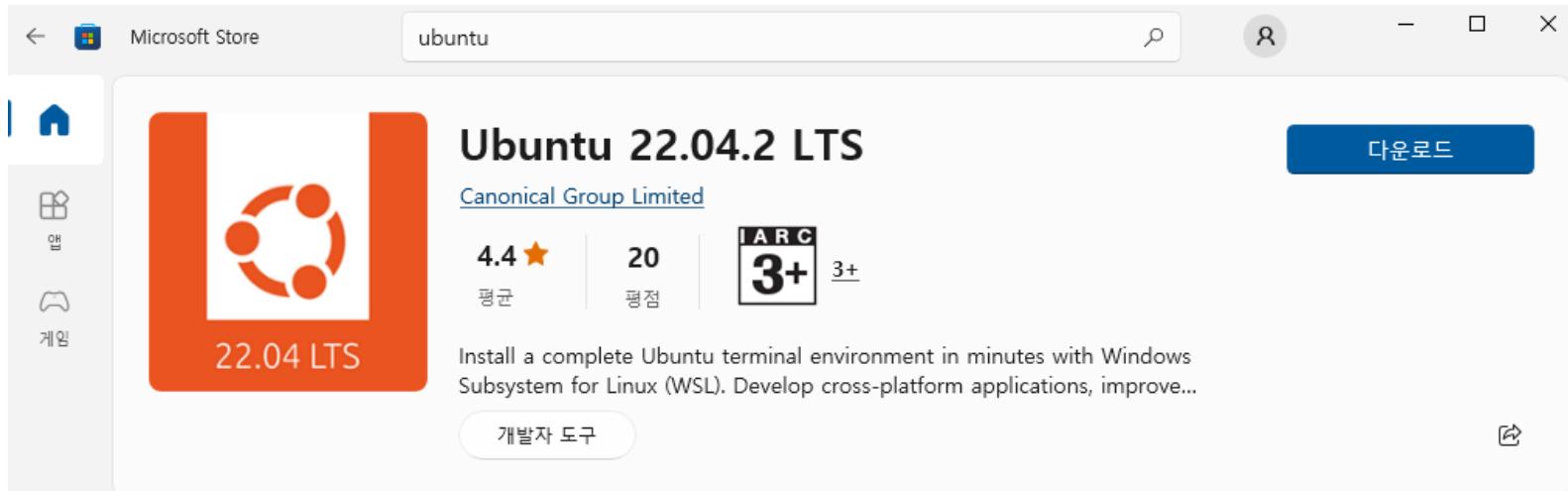




4. Pre-Requisites

1. 원10에 우분투 설치-WSL 설치

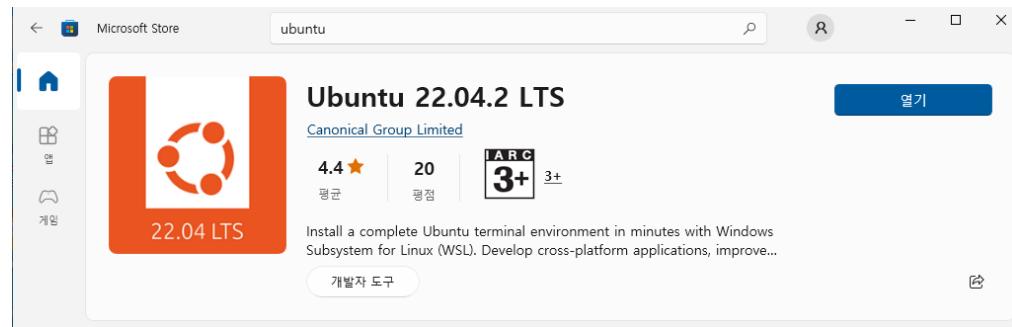
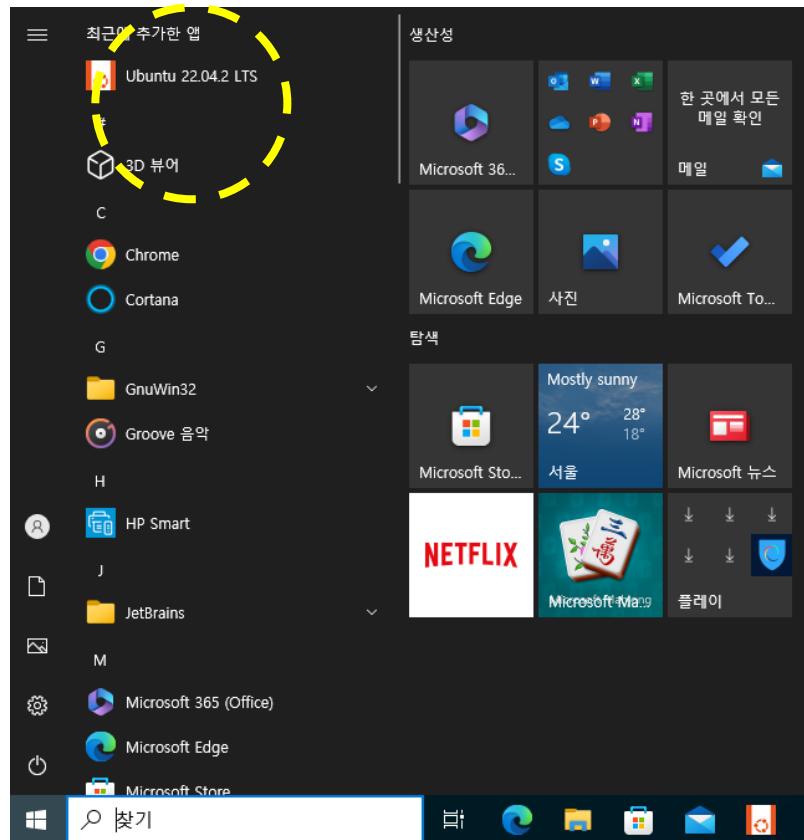
- Ubuntu 다운로드와 열기



4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

● Ubuntu 다운로드와 열기



4. Pre-Requisites

1. 윈10에 우분투 설치-WSL 설치

● Ubuntu 계정과 암호 입력

```
Ubuntu 22.04.2 LTS
Installing, this may take a few minutes...
```

```
k8s@DESKTOP-R0EQ2U6: ~
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows u
sername.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: k8s
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
이제 Linux용 Windows 하위 시스템을 Microsoft Store에서 사용할 수 있습니다.
'wsl.exe --update'를 실행하거나 https://aka.ms/wslstorepage를 방문하여 업그레이드할 수 있습니
다.
```



4. Pre-Requisites

sudo apt-get update

```
k8s@DESKTOP-RoEQ2U6:~$ sudo apt-get update
```



4. Pre-Requisites

gcc

```
k8s@DESKTOP-RoEQ2U6:~$ gcc --version
```

Command 'gcc' not found, but can be installed with:

```
sudo apt install gcc
```

```
k8s@DESKTOP-RoEQ2U6:~$ sudo apt install gcc
```

[sudo] password for k8s:

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following additional packages will be installed:

...
...

Do you want to continue? [Y/n]

Setting up libdce_dextools (2/25)

Setting up libfc-devtools (2.35-
Ubuntu 2.1) ... #######

Processing triggers for man-db (2.10.3-
Ubuntu3.1) ...#

Processing triggers for main-db (2.10.2-1) #####

Processing triggers for libc-bin (2.35-0ubuntu3.1) ...

k8s@DESKTOP-R0EQ2U6

gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0

gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
Copyright (C) 2021 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

This is free software; see the source code for warranty: not even for MERCHANT.

k8s@DESKTOP-8OEQ3U6:~\$



4. Pre-Requisites

g++

k8s@DESKTOP-RoEQ2U6:~\$ g++ --version

Command 'g++' not found, but can be installed with:

sudo apt install g++

k8s@DESKTOP-RoEQ2U6:~\$ sudo apt-get install g++

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following additional packages will be installed:

....

Do you want to continue? [Y/n] Y

...

Setting up g++ (4:11.2.0-1ubuntu1) ...

update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode

Processing triggers for man-db (2.10.2-1) ...

k8s@DESKTOP-RoEQ2U6:~\$ g++ --version

g++ (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0

Copyright (C) 2021 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

k8s@DESKTOP-RoEQ2U6:~\$



4. Pre-Requisites

Java

k8s@DESKTOP-RoEQ2U6:~\$ java --version

Command 'java' not found, but can be installed with:

```
sudo apt install openjdk-11-jre-headless # version 11.0.18+10-oubuntu1~22.04, or  
sudo apt install default-jre          # version 2:1.11-72build2  
sudo apt install openjdk-17-jre-headless # version 17.0.6+10-oubuntu1~22.04  
sudo apt install openjdk-18-jre-headless # version 18.0.2+9-2~22.04  
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-oubuntu3~22.04  
sudo apt install openjdk-8-jre-headless # version 8u362-ga-oubuntu1~22.04
```

k8s@DESKTOP-RoEQ2U6:~\$ sudo apt install openjdk-17-jdk openjdk-17-jre -y

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following additional packages will be installed:

...

done.

Setting up openjdk-17-jre-headless:amd64 (17.0.6+10-oubuntu1~22.04) ...

update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/java to provide /usr/bin/java (java) in auto mode

update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jpackage to provide /usr/bin/jpackage (jpackage) in auto mode

update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode

update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode

update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/lib/jexec to provide /usr/bin/jexec (jexec) in auto mode

k8s@DESKTOP-RoEQ2U6:~\$ java --version

openjdk 17.0.6 2023-01-17

OpenJDK Runtime Environment (build 17.0.6+10-Ubuntu-oubuntu122.04)

OpenJDK 64-Bit Server VM (build 17.0.6+10-Ubuntu-oubuntu122.04, mixed mode, sharing)

k8s@DESKTOP-RoEQ2U6:~\$ javac --version

javac 17.0.6

k8s@DESKTOP-RoEQ2U6:~/java_workspaces\$



4. Pre-Requisites

Java

JDK Version	Type	Release Date	Highlights	Recommendation
8	LTS	03/2014	Lambdas	Last LTS version under previous release model. Free updates by Oracle <u>ended</u> , but still maintained by others. Upgrade to a 11 or 17 within the next months!
9	Feature	09/2017	Modules	New release model was introduced. EOL. Upgrade to 11 or 17 now!
10	Feature	03/2018	var	EOL. Upgrade to 11 or 17 now!
11	LTS	09/2018	New HTTP Client	Widely used LTS version. Plan upgrade to version 17 within the next months.
12	Feature	03/2019		EOL. Upgrade to 17 now!
13	Feature	09/2019		EOL. Upgrade to 17 now!
14	Feature	03/2020	Switch expressions	EOL. Upgrade to 17 now!
15	Feature	09/2020	Text blocks	EOL. Upgrade to 17 now!
16	Feature	03/2021	Records	EOL. Upgrade to 17 now!
17	LTS	09/2021	Sealed Classes	Current LTS version.

『1과목』

2-4교시 :

컴퓨터 시스템 기본





학습목표

- 이 워크샵에서는 컴퓨터 시스템의 일반적인 구성과 인터럽트의 역할 설명할 수 있다.
- 최신 다중 프로세서 컴퓨터 시스템의 구성 요소 설명할 수 있다.
- 사용자 모드에서 커널 모드로의 전환 설명할 수 있다.
- 다양한 컴퓨팅 환경에서 운영 체제를 사용하는 방법 논의할 수 있다.
- 무료 및 오픈 소스 운영 체제의 예 제공할 수 있다.



눈높이 체크

- 운영 체제가 수행하는 작업을 알고 계신가요?
- 컴퓨터 시스템 구성을 알고 계신가요?
- 컴퓨터 시스템 아키텍처를 알고 계신가요?
- 운영 체제 운영을 알고 계신가요?
- 자원 관리를 알고 계신가요?
- 보안 및 보호를 알고 계신가요?
- 가상화를 알고 계신가요?
- 분산 시스템을 알고 계신가요?
- 커널 데이터 구조를 알고 계신가요?
- 컴퓨팅 환경을 알고 계신가요?
- 무료/자유 및 오픈 소스 운영 체제를 알고 계신가요?



1. 운영 체제란?

운영 체제가 수행하는 **작업**

- **관점에 따라 다름**

- 사용자는 편리함, 사용 용이성 및 우수한 성능을 필요로 함.
 - 리소스 활용에 대해 신경 쓰지 않음
- 그러나 메인프레임이나 미니컴퓨터와 같은 공유 컴퓨터는 모든 사용자를 만족시켜야 함.
 - 운영 체제는 HW를 효율적으로 사용하고 사용자 프로그램의 실행을 관리하는 자원 할당자 및 제어 프로그램.
- 워크스테이션과 같은 전용 시스템 사용자는 전용 리소스를 가지고 있지만 서버에서 공유 리소스를 자주 사용.
- 스마트 폰 및 태이블과 같은 모바일 장치는 리소스가 부족하고 유용성 및 배터리 수명에 최적화되어 있다.
 - 터치스크린, 음성인식 등 모바일 사용자 인터페이스
- 일부 컴퓨터에는 장치 및 자동차에 내장된 컴퓨터와 같이 사용자 인터페이스가 거의 없거나 전혀 없다.
 - 주로 사용자 개입 없이 실행



1. 운영 체제란?

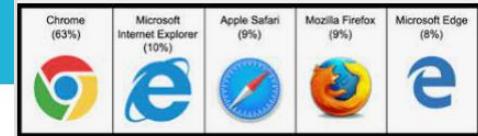
운영 체제 **목표**

- 컴퓨터 사용자와 컴퓨터 하드웨어 사이에서 중개자 역할을 하는 프로그램
- 운영 체제 목표:
 - 1) 사용자 프로그램을 실행하고 사용자 문제를 더 쉽게 해결
 - 2) 컴퓨터 **시스템**을 사용하기 편리하게 만들고자 함.
 - 3) 컴퓨터 하드웨어를 효율적으로 사용

시스템이란?



1. 운영 체제란?



운영 체제 정의

Operating System

- 보편적으로 인정되는 정의 없음
- "운영 체제를 주문할 때 공급업체에서 제공하는 모든 것"
- "컴퓨터에서 항상 실행되는 하나의 프로그램"은 운영 체제의 일부인 커널.
- 다른 모든 것은
- 시스템 프로그램(운영 체제와 함께 제공되지만 커널의 일부는 아님) 또는
- 응용 프로그램, 운영 체제와 연결되지 않은 모든 프로그램
- 범용 및 모바일 컴퓨팅을 위한 오늘날의 OS에는 데이터베이스, 멀티미디어, 그래픽과 같은 애플리케이션 개발자에게 추가 서비스를 제공하는 소프트웨어 프레임워크 세트인 미들웨어도 포함.

컴퓨터란?

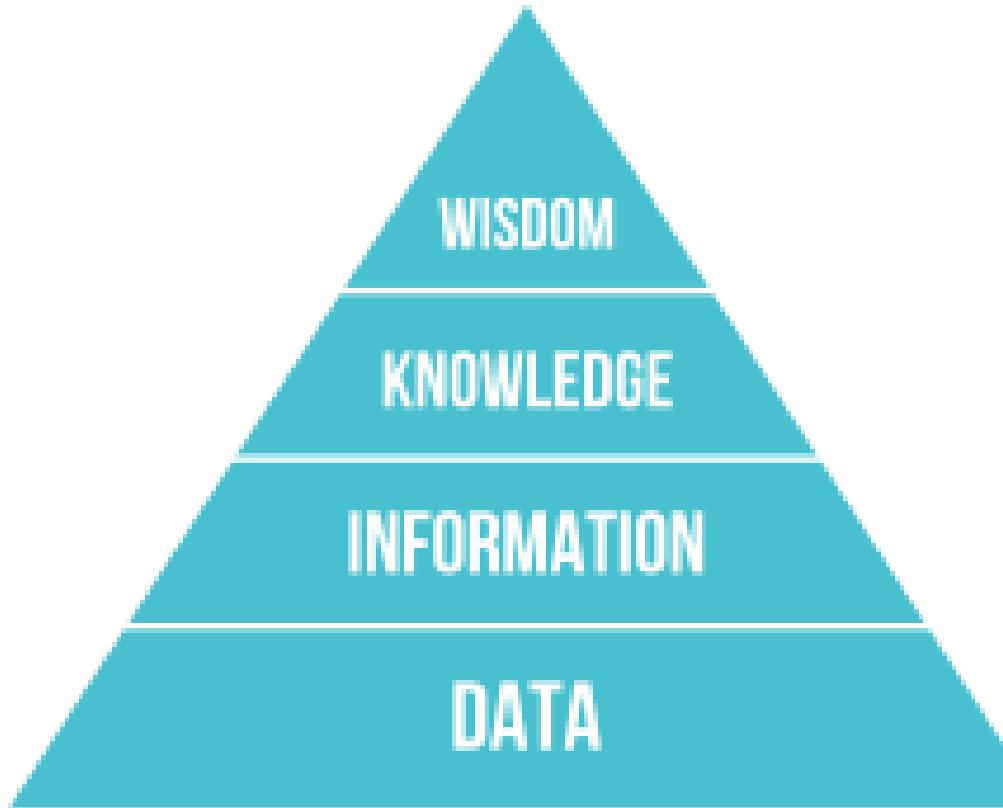
- A computer is a machine that processes the information.



2. 컴퓨터 시스템 구조

정보란?

- The DIKW pyramid



1948년 클로드 셜너은
<통신의 수학적 이론 Mathematical Theory of Communication>라는 논문을 발표하여 정보이론의 토대를 마련합니다.

양수 만들기

$$H = -k \sum_{i=1}^n p(i) \cdot \log_2 p(i)$$

해당 정보 나올 확률

정보엔트로피
(정보량)



2. 컴퓨터 시스템 구조

| 정보란?

- 정보의 최소 단위 : bit(binary digit)
- 정보의 처리: 정보의 상태 변환(0에서 1로, 1에서 0으로)
- 부울 대수(Boolean Algebra): NOT, AND, OR
- 논리 게이트 : NOT, AND, OR, XOR, NAND, NOR
- 논리 회로 : IC, LSI, VLSI, ULSI, Soc, ...
- 무어의 법칙(Moore's Law)과 황의 법칙(Huang's Law)
- 정보의 저장과 전송: 플립-플롭, 데이터 버스, RF



2. 컴퓨터 시스템 구조



| 정보 처리

- 덧셈 : 반가산기, 전가산기
- 뺄셈 : 2의 보수 표현
- 곱셈과 나눗셈 : 덧셈과 뺄셈의 반복
- 실수 연산 : 부동 소수점 표현
- 함수? GOTO
- 삼각함수, 미분, 적분, 이미지 처리, 동영상 처리...



2. 컴퓨터 시스템 구조

컴퓨터 시스템 구조

- 컴퓨터 시스템은 네 가지 구성 요소로 나눌 수 있다.
- 하드웨어 – 기본 컴퓨팅 리소스 제공
 - CPU, 메모리, I/O 장치
- 운영 체제
 - 다양한 애플리케이션과 사용자 간의 하드웨어 사용을 제어하고 조정한다.
- 응용 프로그램 – 사용자의 컴퓨팅 문제를 해결하기 위해 시스템 자원이 사용되는 방식을 정의합니다.
 - 워드 프로세서, 컴파일러, 웹 브라우저, 데이터베이스 시스템, 비디오 게임
- 사용자
 - 사람, 기계, 기타 컴퓨터

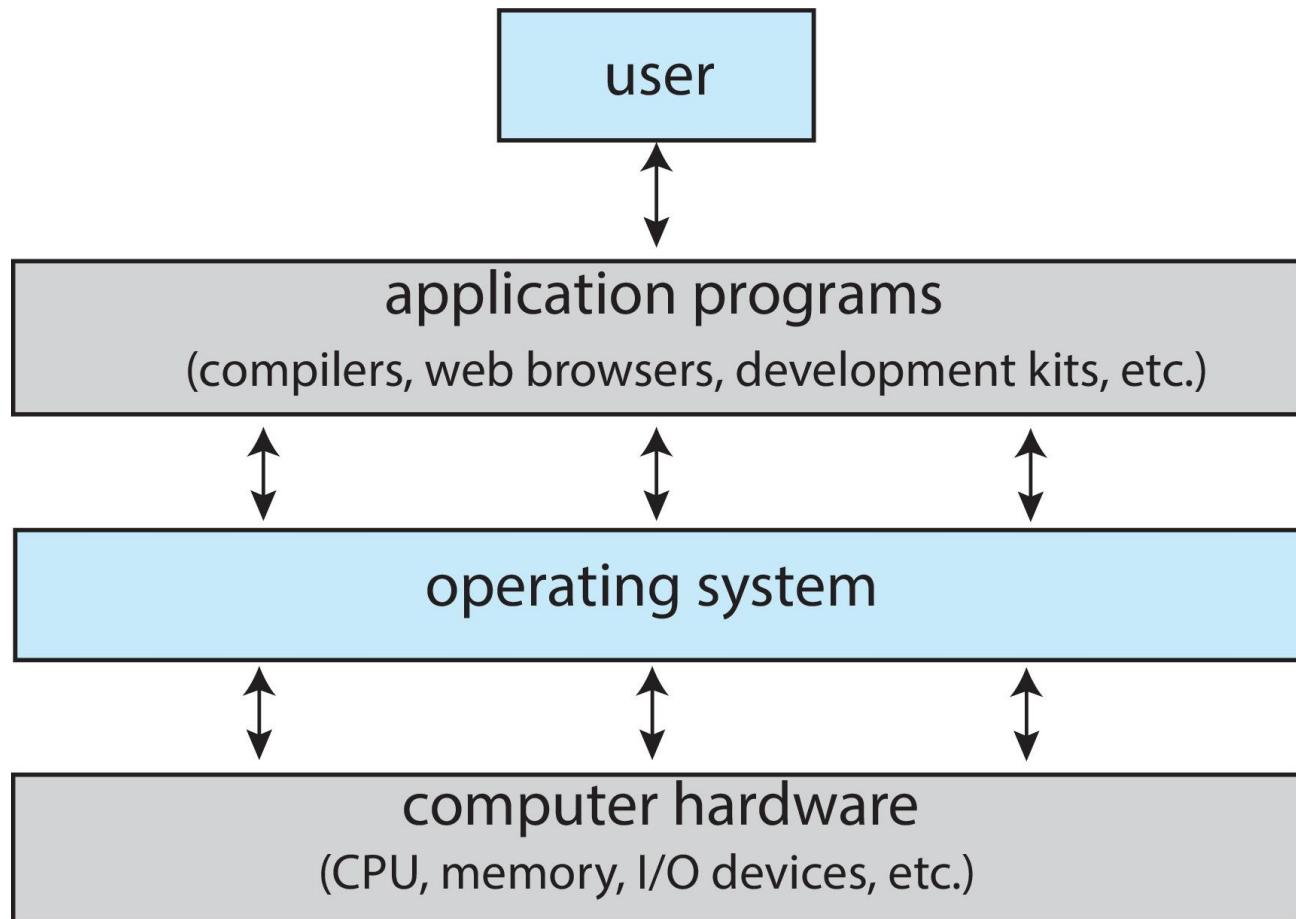


2. 컴퓨터 시스템 구조

컴퓨터 구성 요소의 추상

❖ System Call

- 응용 프로그램은 운영 체제의 커널(Kernel)이 제공하는 함수를 어셈블리어나 C/C++과 같은 저수준 프로그래밍 언어를 사용하여 호출하여 응용 프로그램이 운영 체제의 기능을 활용하여 파일 및 입출력 관리, 네트워크 통신, 프로세스 관리, 메모리 관리 등 다양한 작업을 수행하는데, 인터럽트를 통해 운영 체제로 제어를 넘기는 방식으로 동작
- 인터럽트는 다양한 종류가 있으며, 시스템 콜을 호출하는 경우에는 주로 소프트웨어 인터럽트(Software Interrupt) 또는 트랩(Trap) 인터럽트가 사용

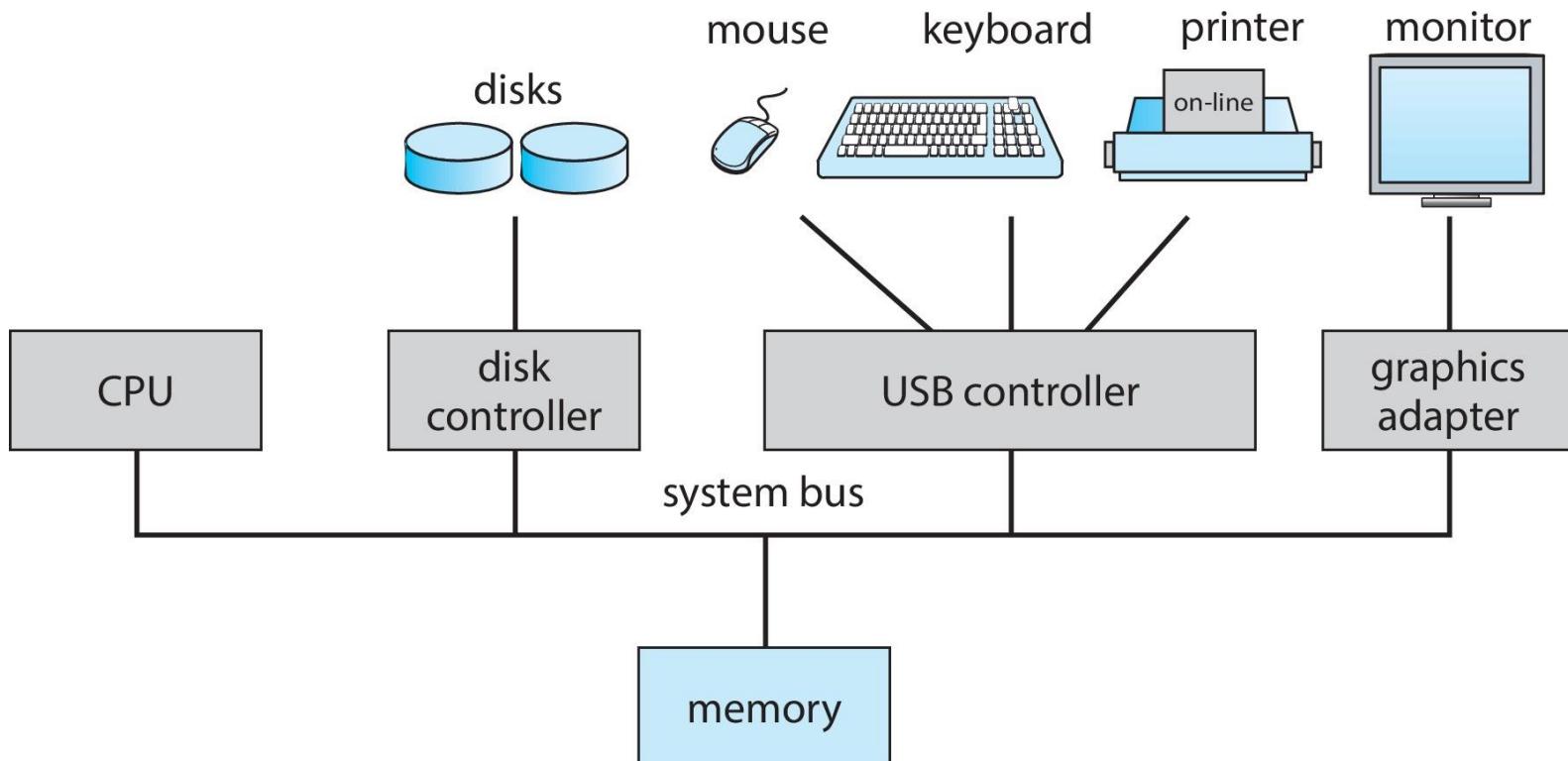




3. 컴퓨터 시스템 구성

컴퓨터 시스템 구성

- 컴퓨터 시스템 작동
- 하나 이상의 CPU, 장치 컨트롤러는 공유 메모리에 대한 액세스를 제공하는 공통 버스를 통해 연결.
- 메모리 주기를 위해 경쟁하는 CPU 및 장치의 동시 실행





4. 컴퓨터 시스템 작동

컴퓨터 시스템 작동

- I/O 장치와 CPU는 동시에 실행할 수 있다.
- 각 장치 컨트롤러는 특정 장치 유형을 담당.
- 각 장치 컨트롤러에는 로컬 버퍼가 있다.
- 각 장치 컨트롤러 유형에는 이를 관리하는 운영 체제 장치 드라이버가 있다.
- CPU는 메인 메모리에서 로컬 버퍼로/에서 데이터를 이동.
- I/O는 장치에서 컨트롤러의 로컬 버퍼로
- 장치 컨트롤러는 CPU에 인터럽트를 발생시켜 작업이 완료되었음을 알린다.



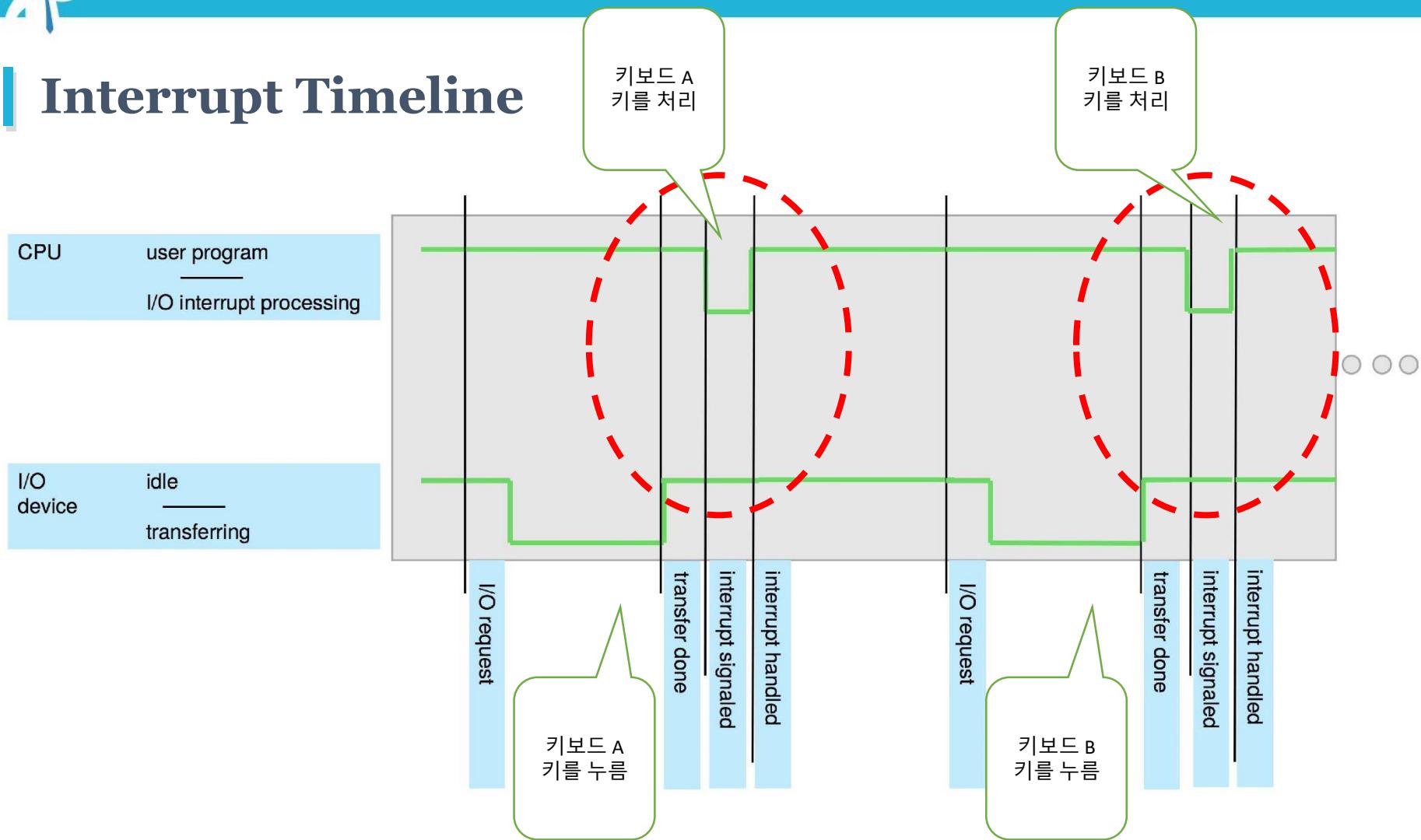
4. 컴퓨터 시스템 작동

| 인터럽트의 일반적인 기능

- 인터럽트는 일반적으로 모든 서비스 루틴의 주소를 포함하는 인터럽트 벡터를 통해 인터럽트 서비스 루틴으로 제어를 전송.
- 인터럽트 아키텍처는 인터럽트된 명령의 주소를 저장.
- 트랩 또는 예외는 오류 또는 사용자 요청으로 인해 발생하는 소프트웨어 생성 인터럽트.
- 운영 체제는 인터럽트 기반.

4. 컴퓨터 시스템 작동

Interrupt Timeline



Interrupt CPU와 I/O 사이 통신 방식



4. 컴퓨터 시스템 작동

Interrupt Handling

- 운영 체제는 레지스터와 프로그램 카운터를 저장하여 CPU의 상태를 보존.
- 어떤 유형의 인터럽트가 발생했는지 결정.
- 별도의 코드 세그먼트는 각 유형의 인터럽트에 대해 수행해야 하는 작업을 결정.

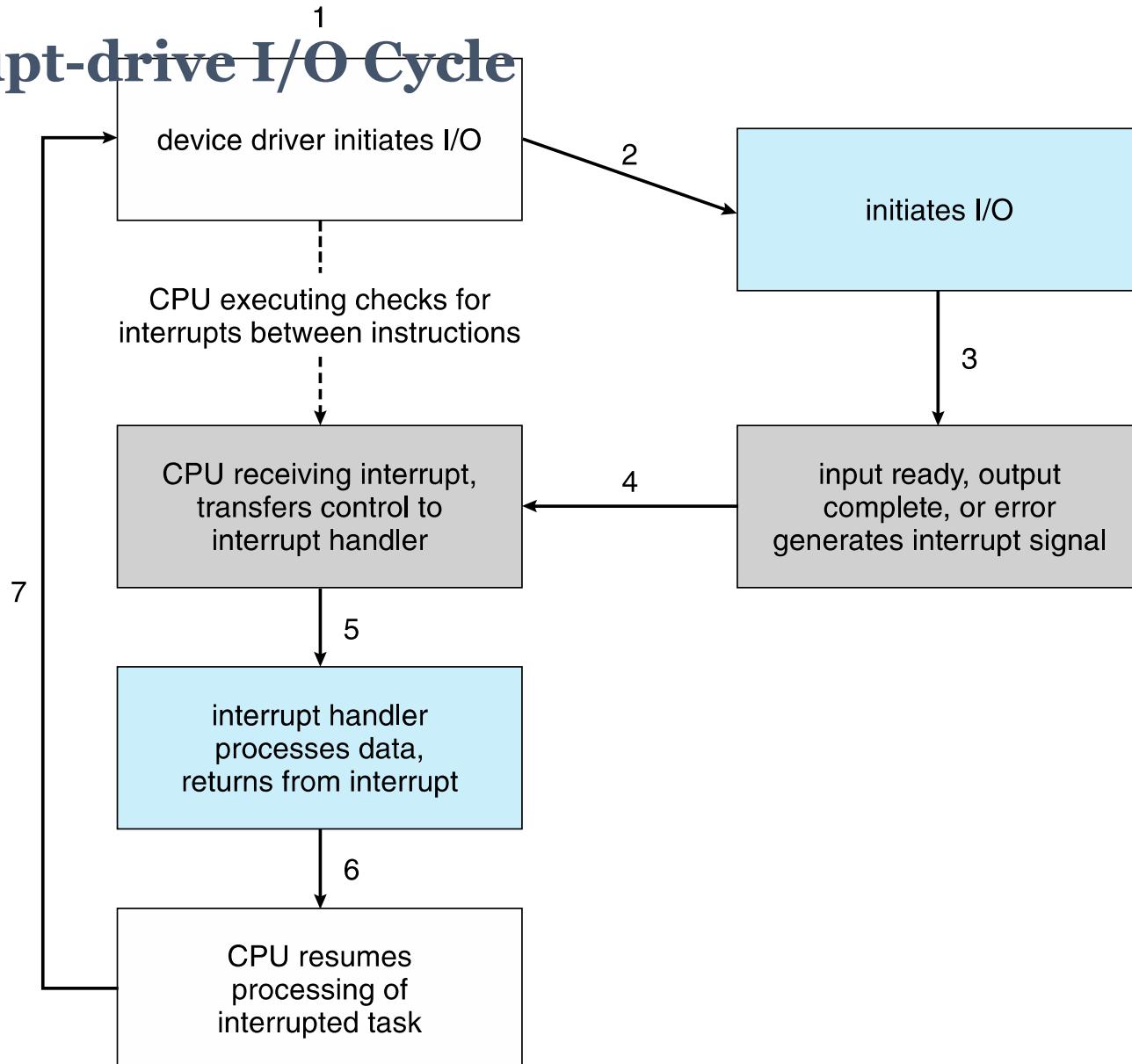


4. 컴퓨터 시스템 작동

CPU

I/O controller

Interrupt-drive I/O Cycle





4. 컴퓨터 시스템 작동

I/O Structure

- I/O를 처리하는 두 가지 방법

1. I/O가 시작된 후 제어는 I/O 완료 시에만 사용자 프로그램으로 돌아간다.

- 대기 명령은 다음 인터럽트까지 CPU를 유휴 상태로 만든다.
- 대기 루프(메모리 액세스 경합)
- 한 번에 최대 하나의 I/O 요청이 미결 상태이며 동시 I/O 처리가 없다.

2. I/O가 시작된 후 제어는 I/O 완료를 기다리지 않고 사용자 프로그램으로 돌아간다.

- 시스템 호출 – 사용자가 I/O 완료를 기다릴 수 있도록 OS에 요청
- 장치 상태 테이블에는 유형, 주소 및 상태를 나타내는 각 I/O 장치에 대한 항목이 포함되어 있다.
- 장치 상태를 결정하고 인터럽트를 포함하도록 테이블 항목을 수정하기 위해 I/O 장치 테이블에 대한 OS 인덱스

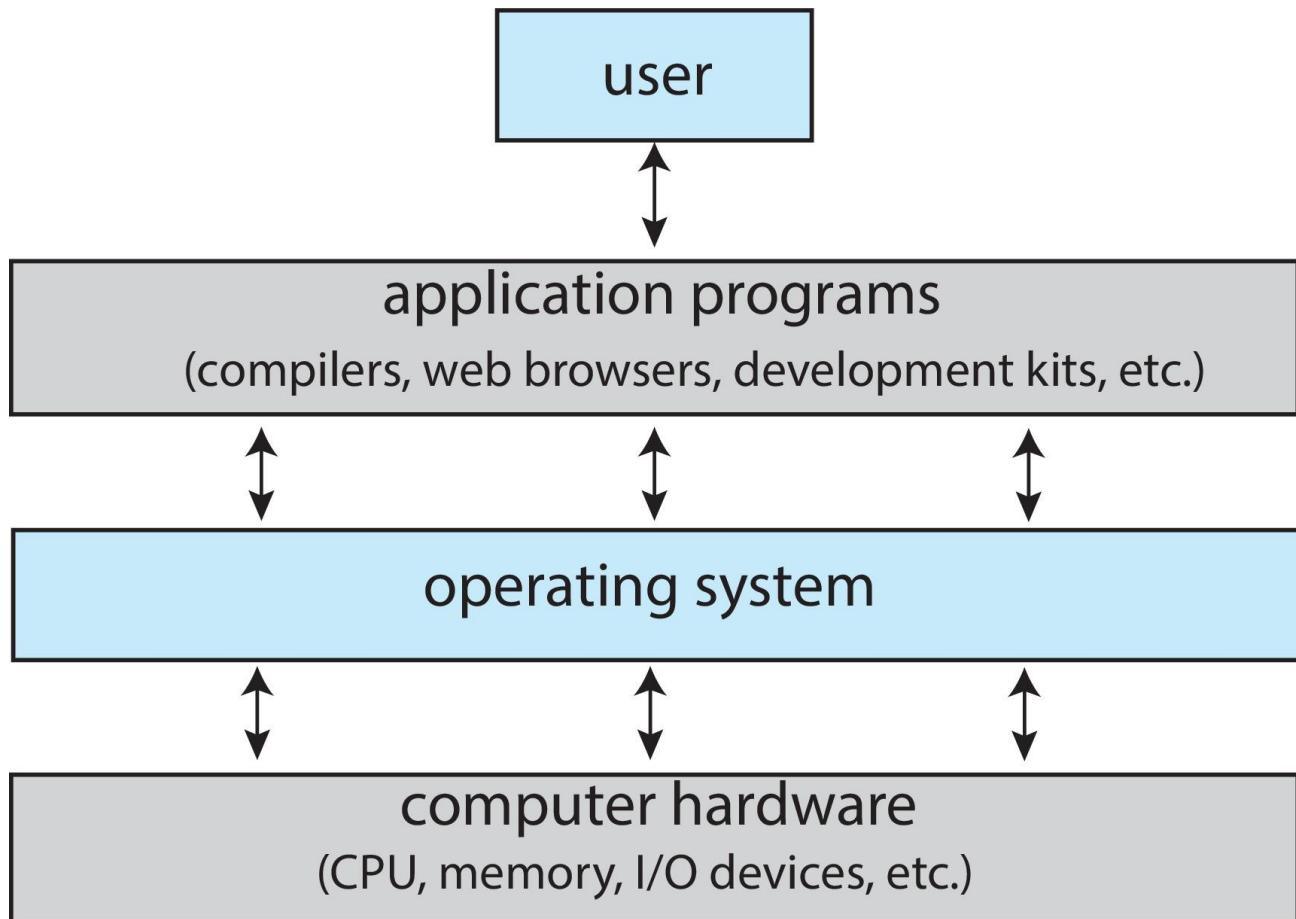


4. 컴퓨터 시스템 작동

컴퓨터 구성 요소의 추상에서 System Call

❖ System Call

- 응용 프로그램은 운영 체제의 커널(Kernel)이 제공하는 함수를 어셈블리어나 C/C++과 같은 저수준 프로그래밍 언어를 사용하여 호출하여 응용 프로그램이 운영 체제의 기능을 활용하여 파일 및 출력 관리, 네트워크 통신, 프로세스 관리, 메모리 관리 등 다양한 작업을 수행하는데, 인터럽트를 통해 운영 체제로 제어를 넘기는 방식으로 동작
- 인터럽트는 다양한 종류가 있으며, 시스템 콜을 호출하는 경우에는 주로 소프트웨어 인터럽트(Software Interrupt) 또는 트랩(Trap) 인터럽트가 사용





4. 컴퓨터 시스템 작동

Computer Startup

- 부트스트랩 프로그램은 전원을 켜거나 재 부팅할 때 로드.
 1. 일반적으로 펌웨어로 알려진 ROM 또는 EPROM에 저장됨
 2. 시스템의 모든 측면을 초기화.
 3. 운영 체제 커널을 로드하고 실행을 시작.



5. Resource Management

Storage Structure

- 주 메모리 – CPU가 직접 액세스할 수 있는 대용량 저장 매체만
 - 임의 액세스
 - 일반적으로 휘발성
 - 일반적으로 DRAM(Dynamic Random-access Memory) 형태의 임의 액세스 메모리
- 보조 스토리지 – 큰 비휘발성 스토리지 용량을 제공하는 메인 메모리의 확장



5. Resource Management

Storage Structure

- **하드 디스크 드라이브(HDD)** – 자기 기록 재료로 덮인 단단한 금속 또는 유리 플래터
 - 디스크 표면은 섹터로 세분되는 트랙으로 논리적으로 나뉜다.
 - 디스크 컨트롤러는 장치와 컴퓨터 간의 논리적 상호 작용을 결정.
- **비휘발성 메모리(NVM)** 장치 – 하드 디스크보다 빠르고 비휘발성
 - 다양한 기술
 - 용량 및 성능 증가, 가격 인하로 대중화



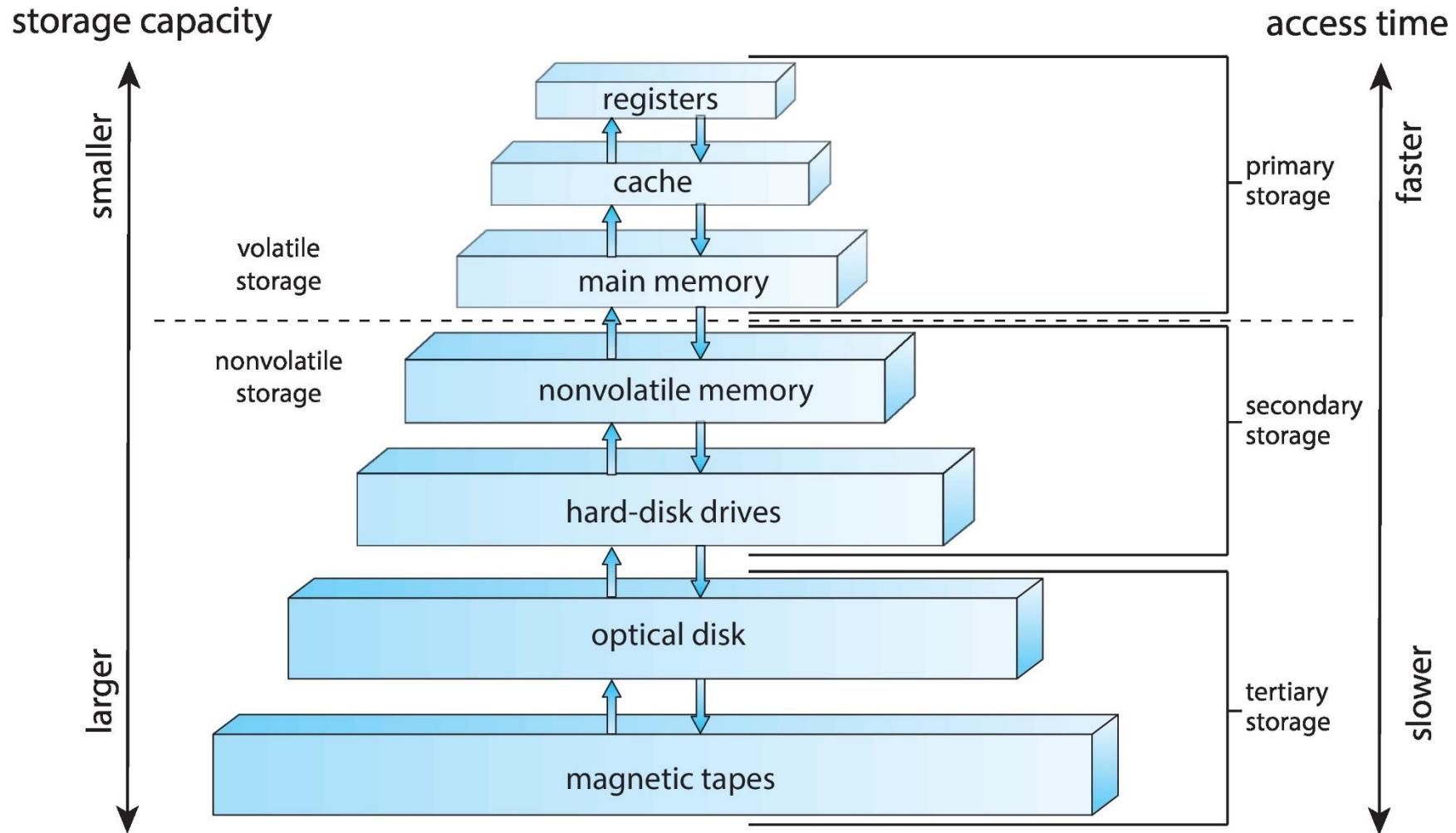
5. Resource Management

Storage Structure

- 계층 구조로 구성된 스토리지 시스템
 1. 속도 Speed
 2. 비용 Cost
 3. 휘발성 Volatility
- 캐싱 Caching – 정보를 더 빠른 스토리지 시스템으로 복사. 주 메모리는 보조 저장소의 캐시로 볼 수 있다.
- I/O 관리를 위한 각 디바이스 컨트롤러용 디바이스 드라이버
 - 컨트롤러와 커널 간의 균일한 인터페이스 제공

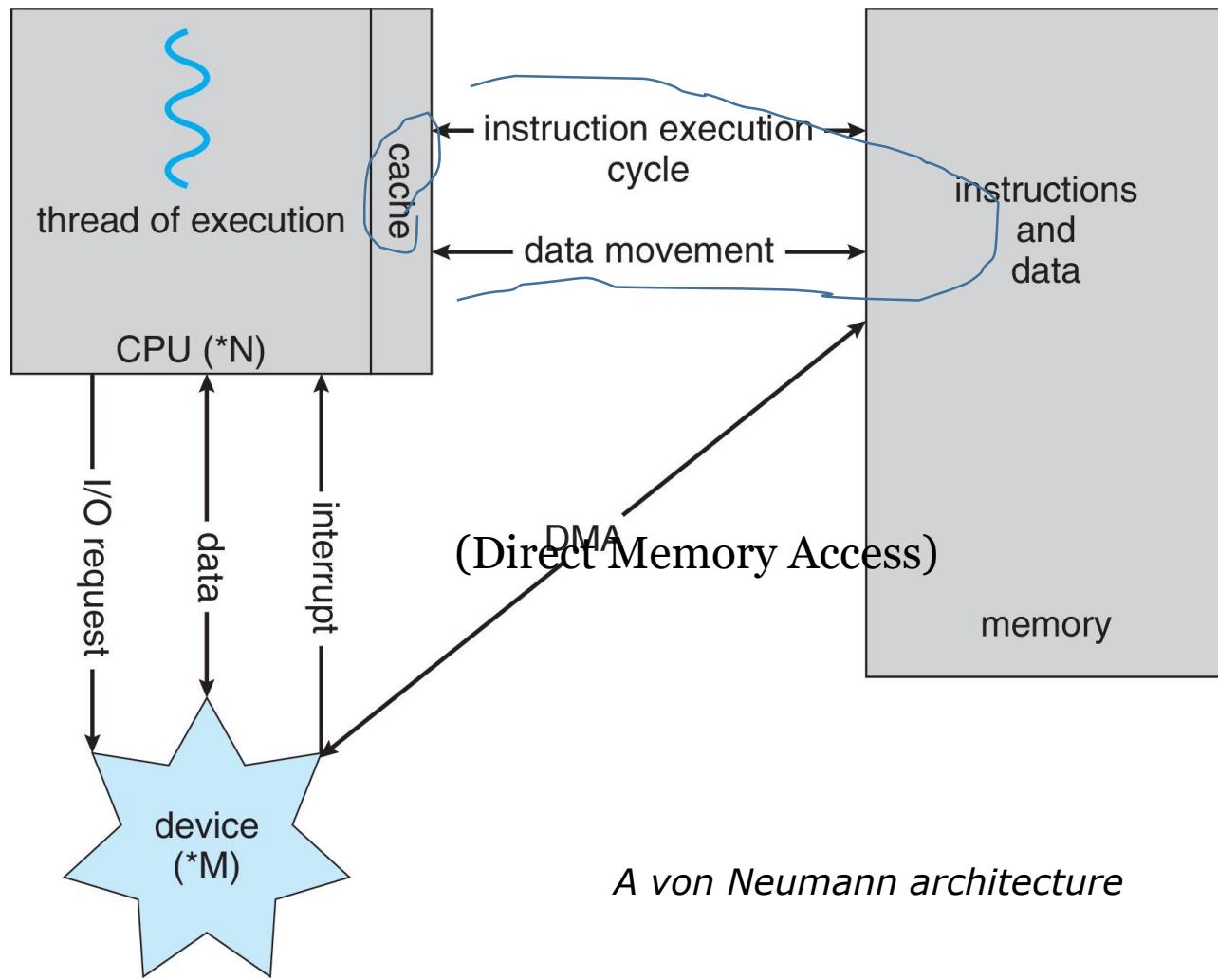
5. Resource Management

Storage-Device Hierarchy



5. Resource Management

최신 컴퓨터의 작동 방식





5. Resource Management

직접 메모리 액세스 구조

- 메모리 속도에 가까운 속도로 정보를 전송할 수 있는 고속 I/O 장치에 사용
- 장치 컨트롤러는 CPU 개입 없이 버퍼 저장소에서 주 메모리로 직접 데이터 블록을 전송.
- 바이트당 하나의 인터럽트가 아니라 블록당 하나의 인터럽트만 생성.



5. Resource Management

Operating-System Operations

- 부트스트랩 프로그램 – 시스템을 초기화하고 커널을 로드하는 간단한 코드
- 커널 로드
- 시스템 데몬 시작(커널 외부에서 제공되는 서비스)
- 커널 인터럽트 기반(하드웨어 및 소프트웨어)
- 장치 중 하나에 의한 하드웨어 인터럽트
- 소프트웨어 인터럽트(예외 또는 트랩):
 - ① 소프트웨어 오류(예: 0으로 나누기)
 - ② 운영체제 서비스 요청 - 시스템 콜
 - ③ 다른 프로세스 문제에는 무한 루프, 서로 수정하는 프로세스 또는 운영 체제가 포함.



5. Resource Management

Multiprogramming (Batch system)

- 단일 사용자가 CPU 및 I/O 장치를 항상 바쁘게 유지할 수는 없다.
- 멀티프로그래밍은 작업(코드 및 데이터)을 구성하여 CPU가 항상 실행할 작업을 갖도록 한다.
- 시스템의 전체 작업 중 일부는 메모리에 보관.
- 하나의 작업을 선택하고 작업 예약을 통해 실행
- 작업이 대기해야 하는 경우(예: I/O) OS가 다른 작업으로 전환



5. Resource Management

Multitasking (Timesharing)

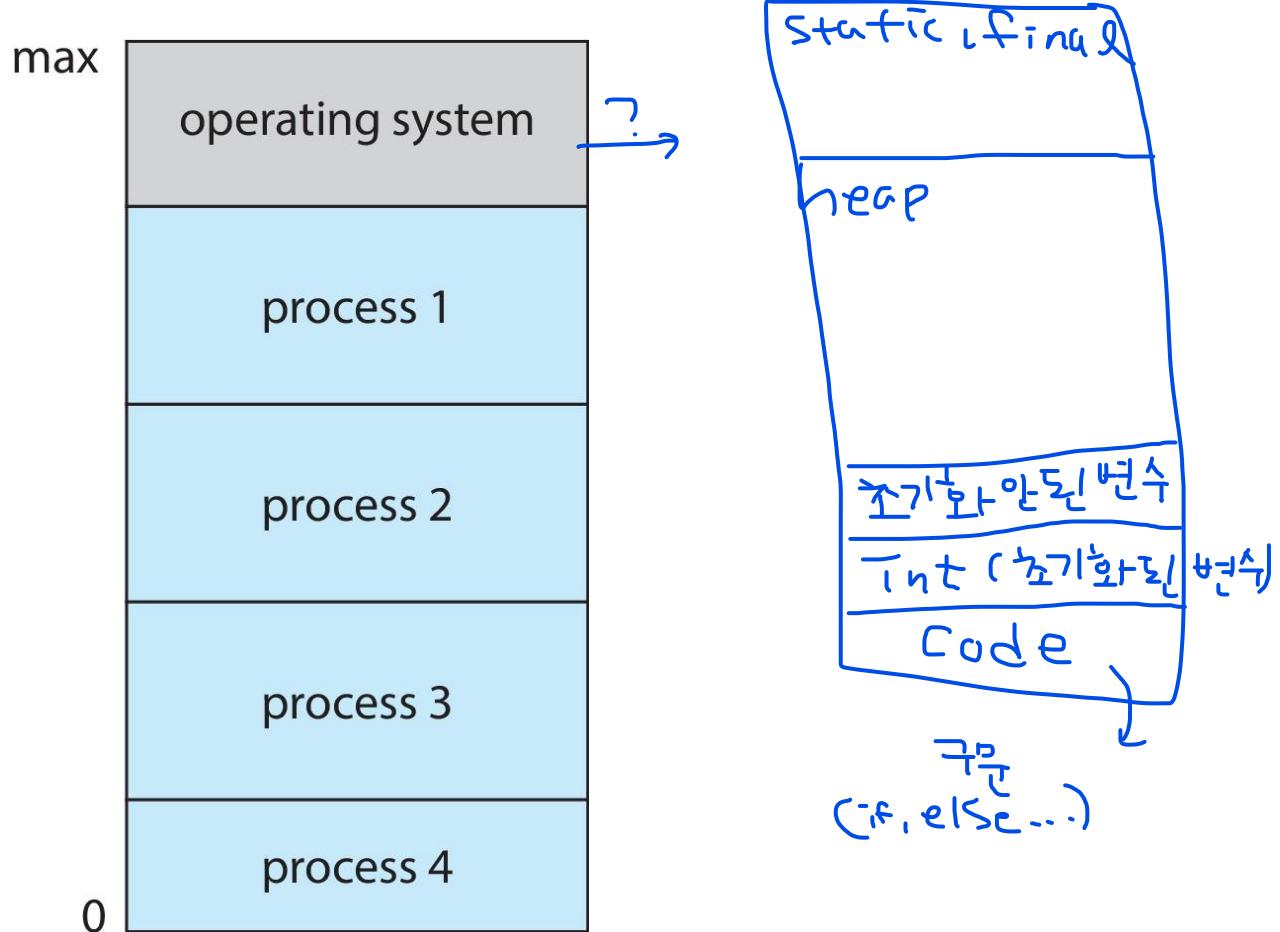
- Batch 시스템의 논리적 확장 - CPU가 작업을 자주 전환하므로 사용자는 실행 중인 각 작업과 상호 작용하여 대화형 컴퓨팅을 생성할 수 있.
- 응답 시간은 1초 미만이어야 한다.

Response time < 1 second

- 각 사용자는 메모리에서 실행되는 프로그램을 하나 이상 가지고 있다. > 프로세스
- 동시에 여러 작업을 실행할 준비가 된 경우 > CPU 스케줄링
- 프로세스가 메모리에 맞지 않으면 스와핑을 통해 프로세스를 안팎으로 이동하여 실행
- 가상 메모리는 완전히 메모리에 있지 않은 프로세스의 실행을 허용.

5. Resource Management

다중 프로그래밍 시스템을 위한 메모리 레이아웃





5. Resource Management

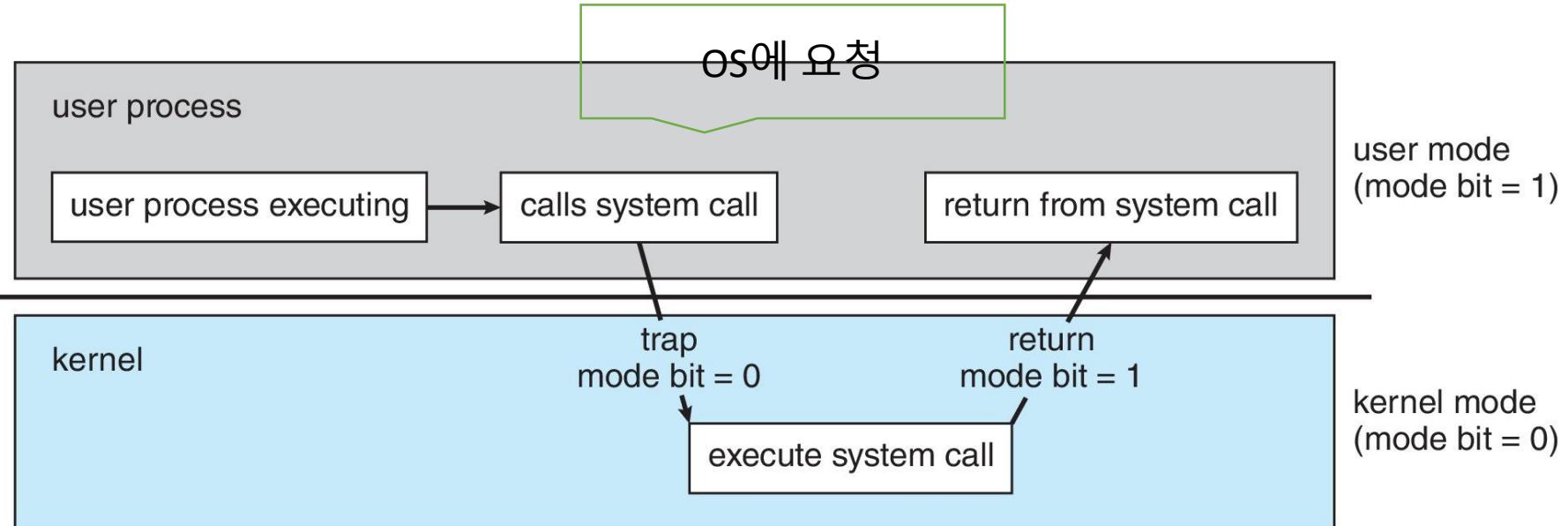
듀얼 모드 작동

- 이중 모드 작동으로 OS는 자체 및 기타 시스템 구성 요소를 보호할 수 있.
 - 사용자 모드와 커널 모드
- 하드웨어에서 제공하는 모드 비트
 - 시스템이 사용자 코드 또는 커널 코드를 실행 중일 때를 구별하는 기능을 제공.
 - 사용자가 실행 중일 때 > 모드 비트는 “사용자”
 - 커널 코드가 실행 중일 때 > 모드 비트는 "커널"
- 사용자가 모드 비트를 "커널"로 명시적으로 설정하지 않는다는 것을 어떻게 보장할까?
 - 시스템 호출은 모드를 커널로 변경하고 호출에서 반환하면 사용자로 재설정됨.
- 특권으로 지정된 일부 명령어는 커널 모드에서만 실행 가능



5. Resource Management

사용자에서 커널 모드로 전환



커널 모드에서만
실행



5. Resource Management

Timer

- 무한 루프를 방지하기 위한 타이머(또는 프로세스 독차지 리소스)
 - 일정 시간이 지나면 컴퓨터를 중단하도록 타이머가 설정되어 있다.
 - 물리적 시계에 의해 감소되는 카운터 유지
 - 운영 체제는 카운터를 설정합니다(특권 명령).
 - 카운터 제로가 인터럽트를 생성하는 경우
 - 제어권을 되찾거나 할당된 시간을 초과하는 프로그램을 종료하기 위해 프로세스를 예약하기 전에 설정



5. Resource Management

Process Management

- 프로세스는 실행 중인 프로그램. 시스템 내의 작업 단위. 프로그램은 수동적인 실체입니다. 프로세스는 활성 엔터티.
- 프로세스는 작업을 수행하기 위해 리소스가 필요.
 - CPU, 메모리, I/O, 파일
 - 초기화 데이터
- 프로세스를 종료하려면 재사용 가능한 리소스를 회수해야 한다.
- 단일 스레드 프로세스에는 실행할 다음 명령의 위치를 지정하는 하나의 프로그램 카운터가 있다.
 - 프로세스는 완료될 때까지 한 번에 하나씩 순차적으로 명령을 실행.
- 다중 스레드 프로세스에는 스레드당 하나의 프로그램 카운터가 있다.
- 일반적으로 시스템에는 하나 이상의 CPU에서 동시에 실행되는 많은 프로세스, 일부 사용자, 일부 운영 체제가 있다.
- 프로세스/스레드 간 CPU 다중화를 통한 동시성



5. Resource Management

프로세스 관리 활동

- 운영 체제는 프로세스 관리와 관련하여 다음 활동을 담당한다.
 - 사용자 및 시스템 프로세스 생성 및 삭제
 - 프로세스 일시 중지 및 재개
 - 프로세스 동기화를 위한 메커니즘 제공
 - 프로세스 통신을 위한 메커니즘 제공
 - 교착 상태 처리를 위한 메커니즘 제공



5. Resource Management

Memory Management

- 프로그램을 실행하려면 명령의 전부(또는 일부)가 메모리에 있어야 한다.
- 프로그램에 필요한 데이터의 전부(또는 일부)가 메모리에 있어야 함
- 메모리 관리는 메모리에 있는 내용과 시기를 결정.
 - CPU 사용률 및 사용자에 대한 컴퓨터 응답 최적화
- 메모리 관리 활동
 - 현재 사용 중인 메모리 부분과 누구에 의해 사용되는지 추적
 - 메모리 안팎으로 이동할 프로세스(또는 그 일부) 및 데이터 결정
 - 필요에 따라 메모리 공간 할당 및 할당 해제



5. Resource Management

File-system Management

- OS는 정보 저장에 대한 균일하고 논리적인 보기를 제공.
 - 물리적 속성을 논리적 저장 단위로 추상화 - 파일
 - 각 매체는 장치(즉, 디스크 드라이브, 테이프 드라이브)에 의해 제어.
 - ✓ 다양한 속성에는 액세스 속도, 용량, 데이터 전송 속도, 액세스 방법(순차적 또는 무작위)이 포함.
- 파일 시스템 관리
 - 일반적으로 디렉토리로 구성된 파일
 - 누가 무엇에 액세스할 수 있는지 결정하기 위해 대부분의 시스템에 대한 액세스 제어
 - OS 활동에는 다음이 포함.
 - ① 파일 및 디렉토리 생성 및 삭제
 - ② 파일 및 디렉토리 조작을 위한 프리미티브
 - ③ 보조 스토리지에 파일 맵핑
 - ④ 안정적인(비휘발성) 저장 매체에 파일 백업



5. Resource Management

대용량 스토리지 관리

- 일반적으로 메인 메모리에 맞지 않는 데이터 또는 "오랜" 기간 동안 보관해야 하는 데이터를 저장하는 데 사용되는 디스크
- 적절한 관리가 가장 중요
- 컴퓨터 작업의 전체 속도는 디스크 하위 시스템과 해당 알고리즘에 달려 있다.
- OS 활동
 - ① 마운트 및 마운트 해제
 - ② 여유 공간 관리
 - ③ 스토리지 할당
 - ④ 디스크 스케줄링
 - ⑤ 파티셔닝
 - ⑥ 보호



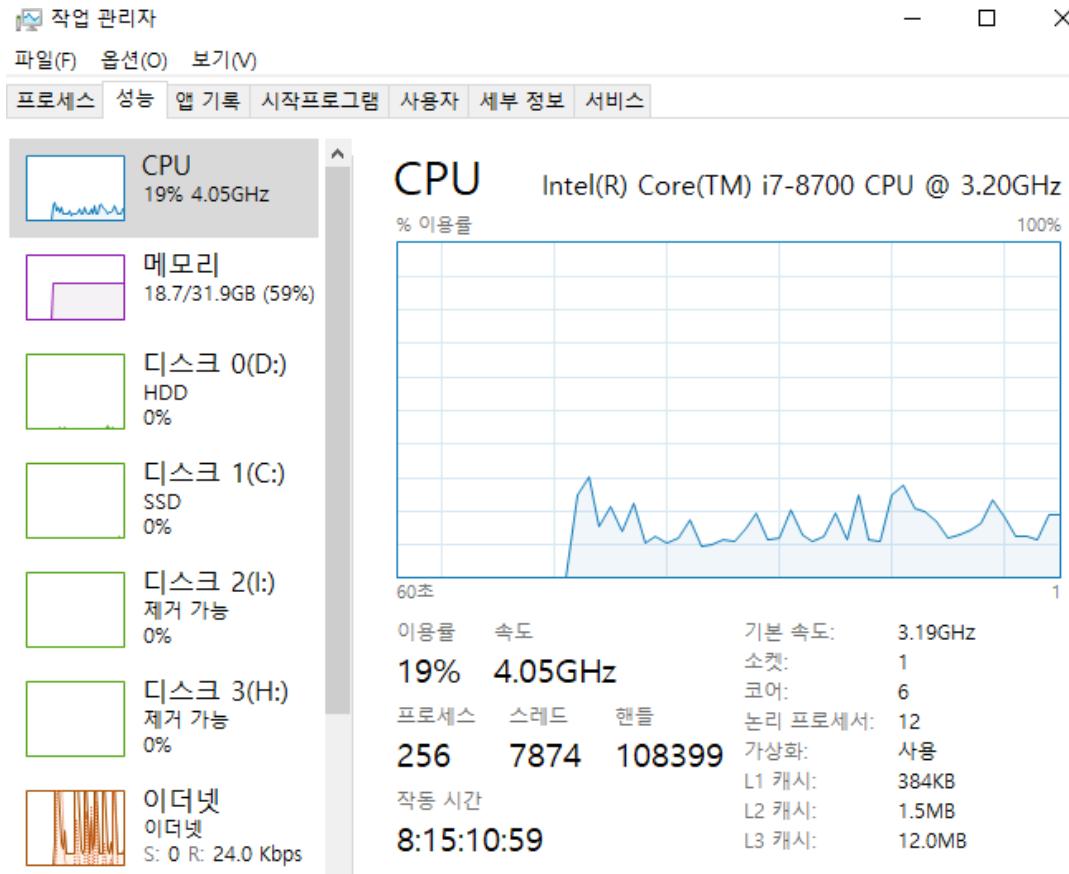
5. Resource Management

Caching

- 컴퓨터의 여러 수준에서 수행되는 중요한 원칙(하드웨어, 운영 체제, 소프트웨어)
- 사용 중인 정보가 느린 스토리지에서 빠른 스토리지로 일시적으로 복사됨
- 정보가 있는지 확인하기 위해 먼저 확인되는 더 빠른 스토리지(캐시)
 - 그렇다면 정보를 캐시에서 직접 사용(빠름)
 - 그렇지 않은 경우 데이터가 캐시에 복사되어 사용.
- 캐시되는 스토리지보다 작은 캐시
 - 캐시 관리 중요한 설계 문제
 - 캐시 크기 및 교체 정책

5. Resource Management

캐시



5. Resource Management

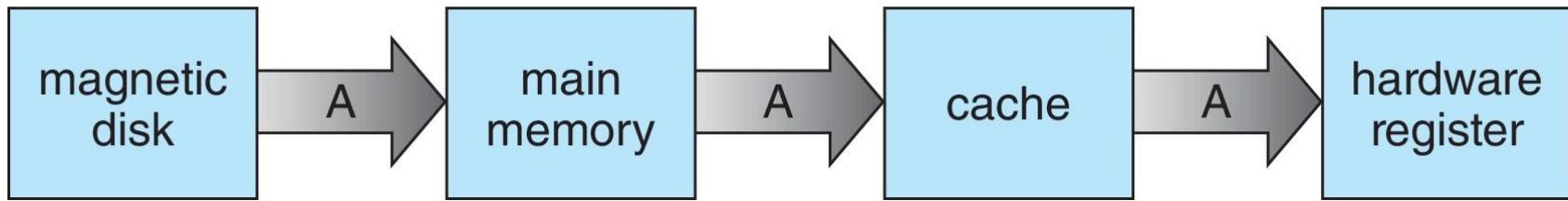
다양한 스토리지 유형의 특성

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

5. Resource Management

데이터 "A"를 디스크에서 레지스터로 마이그레이션

- 멀티태스킹 환경은 스토리지 계층 구조의 어디에 저장되어 있든 가장 최근 값을 사용하도록 주의해야 한다.



- 멀티프로세서 환경은 모든 CPU가 캐시에 최신 값을 갖도록 하드웨어에서 캐시 일관성을 제공해야 한다.
- 더욱 복잡한 분산 환경 상황
- 데이터의 여러 복사본이 존재할 수 있다.



5. Resource Management

데이터의 정의

- 데이터(data)라는 용어는 1646년 영국 문헌에 처음 등장한 후, 1940년 컴퓨터 시대 시작과 함께 자연과학뿐만 아니라 경영학, 통계학 등 다양한 사회과학이 진일보하며, 데이터의 의미는 과거의 관념적이고 추상적인 개념에서 기술적이고 사실적인 의미로 변화
- '데이터(DATA)'란 말의 어원은 라틴어에서 '주다'라는 뜻을 가진 'DARE'라는 동사다. 이것의 수동태인 '주어지다', '주어지는', '주어진 것' "present/gift, that which is given, debit"이라는 뜻의 단어가 'DATUM'인데 이것의 복수형 명사가 'DATA'이다. 즉 데이터란 '주어진 것들'이라는 뜻이므로 우리 눈앞에 주어진, 실재하는 세상 모든 것들이 데이터라고 할 수 있다.¹⁾
- 데이터는 추론과 추정의 근거를 이루는 사실²⁾)
- 데이터는 단순한 객체로서의 가치뿐만 아니라 다른 객체와의 상호관계 속에서 가치를 갖는 것

1) 출처 : https://eiec.kdi.re.kr/material/clickView.do?click_yymm=201512&cidx=2014

2) 옥스퍼드 대사전



5. Resource Management

데이터의 여러 복사본 존재

- 데이터의 복사본이 존재하는 경우에는 동일한 데이터를 여러 개의 변수 또는 객체에 저장할 수 있습니다. 이러한 복사본은 동일한 값을 가지지만 서로 다른 메모리 위치에 저장되어 있습니다. 복사본을 사용하는 이유는 여러 개의 변수나 객체가 동일한 데이터를 독립적으로 사용해야 하는 경우입니다.
- 예를 들어, 언어에서 배열이나 리스트를 사용하는 경우, 동일한 값을 가지는 데이터를 배열 또는 리스트의 각 요소에 복사하여 저장할 수 있습니다. 이렇게 하면 배열 또는 리스트의 각 요소는 독립적으로 값을 변경하거나 참조할 수 있습니다.
- 복사본이 존재할 때는 주의해야 할 사항도 있습니다. 복사본은 메모리 공간을 차지하므로 메모리 사용량이 증가할 수 있습니다. 또한, 복사본이 서로 다른 값을 가지는 경우, 데이터의 일관성을 유지해야 하는 문제가 발생할 수 있습니다.



5. Resource Management

I/O Subsystem

- OS의 한 가지 목적은 사용자로부터 하드웨어 장치의 특성을 숨기는 것.
- 다음을 담당하는 I/O 하위 시스템
 - ① 버퍼링(데이터가 전송되는 동안 일시적으로 데이터 저장), 캐싱(성능을 위해 데이터의 일부를 더 빠른 스토리지에 저장), 스팔링(한 작업의 출력과 다른 작업의 입력 중첩)을 포함한 I/O의 메모리 관리
 - ② 일반 장치 드라이버 인터페이스
 - ③ 특정 하드웨어 장치용 드라이버



6. 보안 및 보호

Protection and Security

- 보호 **Protection** – OS에서 정의한 리소스에 대한 프로세스 또는 사용자의 액세스를 제어하는 모든 메커니즘
- 보안 **Security** - 내부 및 외부 공격에 대한 시스템 방어
 - 서비스 거부, 웜, 바이러스, 신원 도용, 서비스 도용을 포함한 광범위한 범위
- 시스템은 일반적으로 누가 무엇을 할 수 있는지 결정하기 위해 먼저 사용자를 구분.
- 사용자 ID(사용자 ID, 보안 ID)에는 사용자당 하나의 이름 및 관련 번호가 포함.
- 그런 다음 사용자 ID는 액세스 제어를 결정하기 위해 해당 사용자의 모든 파일, 프로세스와 연결.
- 그룹 식별자(그룹 ID)를 사용하면 사용자 집합을 정의하고 제어를 관리한 다음 각 프로세스, 파일과 연결할 수 있다.
- 권한 에스컬레이션을 통해 사용자는 더 많은 권한을 가진 유효한 ID로 변경할 수 있다.



7. Virtualization

Virtualization

- 운영 체제가 다른 OS 내에서 응용 프로그램을 실행할 수 있다.
 - 방대하고 성장하는 산업
- Emulation 소스 CPU 유형이 대상 유형과 다를 때 사용(예: PowerPC에서 Intel x86으로)
 - 일반적으로 가장 느린 방법
컴퓨터 언어가 네이티브 코드로 컴파일되지 않은 경우 – Interpretation
- 가상화 Virtualization – CPU용으로 기본적으로 컴파일된 OS, 실행 중인 게스트 OS도 기본적으로 컴파일됨
 - 기본 WinXP host OS에서 각각 애플리케이션을 실행하는 WinXP 게스트를 실행하는 VMware를 고려.
 - VMM(virtual machine Manager :가상 머신 관리자)은 가상화 서비스를 제공.



7. Virtualization

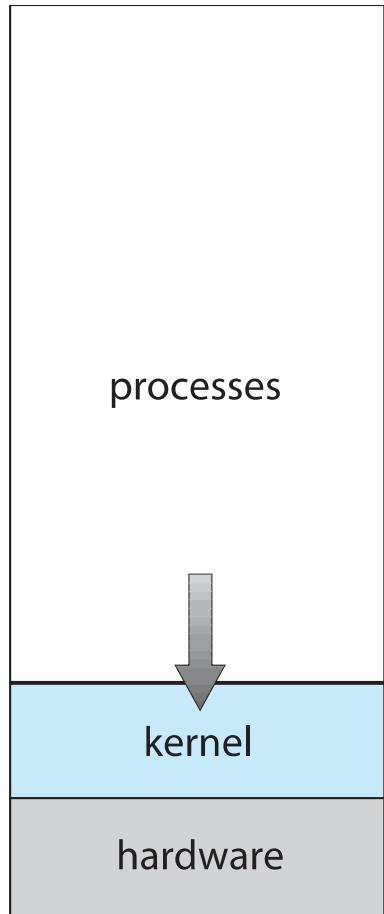
Virtualization

- 사용 사례에는 탐색 또는 호환성을 위해 여러 OS를 실행하는 랩톱 및 데스크톱이 포함.
 - Mac OS X 호스트, Windows를 게스트로 실행하는 Apple 노트북
 - 여러 시스템 없이 여러 OS용 앱 개발
 - 여러 시스템이 없는 품질 보증 테스트 애플리케이션
 - 데이터 센터 내 컴퓨팅 환경 실행 및 관리
- VMM은 기본적으로 실행될 수 있으며 이 경우 호스트이기도 하다.
- 범용 호스트가 없다(VMware ESX 및 Citrix XenServer).

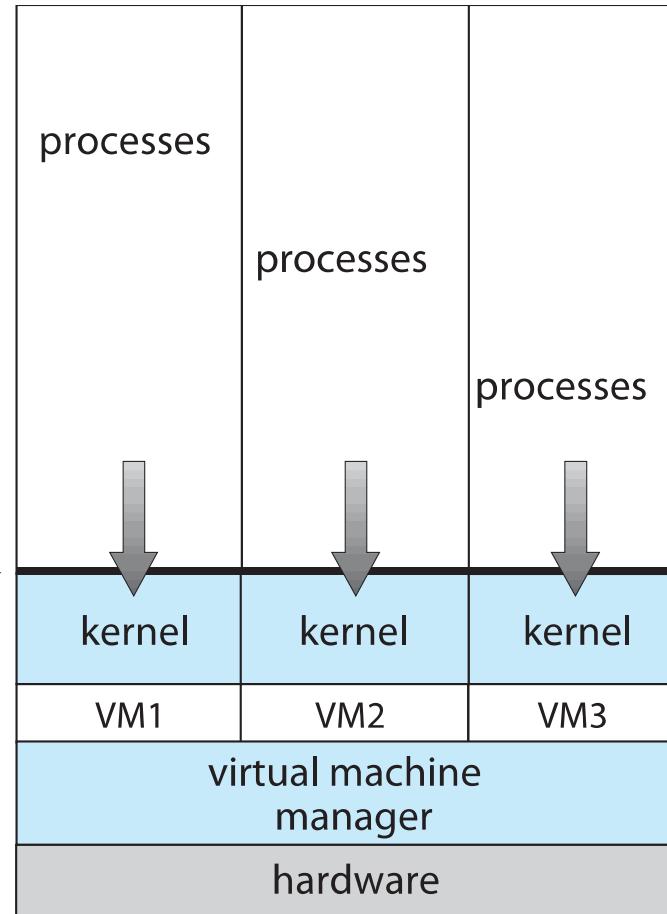


7. Virtualization

Computing Environments - Virtualization



(a)



(b)



8. Distributed Systems

Distributed Systems

- 서로 네트워크로 연결된 별도의 이기종 시스템 모음
- 네트워크는 통신 경로이며 TCP/IP가 가장 일반적.
 - 근거리 통신망(LAN)
 - 광역 네트워크(WAN)
 - 수도권 네트워크(MAN)
 - 개인 영역 네트워크(PAN)
- 네트워크 운영 체제는 네트워크를 통해 시스템 간에 기능을 제공.
 - 통신 체계는 시스템이 메시지를 교환할 수 있도록 합니다.
 - 단일 시스템의 환상



9. Computer System Architecture

컴퓨터 시스템 아키텍처

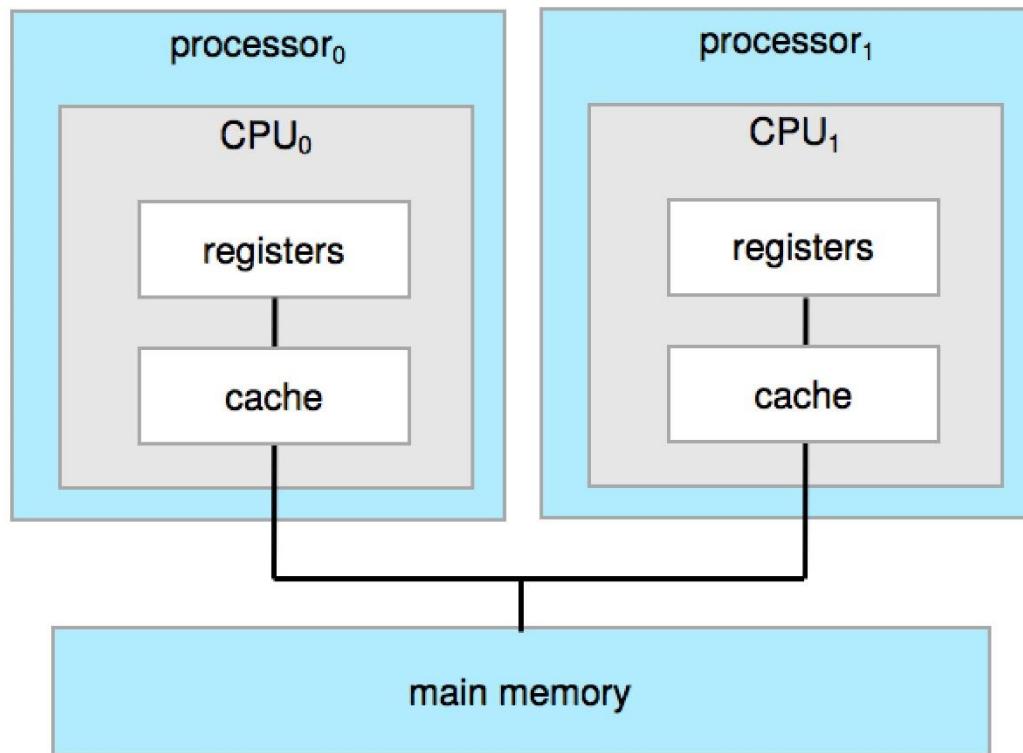
- 대부분의 시스템은 단일 범용 프로세서를 사용.
 - 대부분의 시스템에는 특수 목적 프로세서도 있다.
- 사용 및 중요성이 증가하는 다중 프로세서 시스템 **Multiprocessors**
 - 병렬 시스템, 강결합 시스템 **tightly-coupled systems**이라고도 함
 - 이점은 다음과 같다.
 - ① 처리량 증가
 - ② 규모의 경제
 - ③ 향상된 안정성 – 점진적 저하 또는 내결함성
- 두 가지 유형:
 - ① 비대칭 멀티프로세싱 **Asymmetric Multiprocessing** - 각 프로세서에 특정한 작업이 할당.
 - ② 대칭 다중 처리 **Symmetric Multiprocessing** - 각 프로세서가 모든 작업을 수행.



9. Computer System Architecture

Symmetric Multiprocessing Architecture

- **Symmetric Multiprocessing Architecture(SMP)**는 멀티프로세서 시스템의 일종으로, 여러 개의 프로세서가 동일한 작업을 동시에 처리할 수 있는 구조를 가지고 있습니다. SMP 시스템에서는 모든 프로세서가 메모리를 공유하고, 동일한 운영 체제 커널을 실행하며, 프로세서 간의 작업을 조정하는데 사용되는 통신 메커니즘을 가지고 있습니다.

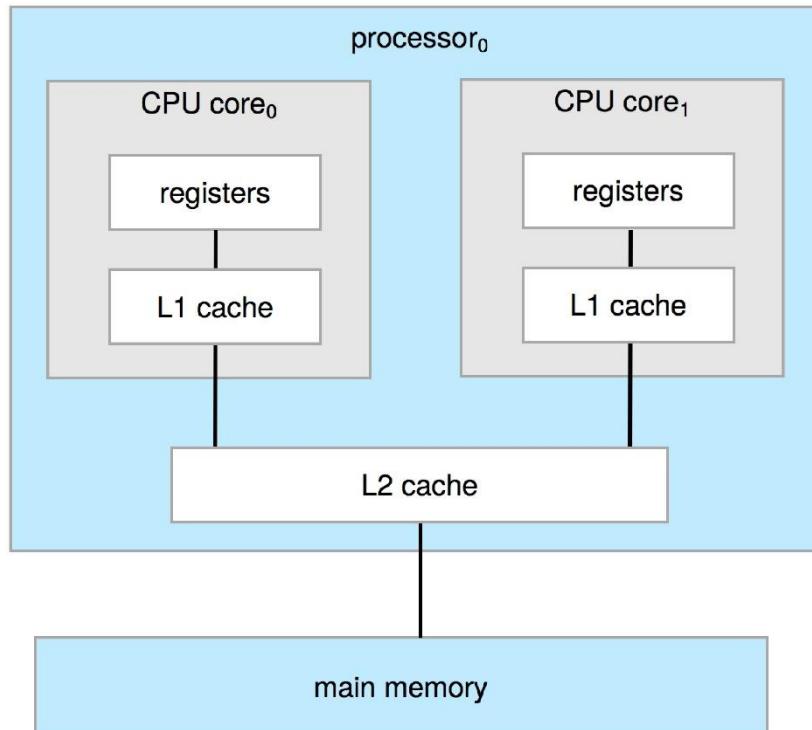




9. Computer System Architecture

Dual-Core Design

- 멀티칩 및 멀티코어
- 모든 칩을 포함하는 시스템
- 여러 개별 시스템을 포함하는 샐시

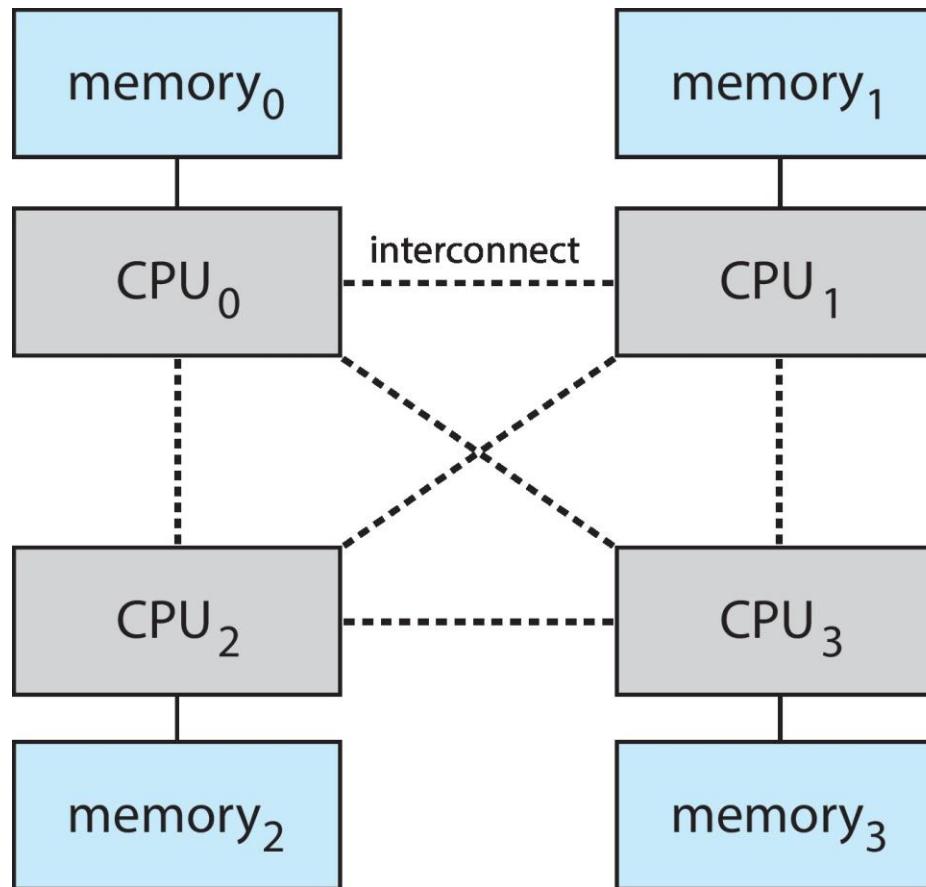




9. Computer System Architecture

Non-Uniform Memory Access System

- Non-Uniform Memory Access System 비균일 메모리 액세스 시스템





9. Computer System Architecture

Clustered Systems

- 멀티프로세서 시스템과 비슷하지만 함께 작동하는 여러 시스템
- 일반적으로 SAN(Storage-Area Network)을 통해 스토리지 공유
- 장애에도 살아남는 고가용성 서비스 제공
 - 비대칭 클러스터링 **Asymmetric clustering**에는 상시 대기 모드에 있는 하나의 시스템이 있다.
 - 대칭 클러스터링에 **Symmetric clustering**은 애플리케이션을 실행하는 여러 노드가 있으며 서로를 모니터링.
- 일부 클러스터는 고성능 컴퓨팅(HPC **high-performance computing**)용.
 - 애플리케이션은 병렬화를 사용하도록 작성되어야 한다.
 - 일부는 작업 충돌을 피하기 위해 분산 잠금 관리자(**DLM distributed lock manager**)를 사용한다.



10. Computer System Environments

Traditional

- 독립형 범용 기계
- 그러나 대부분의 시스템이 다른 시스템(즉, 인터넷)과 상호 연결됨에 따라 흐려짐
- 포털은 내부 시스템에 대한 웹 액세스를 제공합니다.
- 네트워크 컴퓨터(씬 클라이언트)는 웹 터미널과 같습니다.
- 무선 네트워크를 통해 모바일 컴퓨터 상호 연결
- 네트워킹이 유비쿼터스화 – 홈 시스템에서도 방화벽을 사용하여 홈 컴퓨터를 인터넷 공격으로부터 보호



10. Computer System Environments

Mobile

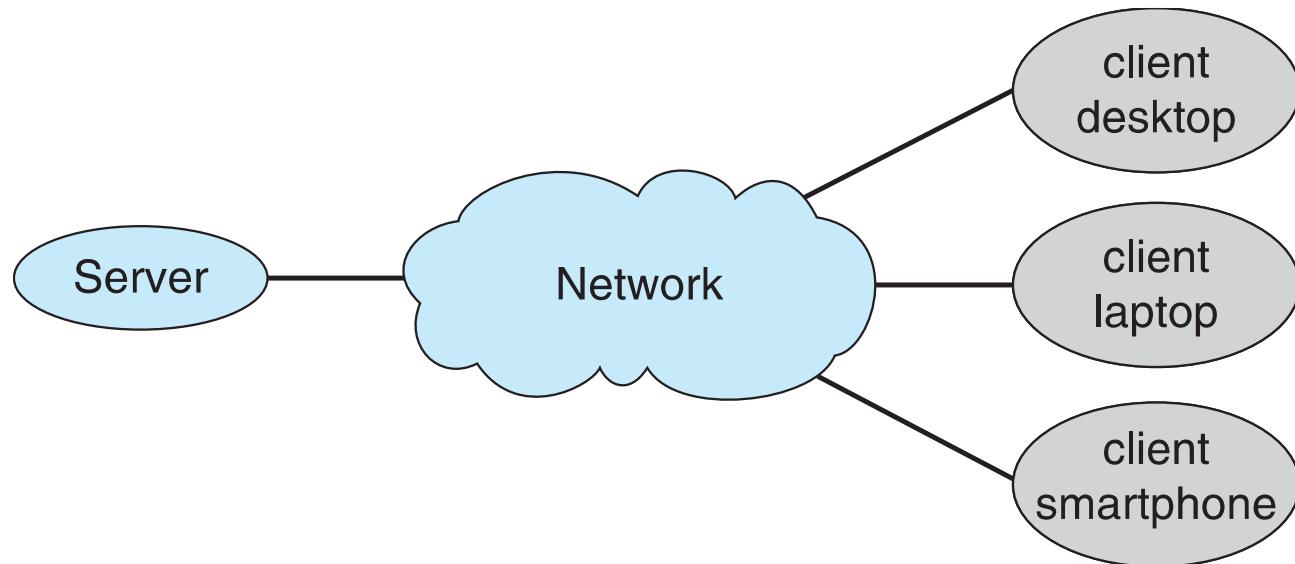
- 휴대용 스마트폰, 태블릿 등
- 그것들과 "전통적인" 노트북의 기능적 차이점은 무엇입니까?
- 추가 기능 – 더 많은 OS 기능(GPS, 자이로스코프)
- 증강 현실과 같은 새로운 유형의 앱 허용
- 연결을 위해 IEEE 802.11 무선 또는 셀룰러 데이터 네트워크 사용
- 리더는 Apple iOS 및 Google Android

10. Computer System Environments



Client Server

- 클라이언트-서버 컴퓨팅
- 스마트 PC로 대체된 멍청한 단말기
- 현재 많은 시스템이 클라이언트에서 생성된 요청에 응답하는 서버
 - 컴퓨터 서버 시스템은 서비스(즉, 데이터베이스)를 요청하는 클라이언트에 대한 인터페이스를 제공.
 - 파일 서버 시스템은 클라이언트가 파일을 저장하고 검색할 수 있는 인터페이스를 제공.

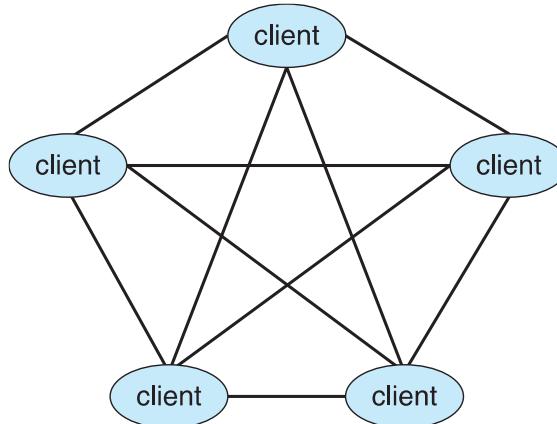


10. Computer System Environments



Peer-to-Peer

- 분산 시스템의 또 다른 모델
- P2P는 클라이언트와 서버를 구분하지 않는다.
 - 대신 모든 노드는 피어로 간주.
 - 각각 클라이언트, 서버 또는 둘 다로 작동할 수 있다.
 - 노드는 P2P 네트워크에 가입.
- ① 네트워크의 중앙 조회 서비스에 서비스를 등록하거나
- ② 서비스 요청 브로드캐스트 및 검색 프로토콜을 통한 서비스 요청 응답
- 예로는 Napster 및 Gnutella, Skype와 같은 VoIP(Voice over IP)가 있다.





10. Computer System Environments

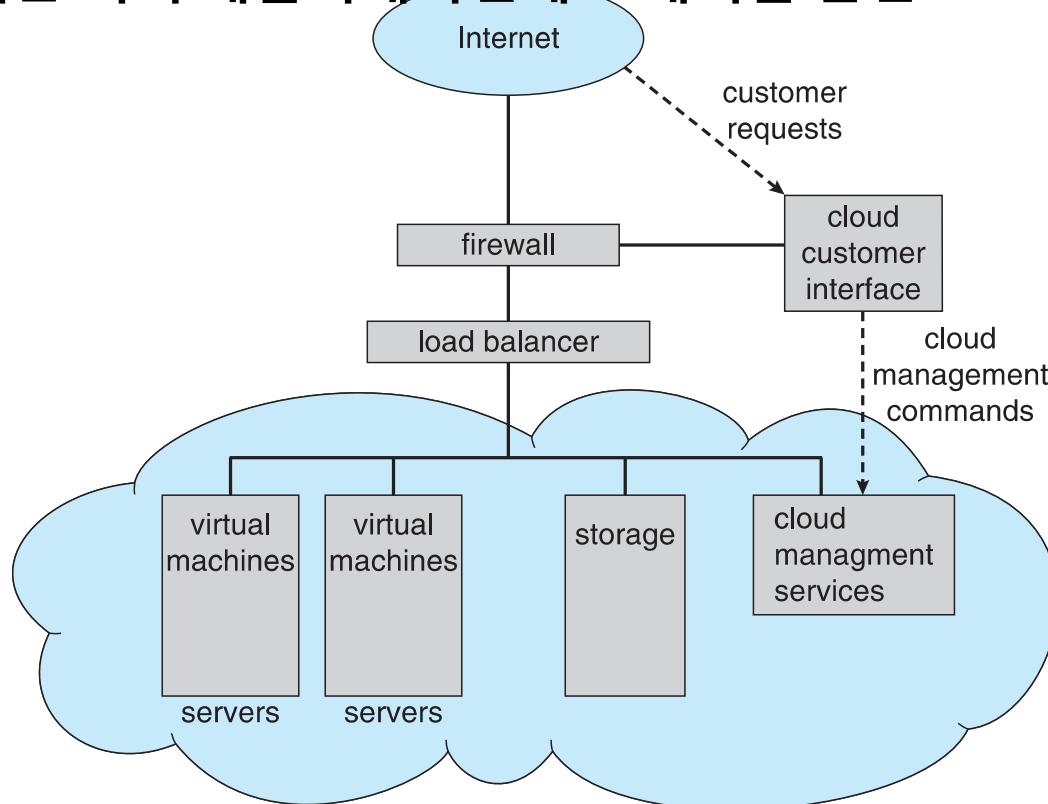
Cloud Computing

- 네트워크를 통해 컴퓨팅, 스토리지, 앱을 서비스로 제공
- 가상화를 기능의 기반으로 사용하기 때문에 가상화의 논리적 확장.
- Amazon EC2에는 수천 대의 서버, 수백만 대의 가상 머신, 인터넷에서 사용 가능한 페타바이트의 스토리지가 있으며 사용량에 따라 비용을 지불.
- 유형
 - 퍼블릭 클라우드 – 지불하려는 모든 사람이 인터넷을 통해 사용 가능
 - 사설 클라우드 – 회사 자체 사용을 위해 회사에서 운영
 - 하이브리드 클라우드 – 퍼블릭 및 프라이빗 클라우드 구성 요소를 모두 포함.
 - SaaS(Software as a Service) - 인터넷을 통해 사용할 수 있는 하나 이상의 응용 프로그램(예: 워드 프로세서)
 - PaaS(Platform as a Service) – 인터넷을 통해 애플리케이션을 사용할 준비가 된 소프트웨어 스택(즉, 데이터베이스 서버)
 - IaaS(Infrastructure as a Service) – 인터넷을 통해 사용할 수 있는 서버 또는 스토리지(즉, 백업용으로 사용할 수 있는 스토리지)

10. Computer System Environments

Cloud Computing

- 기존 OS, VMM, 클라우드 관리 도구로 구성된 클라우드 컴퓨팅 환경
- 인터넷 연결에는 방화벽과 같은 보안이 필요.
- 로드 밸런서는 여러 애플리케이션에 트래픽을 분산.





10. Computer System Environments

Real-Time Embedded Systems

- 컴퓨터의 가장 일반적인 형태의 실시간 임베디드 시스템
 - 다양한 특수 목적, 제한된 목적 OS, 실시간 OS
 - 확장 사용
- 다른 많은 특수 컴퓨팅 환경도 마찬가지.
 - 일부는 OS가 있고 일부는 OS 없이 작업을 수행.
- 실시간 OS에는 잘 정의된 고정 시간 제약이 있다.
 - 제약 내에서 처리해야 함
 - 제약조건이 충족된 경우에만 올바른 작동



10. Computer System Environments

무료 및 오픈 소스 운영 체제

- 바이너리 폐쇄 소스 및 독점이 아닌 소스 코드 형식으로 제공되는 운영 체제
- 복제 방지 및 디지털 권리(DRM) 운동에 대응
- "copyleft" GNU Public License(GPL)가 있는 Free Software Foundation(FSF)에서 시작.
- 자유 소프트웨어와 오픈 소스 소프트웨어는 서로 다른 그룹의 사람들이 옹호하는 서로 다른 두 가지 아이디어입니다.
- <https://www.gnu.org/philosophy/open-source-misses-the-point.en.html>
- GNU/Linux 및 BSD UNIX(Mac OS X의 핵심 포함) 등을 예로 들 수 있다.
- VMware Player(Windows에서 무료), Virtualbox(오픈 소스 및 많은 플랫폼에서 무료 - <http://www.virtualbox.com>)와 같은 VMM을 사용할 수 있다.
- 탐색을 위해 게스트 운영 체제를 실행하는 데 사용





1. 정보량 보충

정보량

- **R. V. Hartley**

- 발생 확률이 적은 사건은 큰 정보량을 갖고, 발생 확률이 큰 사건은 작은 정보량을 갖는다. 정보량을 수식적으로 정의하면 아마도 확률값에 반비례하는 값으로 정의

$$\text{Info} \propto \frac{1}{P(X)}$$

- 통계학에서 정보량은 다음과 같이 정의한다.
- 이산 랜덤변수 X 에 대해,

$$I(x) = -\log_b(P(X))$$

- 이 때, 로그의 밑 b 는 응용 분야에 따라 다르게 쓸 수 있는데, 대개 b 는 2, e, 10 중 하나를 사용할 수 있다. (각각을 사용했을 때의 정보량의 단위는 bit, nit, dit이다). BIT는 이진수 체계에서 사용되는 정보 단위이고, Dit는 모스 부호에서 신호의 길이를 나타내는 단위입니다. Qit는 양자 비트로서 양자 컴퓨팅에서 사용되며, Nit는 신경 비트로서 신경과학 및 신경공학에서 사용됩니다.



1. 정보량 보충

정보량

- **BIT**는 이진수 체계에서 사용되는 가장 작은 정보 단위로서, 0 또는 1의 값을 가집니다. 컴퓨터 과학 및 디지털 통신에서 널리 사용됩니다.
- **Dit**는 모스 부호(Morse Code)에서 사용되는 단위로서, 신호의 길이를 나타냅니다. 모스 부호는 단축된 점(.)과 긴 대시(-)를 사용하여 문자를 나타내는데, Dit은 모스 부호에서 점의 길이에 해당합니다. 따라서 Dit은 시간의 단위로 사용되며, 모스 부호에서 긴 대시의 길이는 3개의 Dit로 표현됩니다.
- **Qit**는 양자 비트(Quantum Bit)의 약자로서, 양자 컴퓨팅에서 사용되는 정보 단위입니다. 양자 비트는 양자 상태의 기본 단위로서, 양자 상태의 이중성 원리에 따라 0과 1의 동시 존재를 가능하게 합니다. 양자 비트는 양자 수학과 양자 연산을 기반으로 하는 양자 컴퓨터에서 정보를 처리하는 데 사용됩니다.
- **Nit**는 신경 비트(Neuron Bit)의 약자로서, 신경과학 및 신경공학 분야에서 사용되는 정보 단위입니다. 신경 비트는 신경세포 뉴런의 활동 상태를 나타내는데 사용됩니다. 신경비트는 일반적으로 뉴런의 활성화 여부를 나타내는 이진 값(0 또는 1)으로 표현되며, 신경망 모델과 관련된 연구 및 신경 시스템의 모델링에 활용됩니다.

1. 정보량 보충

정보량 계산 예

- 미국 통계분석사이트인 '파이브서티에이트'는 2018 러시아 월드컵에서 한국이 독일에게 승리할 확률을 5%로 예측했고, 독일이 승리할 확률은 81%로, 비길 확률은 14%로 예측했다.

https://www.chosun.com/site/data/html_dir/2018/06/23/2018062300599.html

스포츠 > 스포츠 포토

美통계, “韓 승리확률 멕시코전 19%, 독일전 5%”

OSEN

입력 2018.06.23. 08:13

가





1. 정보량 보충

정보량 계산 예

- 독일이 한국을 이기는 경우 정보량은 다음과 같다:

$$-\log_2 0.81 = 0.304$$

- 그리고 한국이 이기는 경우 정보량은 다음과 같다:

$$-\log_2 0.05 = 4.3219$$

- 마지막으로 두 팀이 비기는 경우 정보량은 다음과 같다:

$$-\log_2 0.14 = 2.8365$$



1. 정보량 보충

정보량 계산 예

- 계산한 바와 같이 한국이 이기는 경우가 독일이 이기는 경우와 비기는 경우보다 훨씬 더 정보량이 많다.

4.3219>2.8365>0.304

- 즉, 매우 놀라운 사건이라는 것이다. 두 팀이 비기는 것도 독일이 한국을 이기는 것보다 정보량이 훨씬 크다. 사실 두 팀이 비기는 것도 놀라운 결과였다는 것이다.

1. 정보량 보충

엔트로피란

● 엔트로피란 통계학에서 엔트로피란 평균 정보량

$$H(x) = E\{I(x_j)\} = - \sum_{j=1}^n p(x_j) \log_2 p(x_j)$$

X	x_1	x_2	x_3	\cdots	x_n	합계
$f(x)$	p_1	p_2	p_3	\cdots	p_n	1

$$\bar{x} = x_1 p_1 + x_2 p_2 + x_3 p_3 + \cdots + x_n p_n = \sum_{i=1}^n x_i p_i$$

확률변수 X 에 대해 다음과 같이 정의되는 수치 $\mu = E(X)$ 를 X 의 기댓값 expected value 또는 평균 mean이라 한다.

$$\mu = E(X) = \begin{cases} \sum_{\text{모든 } x} x f(x) & , \quad X \text{ 가 이산확률변수인 경우} \\ \int_{-\infty}^{\infty} x f(x) dx & , \quad X \text{ 가 연속확률변수인 경우} \end{cases}$$



1. 정보량 보충

엔트로피란

- 통계학에서의 엔트로피는 열역학에서 사용하는 Gibb's 엔트로피의 수식과 닮은 점이 있고, 정보 엔트로피(혹은 색년 엔트로피)는 평균 정보량이다.
- 주사위 놀이를 하는데, 1부터 6까지의 주사위 눈이 나왔을 때 각각 100원부터 600 원까지를 받는다고 생각해보자. 이 때의 기댓값은 다음과 같을 것이다.

$$\frac{1}{6} \times 100 + \frac{1}{6} \times 200 + \dots + \frac{1}{6} \times 600$$

$$= \sum_{i=1}^6 P(x_i)M(x_i)$$

- $M(x_i)$ 는 x_i 라는 사건 발생에 대해 받는 돈 (100원~600원)이라고 생각하자.
- 즉, 기댓값의 정의는 일어날 수 있는 사건에 대한 확률 \times 이벤트 값을 모두 합친 것이라고 할 수 있다. 따라서, 정보 엔트로피는 모든 일어날 수 있는 사건에 대한 확률 \times 정보량 값을 합친 것이므로, 따라서, 정보 엔트로피는 모든 일어날 수 있는 사건에 대한 확률 \times 정보량 값을 합친 것



1. 정보량 보충

엔트로피란

- 한국과 독일이 축구경기를 했을 때 엔트로피를 계산해보자. 즉, 평균 정보량은 다음과 같이 계산된다.
- $0.81(-\log_2 0.81) + 0.05(-\log_2 0.05) + 0.14(-\log_2 0.14) = 0.8595$
- 비교를 위해 스웨덴 vs 멕시코 전의 엔트로피도 계산하려고 한다. 미국 통계분석사이트인 '파이브서티에이트'는 2018 러시아 월드컵에서 스웨덴이 멕시코에게 이길 확률을 36%, 멕시코가 이길 확률을 34%, 비길 확률을 30%로 잡았다. 매우 박빙의 승부로 예측한 것이다. 그러면 스웨덴과 멕시코 경기의 엔트로피는 얼마일까?

$$0.36(-\log_2 0.36) + 0.34(-\log_2 0.34) + 0.30(-\log_2 0.30) = 1.5809$$



1. 정보량 보충

엔트로피란

- 1.5809비트로 한국과 독일전의 0.8595비트보다 약 두 배가량 크다. 따라서 스웨덴 vs 멕시코 전이 한국 vs 독일 전보다 정보량이 더 큰 경기였다고 볼 수 있다. 결과가 뻔히 예상되는 사건일 수록 엔트로피가 작고, 결과 예측이 힘들수록 엔트로피가 크다.



2. 컴퓨터 시스템 보충

1. 기능 분할

● 시스템과 기능

1. 시스템이란?

- 어떤 기능을 하기 위해 서로 어떤 작용을 주고받는 두 가지 이상의 부품으로 구성된 것
- 예) 망치 - 물체를 치는 기능을 위해 머리와 손잡이로 구성된 시스템
- 예) 문 - 열고 닫는 기능을 위해 몸체와 손잡이, 회전축, 지지대, 경첩 등으로 구성된 시스템

2. 기능(Function)이란?

- 기능이란 시스템이 하는 일, 즉 시스템의 ‘역할’을 뜻한다.

시스템	주요 기능
톱	물체를 자르는 역할
자동차	사람이나 물건을 이동시키는 역할
그릇	음식을 담는 역할
양산	빛을 차단시키는 역할



2. 컴퓨터 시스템 보충

1. 기능 분할

- 시스템에서 발생되는 모든 문제는 시스템 또는 부품의 문제가 아닌 기능의 문제
- 사람들이 구입하는 것 또한 제품 자체라기보다는 그 제품의 기능
- 시스템의 문제를 해결하기 위해서는 먼저 시스템을 기능으로 보는 연습 필요
- 기능 정의 방법
- 기능을 수행하는 주체인 도구(주어), 작용을 받는 대상(목적어), 작용(동사)을 모두 포함해야 함

예) 자동차는 사람을 이동시키는 역할을 한다.

주어

목적어

동사



2. 컴퓨터 시스템 보충

1. 기능 분할

● 기능 정의 사례1

- 선풍기의 기능 정의

잘못된 기능 정의

선풍기는 사람을 시원하게 하는 역할을 한다.

올바른 기능 정의

선풍기는 주변 공기를 움직여서 공기의 온도를 낮추는 역할을 한다.

주체인 선풍기가 대상인 사람을 어떻게 직접적으로 시원하게 변화시키는지가 명확하지 않으므로 정확한 표현이 아님

주체인 선풍기가 대상인 사람이 시원하게 느끼도록 하는데 어떤 역할을 하는지가 분명하게 표현됨



2. 컴퓨터 시스템 보충

1. 기능 분할

● 기능 정의 사례 2

시스템	잘못된 정의	올바른 정의
레이저 포인터	레이저 포인터는 발표 자료를 가리키는 역할을 한다.	레이저 포인터는 사람의 시선을 한 곳에 집중시키는 역할을 한다.
에어컨의 공기 필터	공기필터는 공기를 정화시키는 역할을 한다.	공기필터는 먼지를 잡는 역할을 한다.
진공 청소기	진공 청소기는 먼지를 제거하는 역할을 한다.	진공 청소기는 공기를 이동시키는 역할을 한다.(공기를 이동시키면 진공 상태가 되고, 그 결과 먼지가 다른 곳으로 이동하게 된다.)
칫솔	칫솔은 치아를 깨끗하게 하는 역할을 한다.	칫솔은 치아 표면이나 치아 사이의 이물질을 제거하는 역할을 한다.



2. 컴퓨터 시스템 보충

1. 기능 분할

- **기능 분석(Function Analysis)이란?**

- 시스템을 구성하는 부품, 환경 요소, 대상 간의 역할 관계를 분석하는 것
- 즉, 시스템의 본질을 파악하기 위해 시스템 및 구성 부품의 진짜 기능이 무엇인지 분석하는 것

1. **부품 - 시스템을 구성하는 구성 요소, 하위 시스템(subsystem)**

예) 자동차의 부품 : 프레임, 엔진, 바퀴 등

2. **환경 요소 - 시스템에 포함되지는 않지만 시스템에 영향을 주는 모든 것**

상위 시스템(supersystem)

예) 자동차의 환경 요소 : 도로, 연료, 매연 등

3. **대상 - 시스템을 이용해서 특성을 변화시키고 싶은 물체**

예) 자동차의 대상 : 사람, 물건 등



2. 컴퓨터 시스템 보충

1. 기능 분할

- **기능 분석(Function Analysis)이란?**

- 시스템을 구성하는 부품, 환경 요소, 대상 간의 역할 관계를 분석하는 것
- 즉, 시스템의 본질을 파악하기 위해 시스템 및 구성 부품의 진짜 기능이 무엇인지 분석하는 것

1. **부품 - 시스템을 구성하는 구성 요소, 하위 시스템(subsystem)**

- 예) 자동차의 부품 : 프레임, 엔진, 바퀴 등

2. **환경 요소 - 시스템에 포함되지는 않지만 시스템에 영향을 주는 모든 것**

- 상위 시스템(supersystem)
- 예) 자동차의 환경 요소 : 도로, 연료, 매연 등

3. **대상 - 시스템을 이용해서 특성을 변화시키고 싶은 물체**

- 예) 자동차의 대상 : 사람, 물건 등



2. 컴퓨터 시스템 보충

1. 기능 분할

● 기능 분석도 작성 기호

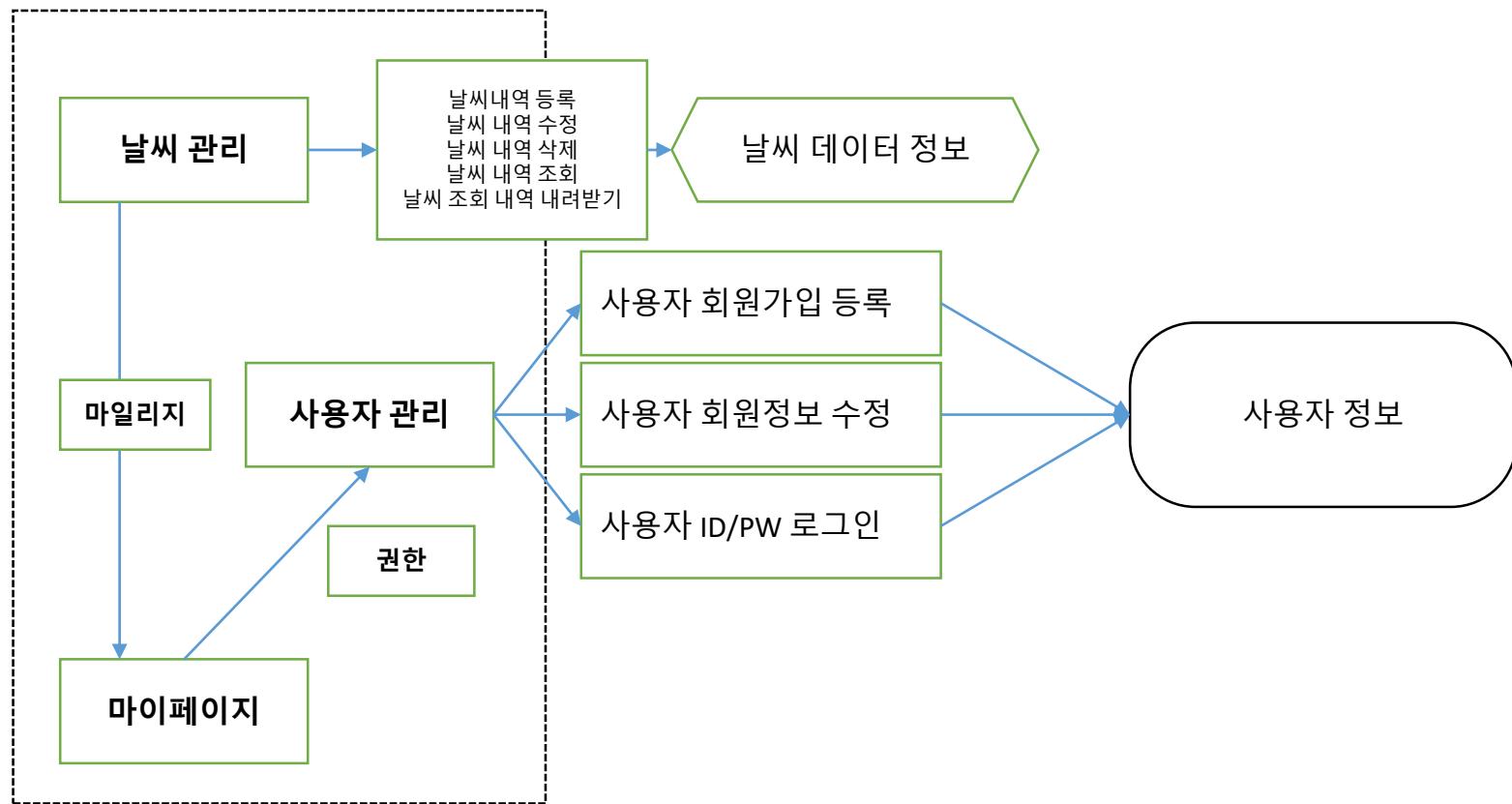
구분	기호	
시스템 및 부품	직사각형	
대상(객체)	둥근 사각형	
환경 요소(상위 시스템)	육각형	
유용한 기능	실선 화살표	→
유해한 기능	점선 화살표	-----→
상세 기능 분석도의 시스템	점선 사각형	



2. 컴퓨터 시스템 보충

1. 기능 분할

- 기능 분석도 작성 기호





2. 컴퓨터 시스템 보충

2. 반도체 기술의 진화와 집적 회로의 발전 단계

- **IC (Integrated Circuit):** 집적 회로를 의미하며, 반도체 칩에 여러 개의 전자 부품 및 회로가 집적되어 있는 것을 말합니다. 작은 크기와 높은 신호 처리 능력을 가지고 있으며, 주로 전자 기기와 컴퓨터 시스템에서 사용됩니다.
- **LSI (Large-Scale Integration):** 대규모 집적 회로를 의미합니다. LSI는 한 개의 반도체 칩에 수백 개에서 수천 개의 게이트와 전자 부품이 집적되어 있는 기술입니다. LSI는 IC 기술의 발전으로 탄생하였으며, 고밀도와 복잡한 기능을 가진 회로를 구현할 수 있습니다.
- **VLSI (Very Large-Scale Integration):** 매우 대규모 집적 회로를 의미합니다. VLSI는 한 개의 칩에 수십만 개에서 수천만 개의 트랜지스터가 집적되어 있는 기술입니다. VLSI는 LSI 보다 훨씬 더 높은 집적도와 더 복잡한 기능을 제공하며, 주로 현대 반도체 기술의 핵심입니다.
- **ULSI (Ultra Large-Scale Integration):** 초대규모 집적 회로를 의미합니다. ULSI는 한 개의 칩에 수억 개 이상의 트랜지스터가 집적되어 있는 기술로, VLSI보다 더 높은 집적도와 더 복잡한 기능을 갖추고 있습니다. ULSI는 최신 반도체 기술의 발전으로 이어지고 있으며, 고급 컴퓨터 프로세서와 메모리 칩 등에 사용됩니다.
- **SoC (System on a Chip):** 칩 내 시스템을 의미합니다. SoC는 전체 시스템의 주요 기능을 한 개의 반도체 칩에 통합하여 구현하는 기술입니다. 즉, 프로세서, 메모리, 입출력 인터페이스, 그래픽 처리 장치 등 다양한 하드웨어 컴포넌트가 한 개의 칩에 통합되어 있습니다. SoC는 주로 모바일 기기나 임베디드 시스템에서 사용되며, 높은 성능과 저전력 운영을 제공합니다.



2. 컴퓨터 시스템 보충

4. 플립-플롭, 데이터 버스, RF

- **플립-플롭 (Flip-Flop):** 플립-플롭은 디지털 논리 회로에서 데이터를 저장하고 전송하는 역할을 하는 순차 논리 회로입니다. 플립-플롭은 두 개의 안정 상태(0 또는 1)를 가지며, 클럭 신호에 의해 동작합니다. 가장 일반적인 플립-플롭은 D 플립-플롭(D Flip-Flop)으로, 입력 신호(D)가 클럭 신호의 상승 에지에 따라 현재 상태를 갱신합니다. 플립-플롭은 데이터 레지스터, 메모리, 카운터 등 다양한 응용 분야에서 사용됩니다.
- **데이터 버스 (Data Bus):** 데이터 버스는 컴퓨터 시스템에서 데이터 전송을 위해 사용되는 통신 경로입니다. 데이터 버스는 중앙 처리 장치(CPU), 메모리, 입출력 장치 등 각각의 구성 요소 사이에서 데이터를 전송하는 역할을 합니다. 데이터 버스는 병렬 데이터 전송을 지원하며, 데이터 비트의 크기에 따라 데이터 버스의 폭(비트 수)이 결정됩니다. 일반적으로 데이터 버스는 데이터, 주소, 제어 신호 등 다양한 정보를 전송하기 위해 사용됩니다.
- **RF (Radio Frequency):** RF는 무선 주파수(Radio Frequency)의 약어로, 무선 통신에서 사용되는 전자파의 주파수 범위를 나타냅니다. RF는 일반적으로 3 kHz에서 300 GHz 사이의 주파수를 포함하며, 라디오, 텔레비전, 무선 통신, 위성 통신, 블루투스, Wi-Fi 등 다양한 무선 통신 시스템에서 사용됩니다. RF는 데이터 전송, 신호 송수신, 잡음 제어 등 다양한 무선 통신 기술에 중요한 역할을 합니다.



2. 컴퓨터 시스템 보충

3. 무어의 법칙(Moore's Law)과 황의 법칙(Huang's Law)

- 무어의 법칙(Moore's Law)과 황의 법칙(Huang's Law)은 반도체 산업에서 중요한 법칙으로 알려져 있습니다.
- 무어의 법칙: 무어의 법칙은 인텔(Intel)의 공동 창업자인 고든 무어(Gordon Moore)에 의해 제안된 법칙으로, 1965년에 처음 발표되었습니다. 이 법칙은 초기에는 반도체 집적 회로의 집적도(트랜지스터의 개수)가 약 2년마다 2배씩 증가한다는 내용으로 제시되었습니다. 무어의 법칙은 후에 수정되어 집적도 증가 속도를 2년마다 2배에서 약 18~24개월마다 2배로 나 타내기도 합니다. 이 법칙은 반도체 산업에서 지속적인 기술 발전과 성능 향상을 기대할 수 있는 기준이 되었습니다.
- 황의 법칙: 황의 법칙은 네이비디아(NVIDIA)의 CEO인 재두 황(Jensen Huang)에 의해 제안된 개념으로, 2018년에 처음 언급되었습니다. 황의 법칙은 그래픽 처리 유닛(Graphics Processing Unit, GPU)의 성능이 무어의 법칙에 비해 더 빠르게 발전한다는 것을 주장합니다. 즉, GPU의 성능은 약 2년마다 2배가 아닌, 더 빠른 속도로 향상된다는 것을 의미합니다. 황의 법칙은 GPU 기술의 진화와 인공지능, 딥러닝 분야에서의 역할을 강조하고 있습니다.
- 무어의 법칙은 집적 회로의 발전과 성능 향상을 예측하였으며, 황의 법칙은 특정 분야(주로 GPU)에서의 성능 향상을 강조한 개념입니다.