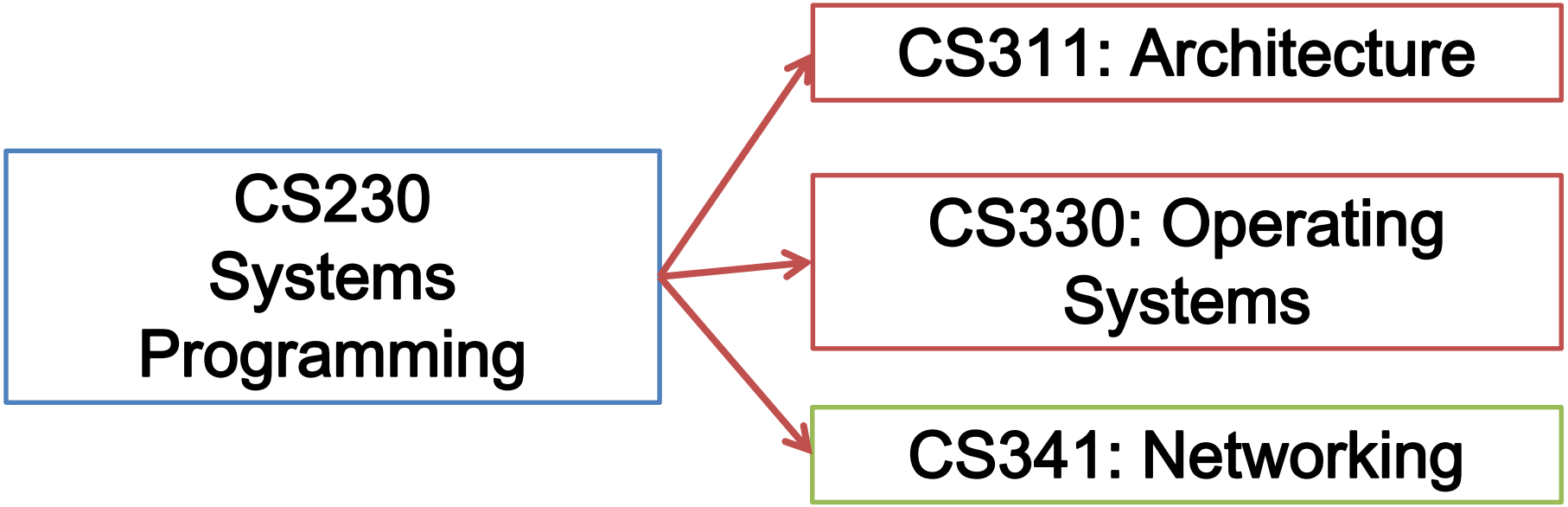
CS330: Introduction

Who am I?

- Youngjin Kwon, Assistant Professor at KAIST
  - who has studied operating systems
  - who loves designing/building systems
  - who enjoys low-level details about computer system
- Office: E3-1, 4405
- Email: yjkwon@cs.kaist.ac.kr

Prerequisites

- Prerequisites
  - CS230 is strongly recommended, and CS211 is preferably recommended

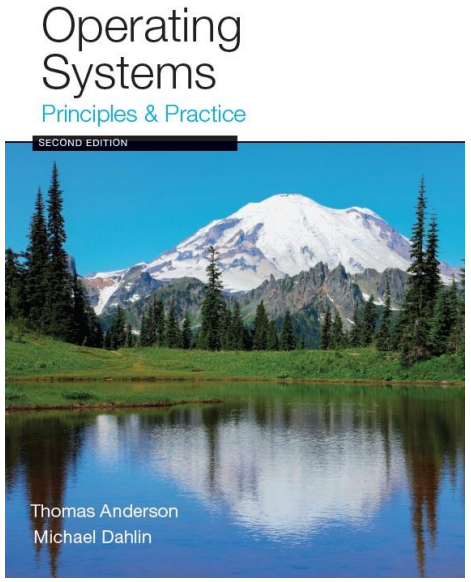


Prerequisites

- You are expected to have
  - Basic understanding of key OS concepts
    - process, system call, virtual memory, file
    - The schedule of programming project goes ahead of lecture
  - Entry level of C programming experience
    - e.g., pointers to structures/arrays
    - Linux programming environment (GNU C)
    - Systems calls, file I/O programming, signal handling

547 -

Logistics



- Textbook
  - Operating Systems Principles & Practice
- Discussion board

  https://piazza.com/kaist.ac.kr/spring2019/cs330/home
- Course webpage

  https://sites.google.com/view/cs330/

  http://klms.kaist.ac.kr

Learning pyramid

- A various percentage of retention of learning with different ways of teaching methods
- Studied by National Training Laboratory

	% of retention
Lecturing (listening)	5%
Reading	10%
Demonstration	30%
Discussion	50%
Practice by doing	70%
Teaching others	90%

Credit criteria

- Mid-term exam: 20%
- Hands-on labs: 40%
- Final exam: 30%
- Pintos exam: 10%

Project: Pintos

- Educational operating system
  - Simplified OS with essential features

- We give you some basic building blocks
  - Four assignments, that build on each other
    - Threads, user programs, virtual memory
  - Work in **groups of 2**
    Pair-programming:
    https://en.wikipedia.org/wiki/Pair_programming

- Project document

Short Contents

1 Introduction 1
2 Project 1: Threads 9
3 Project 2: User Programs 22
4 Project 3: Virtual Memory 39
5 Project 4: File Systems 50
A Reference Guide 58

Project: Pintos

- How to download pinto code will be announce in the discussion board or course webpage

- Four implementation projects:
  - Threads (Thread support, synchronization)
  - User programs
  - Virtual memory
  - File systems

- Project Guide & Help Sessions by TA
    - Wednesday, 7pm – 10pm

Project Sessions (by TAs)

- Project sessions (online help)
  - **Lab hours** for programming help
    - Programming help for 25 min per team
    - You must request by one day before the lab hours day, describing the problem you want to ask
      - Please make sure your question is specific and clear
        - » TAs cannot help you if your question is too general
        - » Good example: *In the test case A, I have a problem at the line #XX*
          - *with explanation of your design and where you changed*
        - » Bad example: *My PintOS is not working*
- Due date is **Sunday midnight in general**
  - A late penalty is 20% off per day (up to 3 days)
  - 5 tokens for an entire semester
    - You can use a token for penalty waiver
  - Once your team submits a project after using all the penalty waivers, you will still get 35% of your score **(do not give up!)**

Course Homepage

- Three assignments spread over quarter
  - Practice for final
  - Done **individually**

Main Points (for today)

- Operating system definition
  - Software to manage a computer's resources for its users and applications
- OS challenges
  - Reliability, security, responsiveness, portability, …
- How to understand OS?

What's going on?

```c
#include <stdio.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/time.h>

int main(void)
{
  void *addr;
  struct timeval start, end, elap;

  addr = mmap(NULL, 1 << 20, PROT_READ | PROT_WRITE,
      MAP_ANONYMOUS| MAP_PRIVATE, -1, 0);

  gettimeofday(&start, NULL);
  memset(addr, 1, 1 << 20);
  gettimeofday(&end, NULL);

  timersub(&end, &start, &elap);

  printf("Time taken to memset1 %ld usec\n", elap.tv_usec);

  gettimeofday(&start, NULL);
  memset(addr, 2, 1 << 20);
  gettimeofday(&end, NULL);

  timersub(&end, &start, &elap);

  printf("Time taken to memset2 %ld usec\n", elap.tv_usec);

  munmap(addr, 1 << 20);

  return 0;
}
```

```
[ yjkwon@tigris02 ~] > gcc -o map map.c
[ yjkwon@tigris02 ~] > ./map
Time taken to memset1 1389 usec
Time taken to memset2 112 usec
[ yjkwon@tigris02 ~] >
```

Lesson from computer architecture..
CPU cache locality

MAP size is way beyond CPU cache size!

```c
#include <stdio.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/time.h>

#define MAP_SIZE (1 << 30)

int main(void)
{
  void *addr;
  struct timeval start, end, elap;

  addr = mmap(NULL, MAP_SIZE, PROT_READ | PROT_WRITE,
      MAP_ANONYMOUS| MAP_PRIVATE, -1, 0);

  gettimeofday(&start, NULL);
  memset(addr, 1, MAP_SIZE);
  gettimeofday(&end, NULL);

  timersub(&end, &start, &elap);

  printf("Time taken to memset1 %0.2lf msec\n",
      (((double)elap.tv_sec * 1000000.0) + (double)elap.tv_usec) / 1000.0 );

  gettimeofday(&start, NULL);
  memset(addr, 2, MAP_SIZE);
  gettimeofday(&end, NULL);

  timersub(&end, &start, &elap);

  printf("Time taken to memset2 %0.2lf msec\n",
      (((double)elap.tv_sec * 1000000.0) + (double)elap.tv_usec) / 1000.0 );

  munmap(addr, MAP_SIZE);

  return 0;
}
```

```
[ yjkwon@tigris02 ~] > !gcc
[ yjkwon@tigris02 ~] > gcc -o map map.c
[ yjkwon@tigris02 ~] > ./map
Time taken to memset1 296.62 msec
Time taken to memset2 155.95 msec
[ yjkwon@tigris02 ~] >
```

Why?

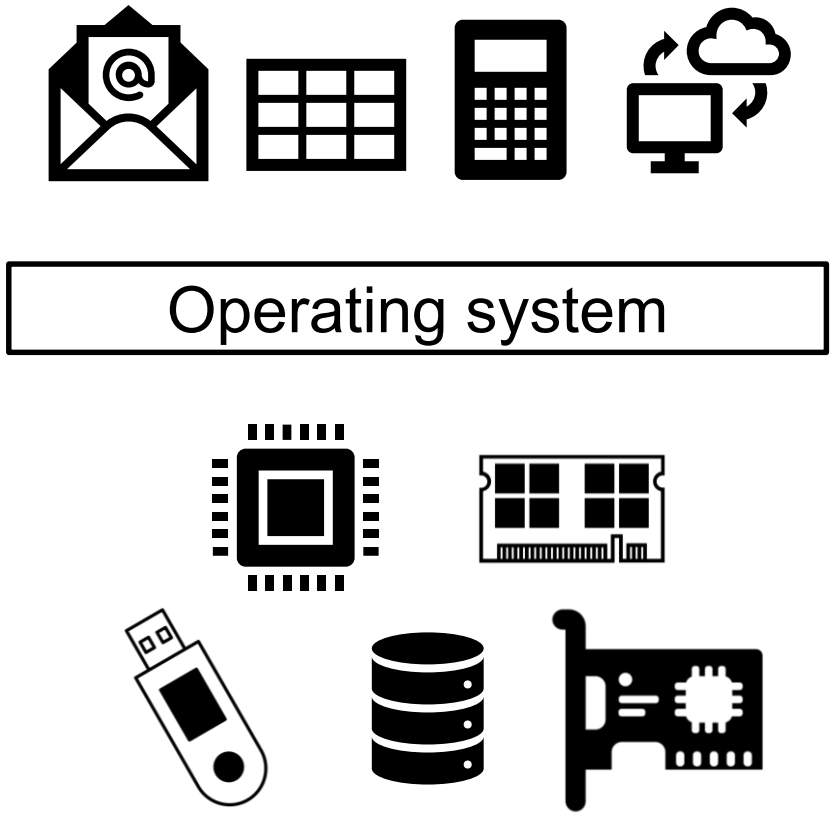
Why is studying OS useful?

- “How operating system works” is essential for any students interested in build modern computer systems like web server, KV store etc

- ***OS as a reference material***
  - Software engineers use many of the same techniques and design patterns as those use in OS

Main Role of OS

- To programs,
  - Providing application programming interface (API) to use hardware
  - Hide details of hardware



OS definition

- An Operating System is a layer of systems software that
  - Has privileged access to the hardware directly;
  - Hide the hardware complexity;
  - Manages hardware on behalf of applications;
  - Additionally, ensures that applications are isolated and protected each other

Question

- Which of the followings are likely components of an OS?
  - File editor
  - Browser
  - File system
  - Device D river
  - Cache memory
  - Process Scheduler
  - C standard library

Design + Implementation

Operating System Roles

- **Referee:** managing hardware and application activities
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications

- **Illusionist:** masking limitations of hardware
  - Each application appears to have the entire machine to itself
  - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport

- **Glue:** providing a set of common, standard services to applications
  - Interface to standard C library, Graphic interfaces

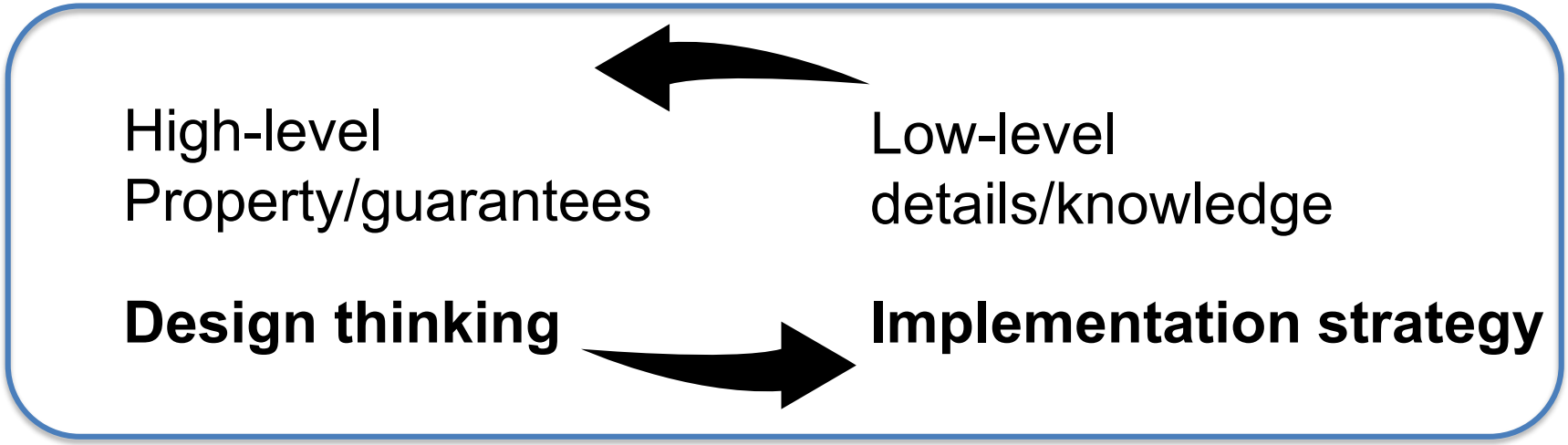
OS Challenges

- Performance
  - Latency/response time
    - How long does an operation take to complete?
  - Throughput
    - How many operations can be done per unit of time?
  - Overhead
    - How much extra work is done by the OS?
  - Fairness
    - How equal is the performance received by different users?
  - Predictability
    - How consistent is the performance over time?

OS Challenges

- Security
  - Can the system be compromised by an attacker?
- Privacy
  - Data is accessible only to authorized users
- Reliability
  - Does the system do what it was designed to do?
- Availability
  - What portion of the time is the system working?
  - Mean Time To Failure (MTTF), Mean Time to Repair

How to understand OS?



"Design Rationale"

For example, **Protection**
- The goal of protection
- Protection model/API
- Efficient mechanisms

- How to use hardware mechanism?
- What part is done by hardware? to be done by software?
- How to separate software layers?

OS in a bird’s-eye view

OS guarantees:

1.
2.
3.
4.

- You need to fill them at the end of the semester!