

# Lecture Note 0.

## Lecture Overview

August 26, 2021

Jongmoo Choi  
Dept. of Software  
Dankook University

<http://embedded.dankook.ac.kr/~choijm>

(Copyright © 2021 by Jongmoo Choi, All Rights Reserved. Distribution requires permission)

# Course Objective

---

## ■ What is System Programming?

- ✓ Application program vs. System program

```
#include <stdio.h>

int main()
{
    printf("Hello, World\n");
}
```

- ☞ How to run this program on CPU?
- ☞ What is the role of printf()?
- ☞ How the string is displayed on Monitor?
- ☞ How this program can be executed with other programs concurrently?
- ☞ What are the differences between local and global variables?
- ☞ What if we split the string "Hello, World\n" into two strings with two printf()s?



# Course Objective

---

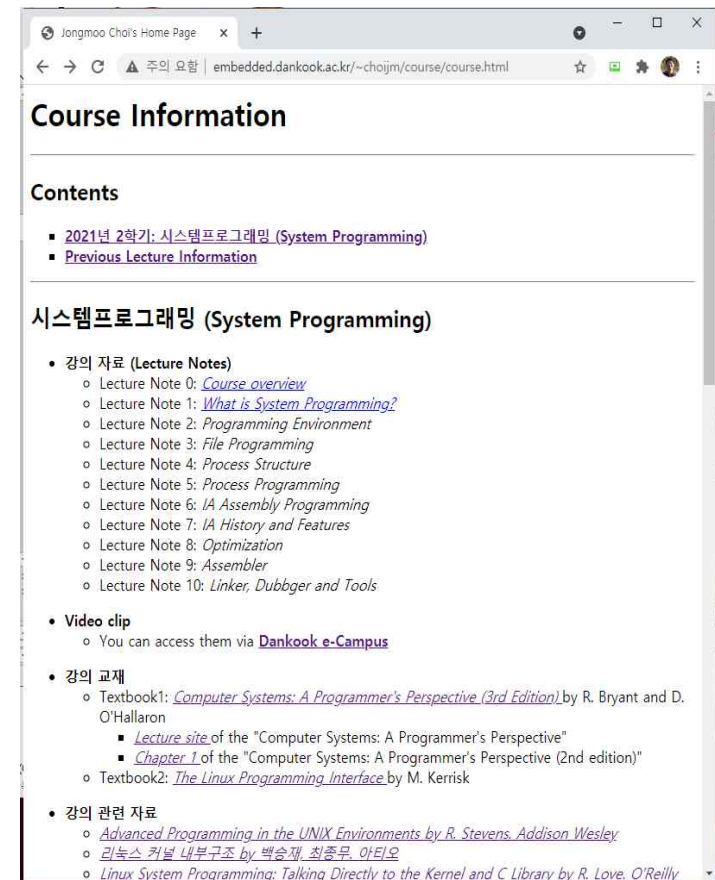
- Understand how software runs on hardware (or how software and hardware are connected)
  - ✓ High-level program for human vs. **Binary** for CPU
  - ✓ Compiler, Assembler, Linker, Loader, Debugger, **Library** (dll), ...
  - ✓ File system, **Device driver**
  - ✓ Concept of Process, **Scheduling** for multiple processes
  - ✓ Memory management (data/stack/heap, **virtual memory**)
  - ✓ Software-level **optimizations**: code motion, loop unrolling, ...
  - ✓ Hardware-level **optimizations**: pipeline, cache, ...
  - ✓ Recent technologies in Intel CPU
  
- Grasp the concept of **abstraction**
  - ✓ Information hiding
  - ✓ Interface vs. Implementation
  - ✓ Layered architecture



# Course Content

## ■ Lecture Notes

- ✓ LN0: Course Overview
- ✓ LN1: What is System Programming?
- ✓ LN2: Programming Environments
- ✓ LN3: File Programming
- ✓ LN4: Process Structure
- ✓ LN5: Process Programming
  
- ✓ LN6: IA Assembly Programming
- ✓ LN7: IA History and Features
- ✓ LN8: Optimization Practice
- ✓ LN9: Assembler
- ✓ LN10: Linker, Debugger and Tools
- ✓ LN11: IPC, Signal and Socket



(<http://embedded.dankook.ac.kr/~choijm/>)

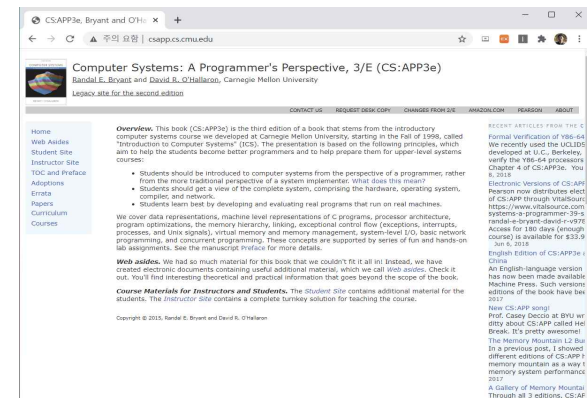
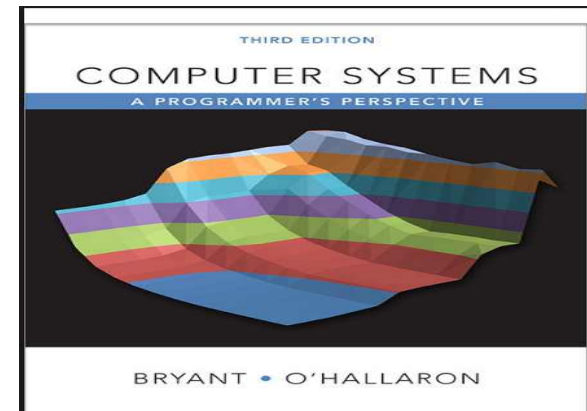


# Course Content

## ■ Textbook 1: CSAPP

- ✓ Computer Systems: A Programmer's Perspective, by R. Bryant and D. O'Hallaron
- ✓ Contents

1. A Tour of Computer Systems
2. Representing and Manipulating Information
3. Machine-level Representation of Programs
4. Processor Architecture
5. Optimizing Program Performance
6. The Memory Hierarchy
7. Linking
8. Exceptional Control Flow
9. Virtual Memory
10. System-Level I/O
11. Network Programming
12. Concurrent Programming



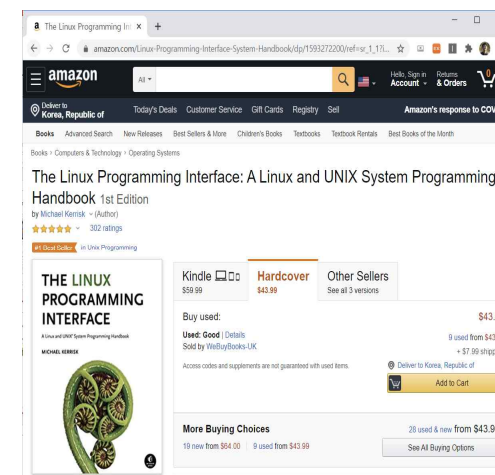
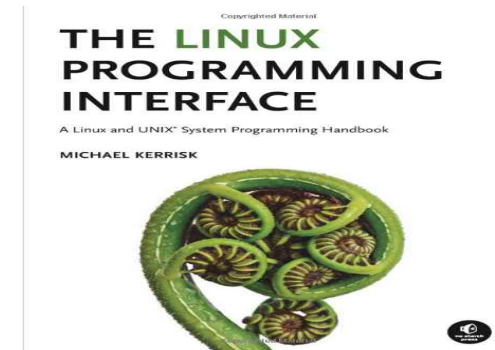
(<http://csapp.cs.cmu.edu/>)

# Course Content

## ■ Textbook 2: LPI

### ✓ The Linux Programming Interface: A Linux and UNIX System Programming Handbook

1. History and Standards
2. Fundamental Concepts
3. System programming concepts
4. File I/O: The Universal I/O Model
5. File I/O: Further Details
6. Process
7. Memory Allocation
8. Users and Groups
- ...
24. Process Creation
25. Process Termination
26. Monitoring Child Processes
27. Program Execution
- ... /\* total 64 chapters \*/



# Course Content

---

## ■ Relation btw Lecture Notes and Textbooks

- ✓ LN1. What is System Programming?
  - [CSAPP Chapter 1](#)
- ✓ LN2. Programming Environment
  - [LPI Chapter 1, 2, 3](#)
- ✓ LN3. File Programming
  - [LPI Chapter 4, 5](#) / CSAPP Chapter 10
- ✓ LN4. Process Structure
  - [LPI Chapter 6](#) / CSAPP Chapter 8, 9
- ✓ LN5. Process Programming
  - [LPI Chapter 24, 25, 27, 29](#) / CSAPP Chapter 8, 12
- ✓ LN6. IA assembly Programming
  - [CSAPP Chapter 2, 3](#) / Intel®64 & IA-32 Architectures Software Developer's Manual
- ✓ LN7. IA History and Features
  - [CSAPP Chapter 4](#) / Intel®64 & IA-32 Architectures Software Developer's Manual
- ✓ LN8. Optimization Practice
  - [CSAPP Chapter 5, 6](#) / LPI Chapter 23
- ✓ LN9. Assembler
  - [CSAPP Chapter 3, 7](#)
- ✓ LN10. Linker, Debugger and Tools
  - [CSAPP Chapter 7](#), <http://beej.us/guide/bggdb/>
- ✓ LN11. IPC, Signal and Socket
  - [LPI Chapter 43, 44, 45](#) / CSAPP Chapter 11





# How to Lecture?

## ■ Off-line case

✓ Mainly teaching (and Q&A)

✓ Using ppt from lecture site

(<http://embedded.dankook.ac.kr/~choijm/course/course.html>)

**Contents**

- 2021년 2학기: 시스템프로그래밍 (System Programming)
- Previous Lecture Information

**시스템프로그래밍 (System Programming)**

- 강의 자료 (Lecture Notes)
  - Lecture Note 0: [Course overview](#)
  - Lecture Note 1: [What is System Programming?](#)
  - Lecture Note 2: [Programming Environment](#)
  - Lecture Note 3: [File Programming](#)
  - Lecture Note 4: [Process Structure](#)
  - Lecture Note 5: [Process Programming](#)
  - Lecture Note 6: [IA Assembly Programming](#)
  - Lecture Note 7: [IA History and Features](#)
  - Lecture Note 8: [Optimization](#)
  - Lecture Note 9: [Assembler](#)
  - Lecture Note 10: [Linker, Debugger and Tools](#)
- Video clip
  - You can access them via [Dankook e-Campus](#)
- 강의 교재
  - Textbook1: [Computer Systems: A Programmer's Perspective \(3rd Edition\)](#) by R. Bryant and D. O'Hallaron
    - [Lecture site](#) of the "Computer Systems: A Programmer's Perspective"
    - [Chapter 1](#) of the "Computer Systems: A Programmer's Perspective (2nd edition)"
  - Textbook2: [The Linux Programming Interface](#) by M. Kerrisk
- 강의 관련 자료
  - [Advanced Programming in the UNIX Environments](#) by R. Stevens, Addison Wesley
  - [리눅스 커널 내부구조](#) by 백승재, 최준우, 아티오
  - [Linux System Programming: Talking Directly to the Kernel and C Library](#) by R. Love, O'Reilly
  - [유닉스/리눅스 프로그래밍 필수 유틸리티](#) by 백창우, 한빛미디어
  - [Intel 64 and IA-32 Architecture Software Developer's Manual](#)
  - [ARM System-on-Chip Architecture \(2nd Edition\)](#) by S. Furber
  - The UNIX time-sharing system: [UNIX paper](#)
  - [Inside of a Hard Disk](#)
  - [Concept of Pipeline](#)
  - [Memory Address](#)
  - [GNU GCC](#)
  - [GNU Assembler](#)
  - [GNU Debugger](#)
  - [Quick Guide to GDB](#)
  - [GNU Make](#)
  - [GNU libraries](#)

**The Linux Programming Interface: A Linux and UNIX System Programming Handbook, 1st Edition**  
by Michael Kerrisk (Author)  
302 ratings

Kindle \$19.99 | Hardcover \$42.99 | Other Sellers

Buy used: \$19.99 | Buy new: \$42.99

**Inside a Hard Disc Drive**  
YouTube

**Beej's Quick Guide to GDB**  
Release 2 (2009 Jun 14)

This is a very quick and dirty guide meant to get you started with the GNU Debugger, **gdb**, from the comfort of your terminal. **GDB** is run via an IDE, but many people out there start GDBs for a variety of reasons, and this tutorial is for you!

Again, this is only a getting-started guide. There's much much MUCH more to learn about what the debugger does than is written in these few short paragraphs. Check out your "man" pages or the online resources listed below for more info.

This tutorial is meant to be read in order, up to, but not including, the "Misc" section.

**Contents**

- Getting to use a debugger
- More information
- Starting **gdb** and getting to **main()**
- Breakpoints
- Stepping & running
- Examining variables
- Examining memory
  - Quick Manipulation
  - Jumping to an arbitrary section of code
  - Examining variables and values at runtime
  - Hardware Watchpoints
  - Attach to a running process
  - Using Corefiles for Postmortem Analysis
  - Window Managers
  - Display Registers and Assembly
  - Writing a Patch
- Quick Reference Cheat Sheet

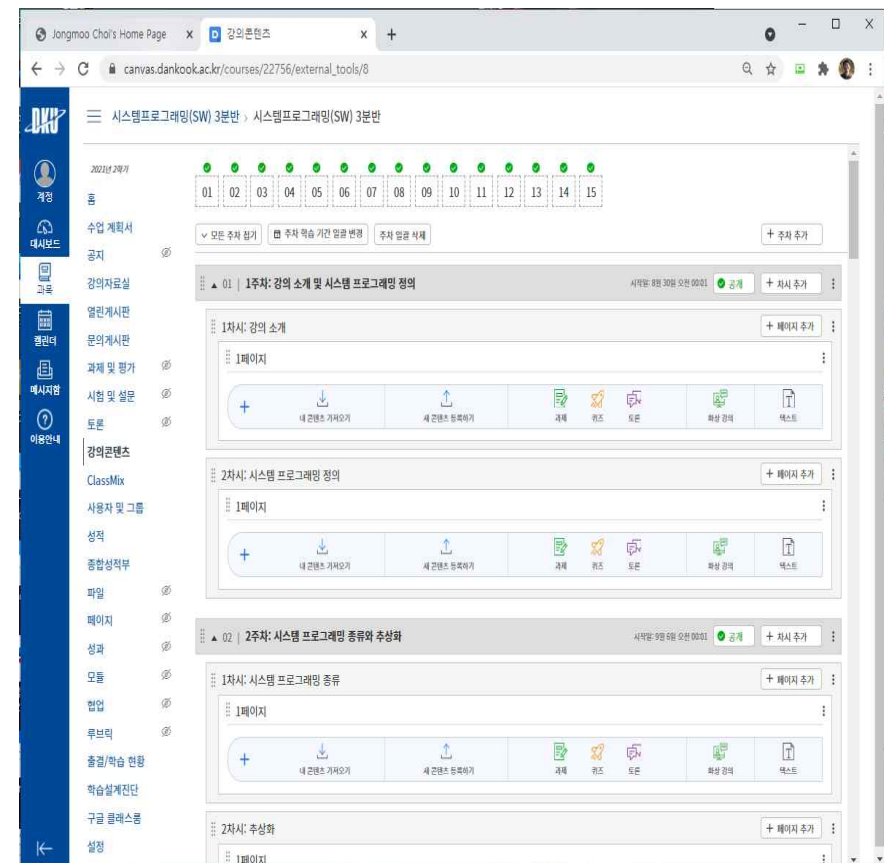
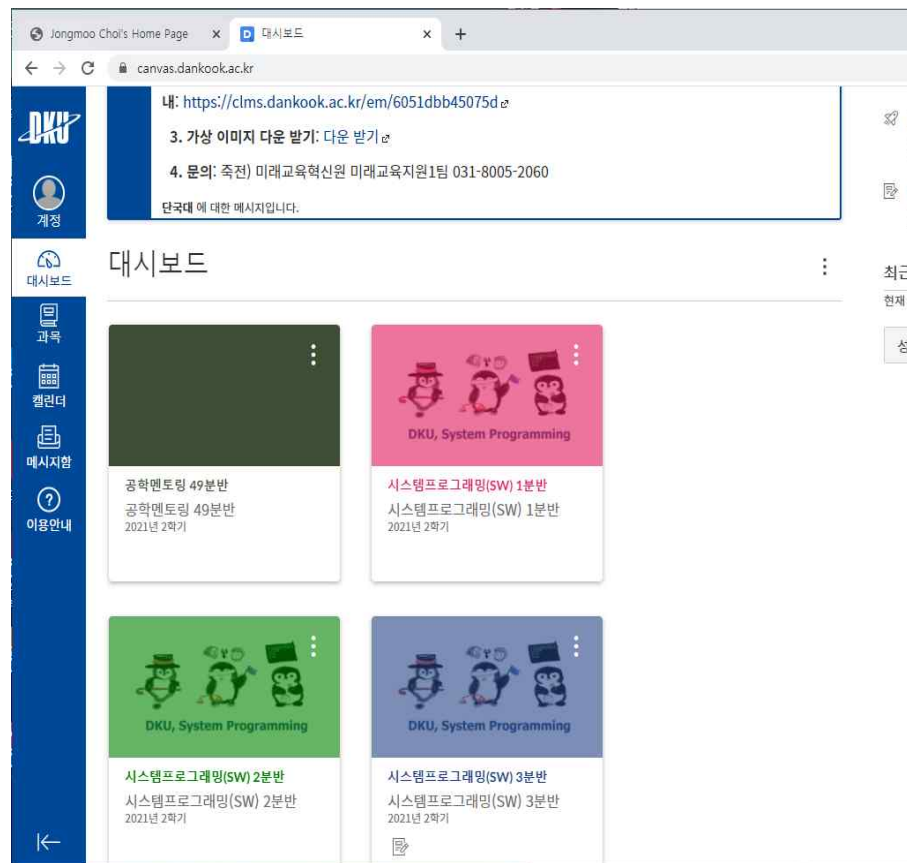
**Compiling**  
You have to tell your compiler to compile your code with symbolic debugging information included. Here's



# How to Lecture?

## ■ On-line case

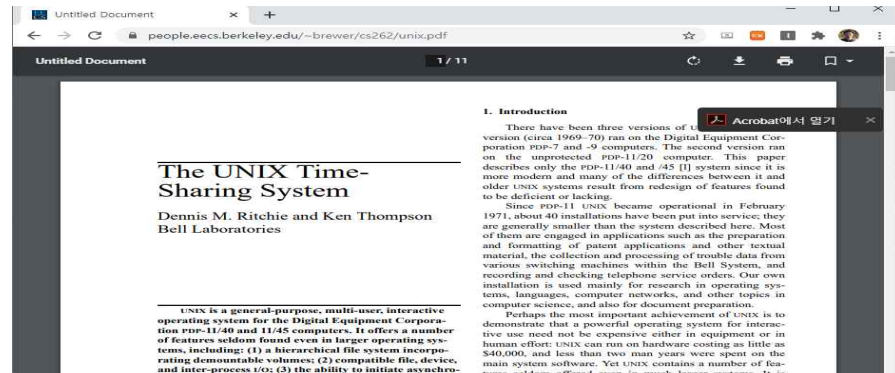
- ✓ Mainly based on lecture video (and Q&A)
- ✓ Using both ppt and video clip from Dankook LMS site  
(<https://nlms.dankook.ac.kr/>)



# How to Lecture?

## ■ Assignment

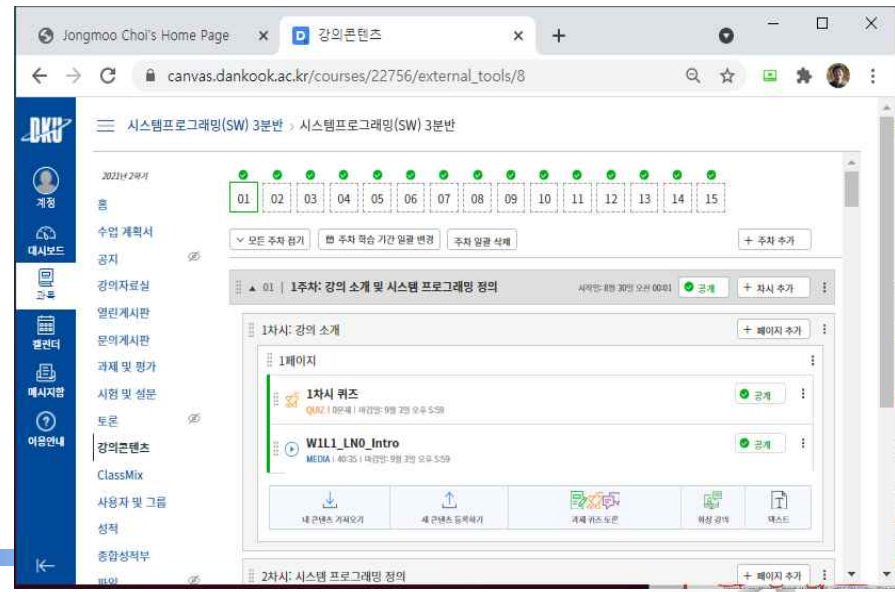
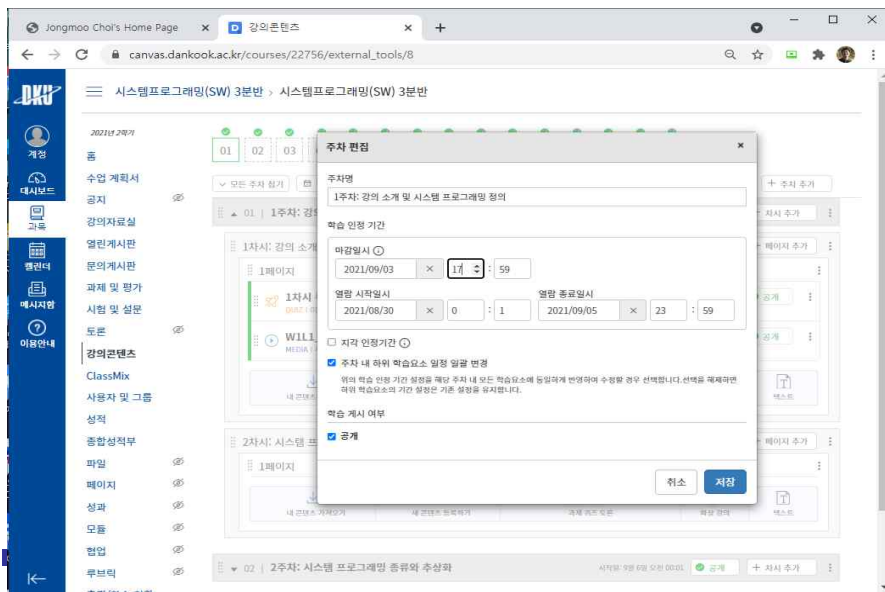
- ✓ Programming Project (3 or 4): Personal (or Team)
  - Make programs in [Linux Environment!!](#)
    - Linux Server: 220.149.236.2 (primary), 220.149.236.4 (secondary)
    - TA: Jeyeun Lee (Room 515, SW-ICT Bldg)
  - Program examples
    - Using vi editor, file I/O, process manipulation, shell, assembly, optimization, ...
- ✓ Documentation project (1 or 2): Personal
  - Reading a chapter in our textbooks
    - E.g. Chapter 1 in CSAPP or Chapter 3 in LPI
  - Reading a well-known paper
    - E.g. UNIX paper



# How to Study?

## ■ 4 steps

- ✓ 1) read ppt first, 2) watch video clips, 3) reply quiz per each video clip, 4) read related chapters in textbooks (CSAPP and LPI)
- ✓ Two video clips (lesson 1 and lesson 2) per each week
  - Upload every Monday, Watch at least 95% until **6 PM Friday**. (Can not access after Sunday)
- ✓ Quiz per each video clip
  - Reply your answer until **6 PM Friday** via **LMS** (after this time, you can not get a score from this quiz.)



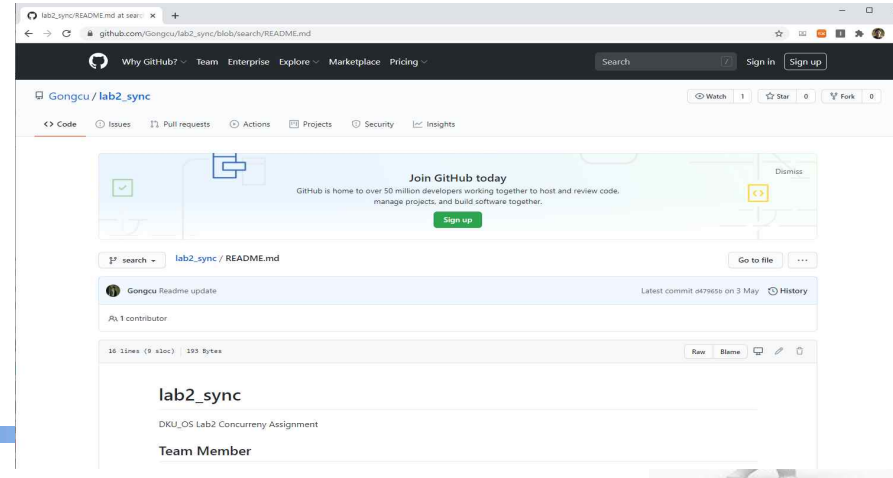
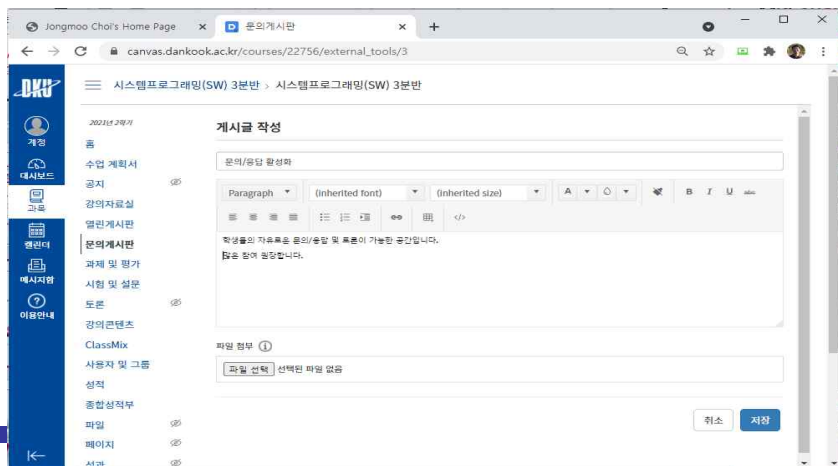
# How to Study?

## ■ Evaluation

- ✓ Mid exam.(25%), Final exam.(25%)
- ✓ Assignment (25%), Quiz/Attendance (25%)
- ✓ Can be changed according to status of COVID-19
  - E.g. If you can not take an off-line mid-exam, the percentage can be changed

## ■ Comments for on-line lecturing

- ✓ Quiz at each video clip is quite important
- ✓ Online Q&A and Assignments becomes more critical
  - Activity will be considered for your grade.



# Discussion

---

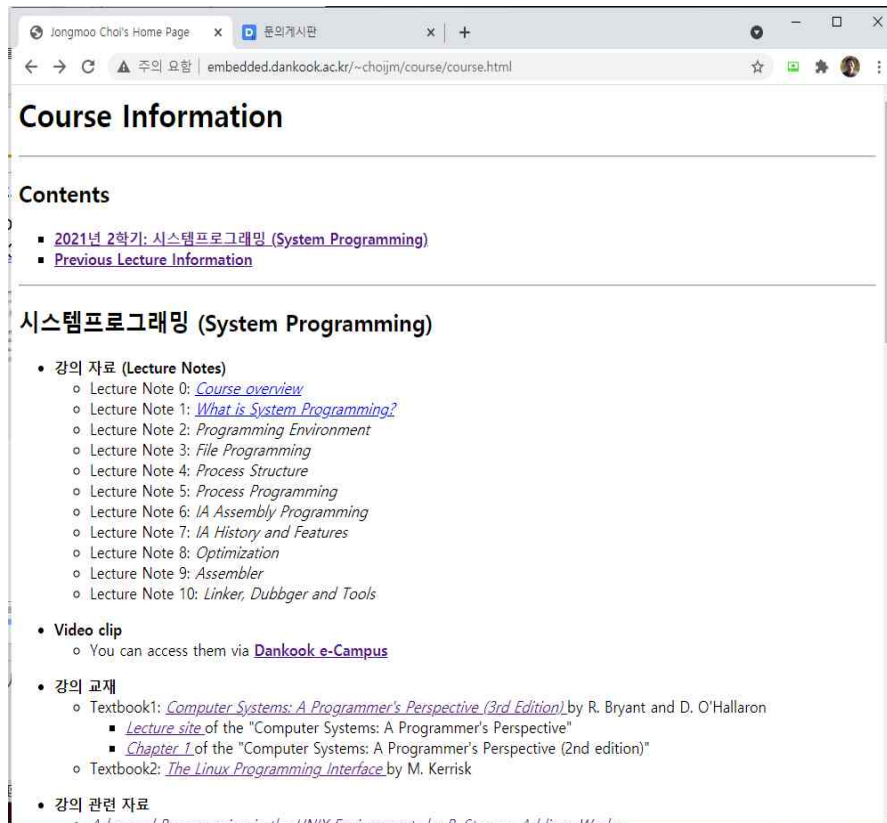
- Q&A
  - ✓ Using bulletin board at LMS site
  - ✓ Using email: [choijm@dankook.ac.kr](mailto:choijm@dankook.ac.kr)



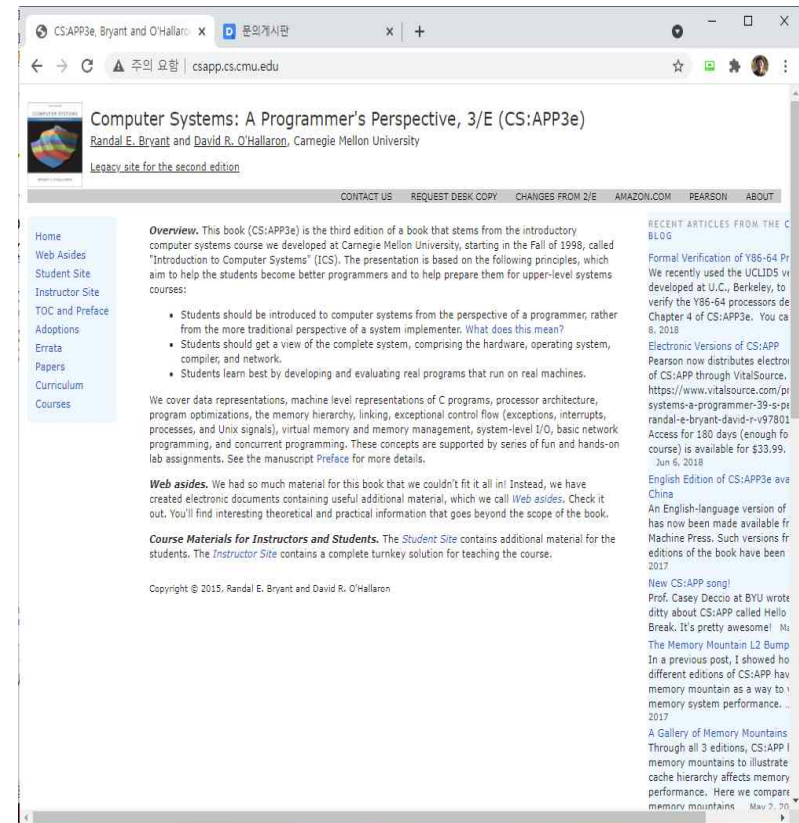


# Quiz for 1st-Week 1st-Lesson

- Find the Dankook University in CSAPP lecture site
  - ✓ Reply where you can find the Dankook University in this site?
  - ✓ Due: until 6 PM Friday of this week(3<sup>rd</sup>, September)



(<http://embedded.dankook.ac.kr/~choijm/course/course.html/>)



(<http://csapp.cs.cmu.edu/>)





## Appendix: Good book for Learning Linux

### ■ Linux Kernel Internal (리눅스 커널 내부 구조)

- ✓ 0장. 운영체제 이야기
- ✓ 1장. 리눅스 소개
- ✓ 2장. 리눅스 커널 구조
- ✓ 3장. 태스크 관리
- ✓ 4장. 메모리 관리
- ✓ 5장. 파일 시스템과 가상 파일 시스템
- ✓ 6장. 인터럽트와 트랩 그리고 시스템 호출
- ✓ 7장. 리눅스 모듈 프로그래밍
- ✓ 8장. 디바이스 드라이버
- ✓ 9장. 네트워킹
- ✓ 10장. 운영체제 관련 실습
- ✓ 부록 A. 리눅스와 가상화 그리고 XEN
- ✓ 부록 B. MTD와 YAFFS
- ✓ 부록 C: Map of the Linux

