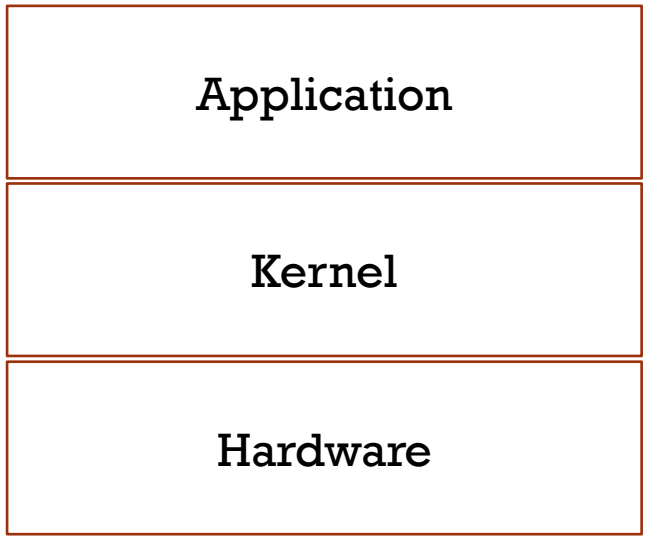
OPERATION SYSTEM INTRODUCTION

Ref: 6.S081, MIT

PREREQUISITE

- Computer architecture (computer organization, assembly language)
- C programming (C pointer, array, struct, stack, heap)
- Data structure and algorithm (linked list, FIFO)
- Microcontroller (UART, timer, device driver)

ORGANIZATION



- User applications: vi, gcc, DB, &c
- Kernel services
- H/W: CPU, RAM, disk, net, &c

we care a lot about the interfaces and internal kernel structure

INTRODUCTION

What is the purpose of an O/S?

- Abstract the hardware for convenience and portability
- Multiplex the hardware among many applications
- Isolate applications in order to contain bugs
- Allow sharing among cooperating applications
- Control sharing for security
- Don't get in the way of high performance
- Support a wide range of applications

SERVICES

What services does an O/S kernel typically provide?

- process (a running program)
- memory allocation
- file contents
- file names, directories
- access control (security)
- many others: users, IPC, network, time, terminals

SYSTEM CALL

- System call is the application / kernel interface
- Examples, in C, from UNIX (e.g. Linux, macOS, FreeBSD)

```
fd = open("out", 1);
write(fd, "hello\n", 6);
pid = fork();
```

- These look like function calls but they aren't

LINUX

Learning O/S user space:

- Environment (CLI)
- Abtraction concepts
- Bash shell and its feature
- C program and C library
- Multithreading program

XV6

Learning O/S kernel space by doing XV6, a multi-processor operating system for RISC-V:

- Reading XV6 textbook & source code
- Experience and extending XV6 by doing exercises

Why XV6?

- It's big enough to illustrate the basic design and implementation ideas
- It's far smaller than modern O/S, and correspondingly easier to understand
- It has a similar structure to many modern O/S; such as Linux kernel

THE GOALS

- Understand user space program
- Understand operating system (O/S) design and implementation
- Hands-on experience extending a small O/S
- Hands-on experience writing systems software

WHY SHOULD YOU TAKE THIS CLASS?

O/S design+implementation is hard and interesting because:

- unforgiving environment: quirky h/w, hard to debug
- many design tensions:
  - efficient vs abstract/portable/general-purpose
  - powerful vs simple interfaces
  - flexible vs secure
- features interact: `fd = open(); fork()`
- uses are varied: laptops, smart-phones, cloud, virtual machines, embedded
- evolving hardware: NVRAM, multi-core, fast networks

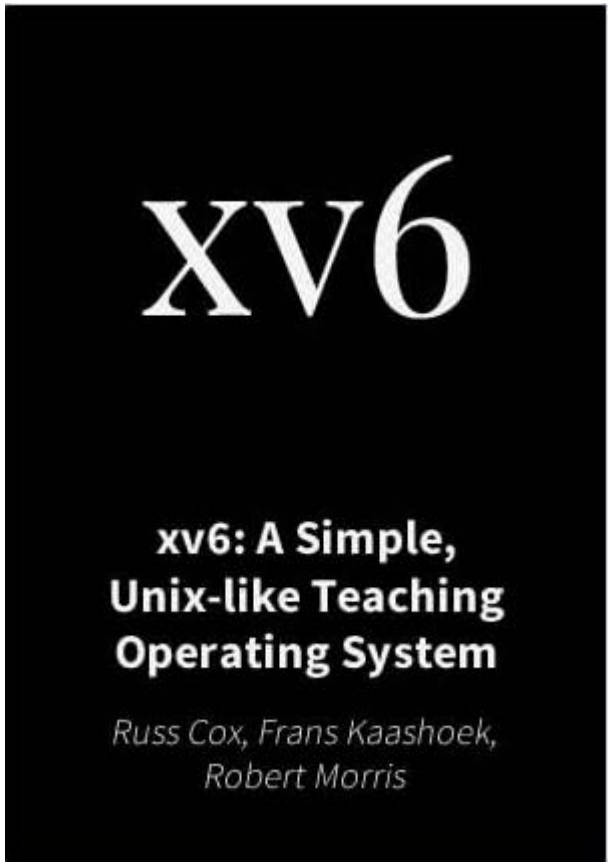
WHY SHOULD YOU TAKE THIS CLASS?

You'll be glad you took this course if you…

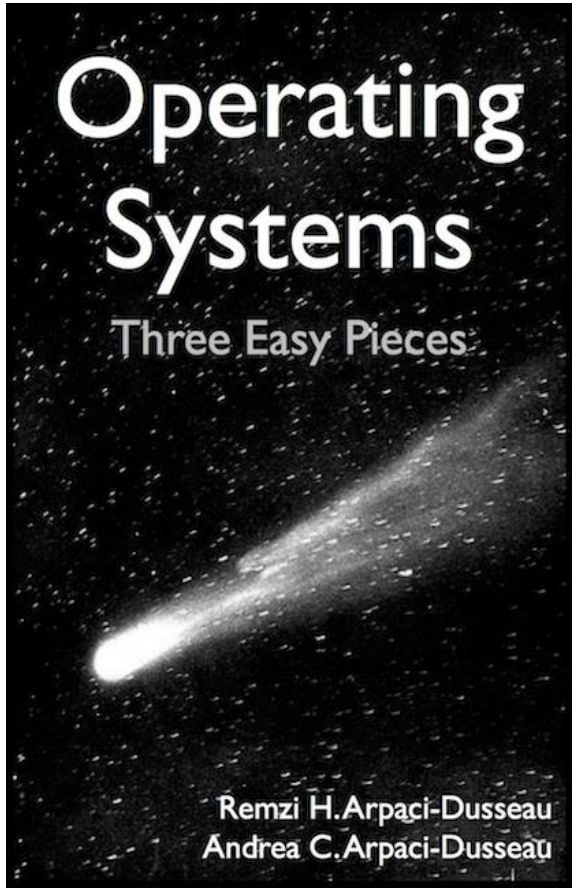
- care about what goes on under the hood
- like infrastructure
- need to track down bugs or security problems
- care about high performance

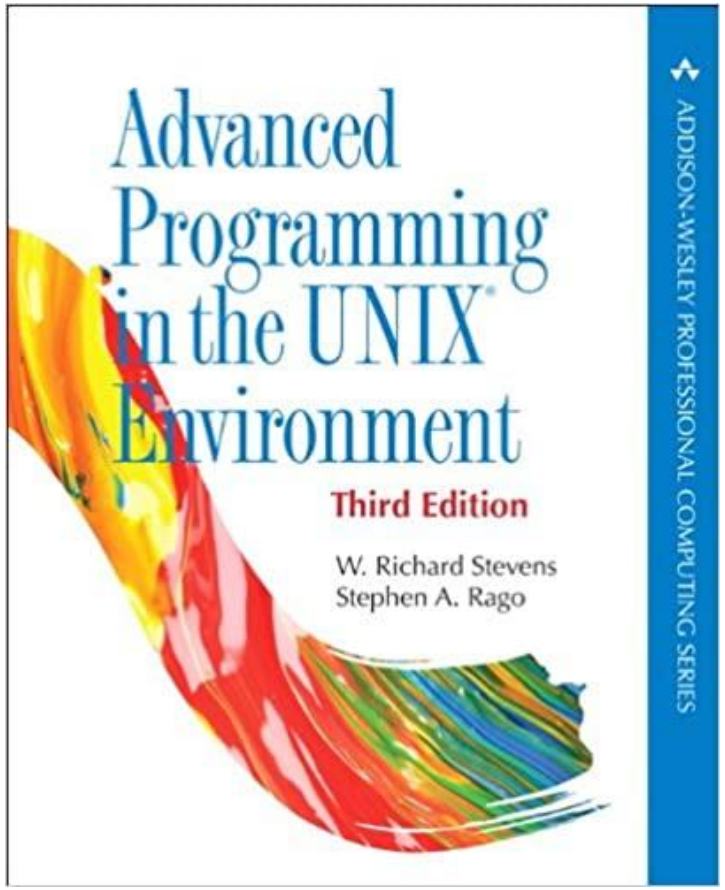
CLASS LOGISTICS

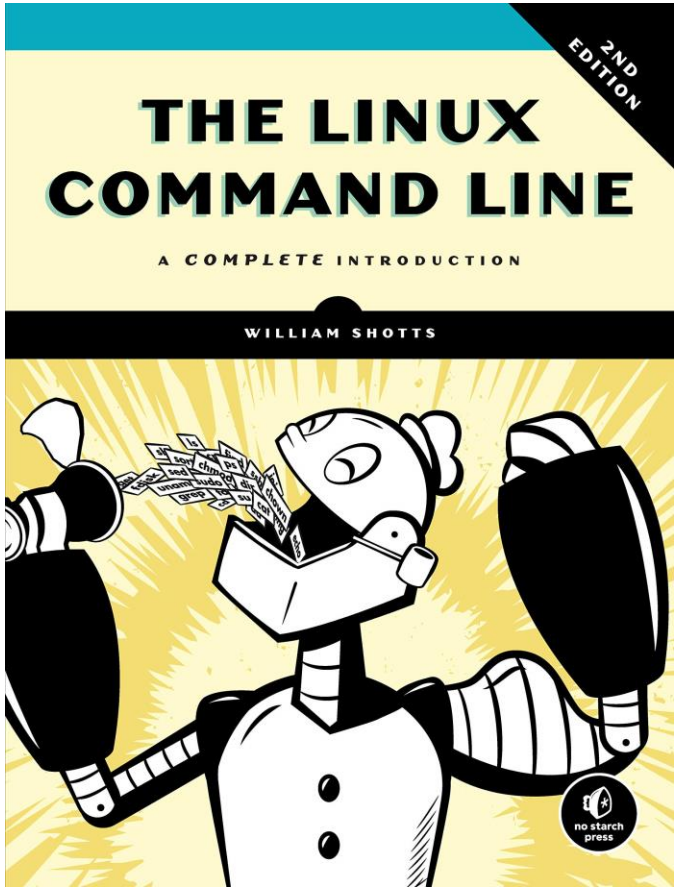
TEXTBOOK


xv6
xv6: A Simple, Unix-like Teaching Operating System
Russ Cox, Frans Kaashoek, Robert Morris

REFERENCE


Operating Systems
Three Easy Pieces
Remzi H.Arpaci-Dusseau
Andrea C.Arpaci-Dusseau


Advanced Programming in the UNIX Environment
Third Edition
W. Richard Stevens
Stephen A. Rago
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


2ND EDITION
THE LINUX COMMAND LINE
A COMPLETE INTRODUCTION
WILLIAM SHOTTS
no starch press

LECTURER

- **Name:** Đỗ Quốc Minh Đăng
- Email: dqmdang@hcmus.edu.vn