Pintos Overview

—— Why, What and How

TA Session

PKUOS

Welcome to the World of Operating System

TA:	xiangyuxing 向昱行
Email:	echostone@stu.pku.edu.cn
GitHub:	EchoStone1101

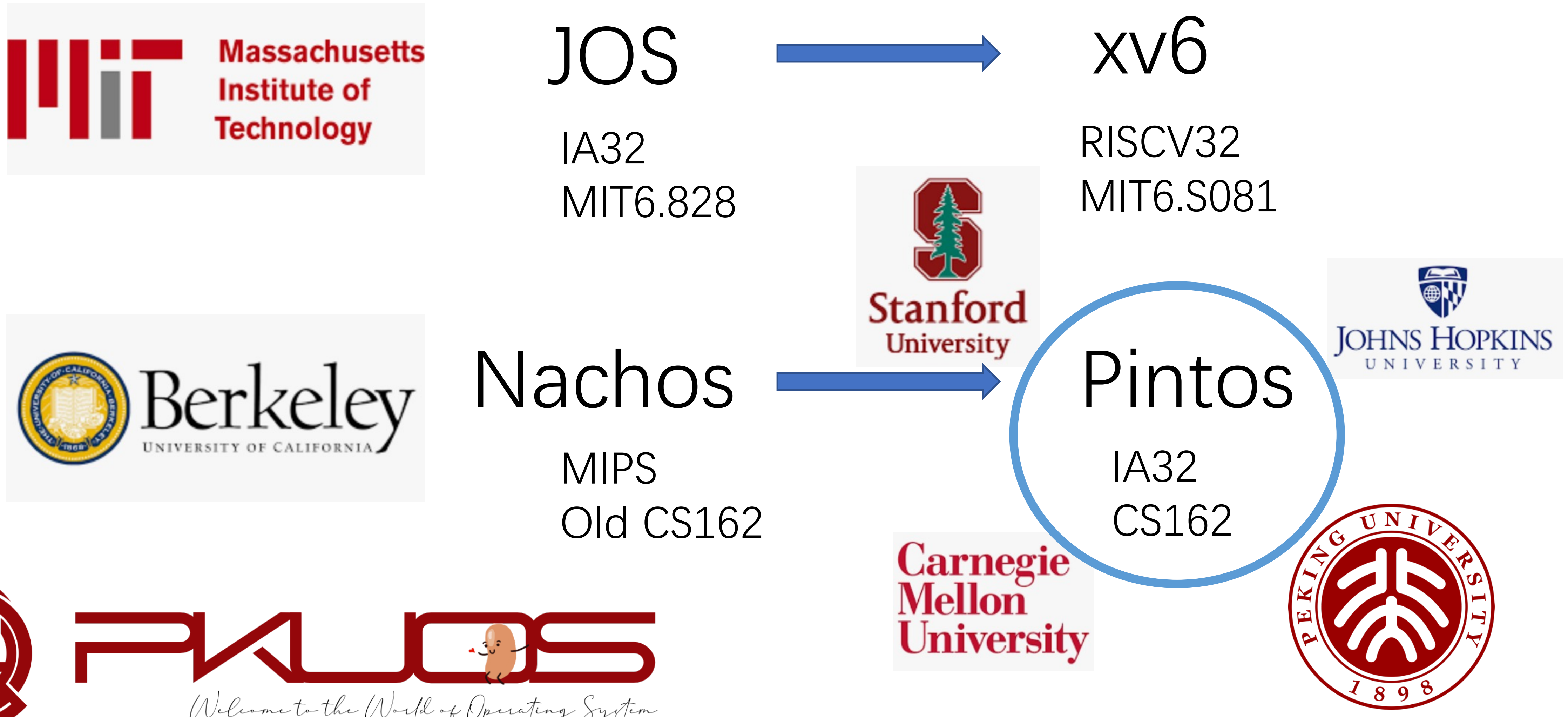
Some announcements:

Getting right started…

- Lab 0 is released this Tuesday
- Lab 0 **Code** will be due next Thursday 11:59 pm
- Lab 0 **Design Doc** will due next Sunday 11:59 pm

The rest Labs have similar deadlines…

Educational OS Project Zoo


Massachusetts Institute of Technology
Berkeley UNIVERSITY OF CALIFORNIA
Stanford University
JOHNS HOPKINS UNIVERSITY
Carnegie Mellon University
PEKING UNIVERSITY 1898

JOS

IA32
MIT6.828

xv6

RISCV32
MIT6.S081

Nachos

MIPS
Old CS162

Pintos

IA32
CS162

PKUOS

Welcome to the World of Operating System

Q : Why Pintos ?

Design and Implementation

- OSDI, NSDI, PLDI … …
- Your design matters !!

- Talk is cheap, show me the code
- Write 2000+ LOC in a 10000+ LOC codebase

Q : Why Pintos ?

You will learn by Read The Code

- important skill both in production and research
- learn from good coding style
- some tools may help you

Q : Why Pintos ?

You will learn by **Design The Code**

- ➢ think tenth, code once
- ➢ design doc template may help you
- ➢ not Pintos, but **Your Pintos**

Q : Why Pintos ?

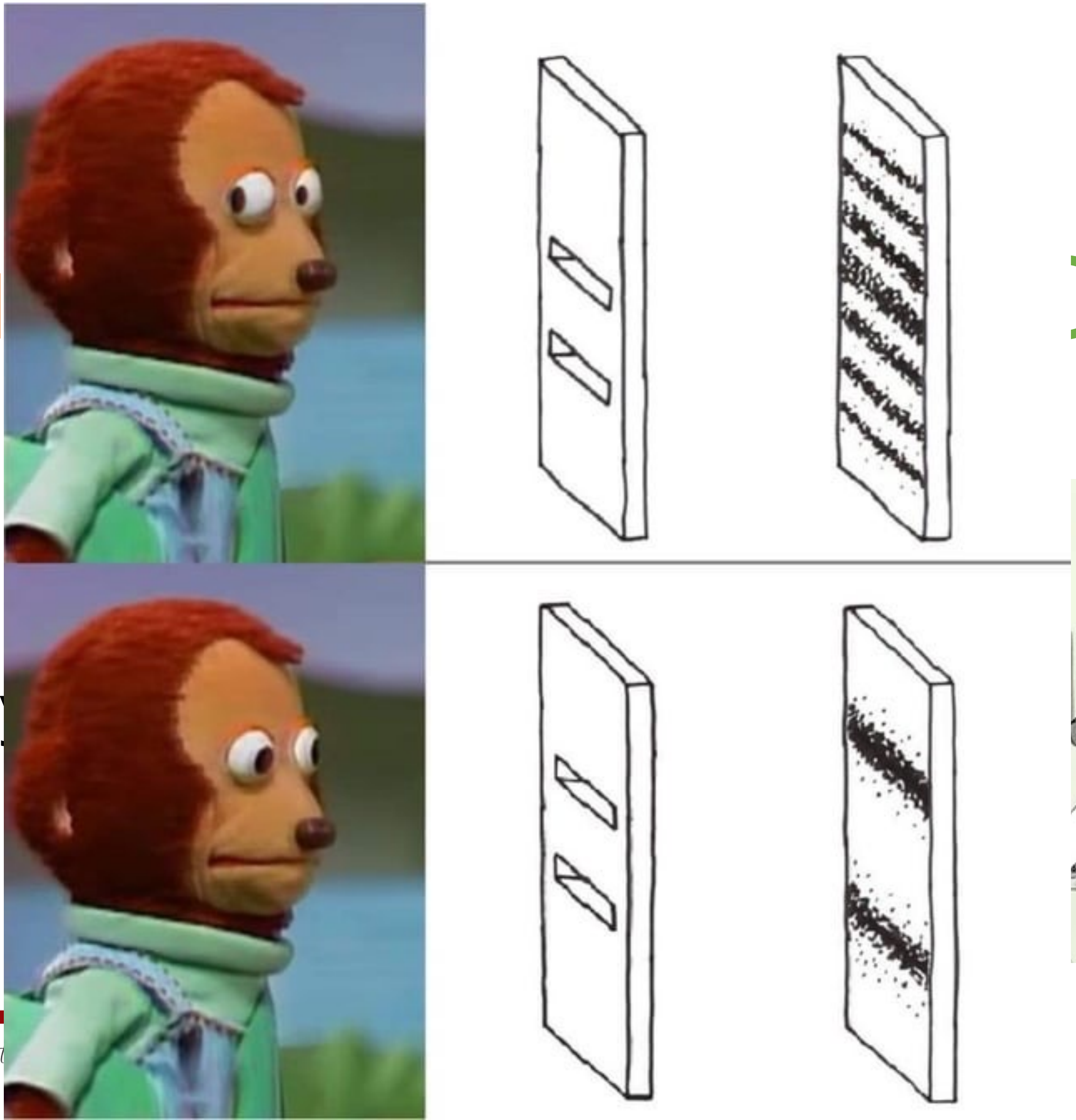
You will learn by Write The Code

- maybe your first time writing 2000+ LOC
- tricky multi-threading synchronization
- test-driven development

Q : Why

You wil

ode


Me, debugging

- You will
- start earl

Q : Why not Pintos ?

- ➢ IA32 architecture : CISC ISA, historical legacy

Pintos → Tacos

RUST OS NEVER GETS RUSTY.

Pintos reimplemented in Rust based on RISCV64.

Q : Why **not** Pintos ?

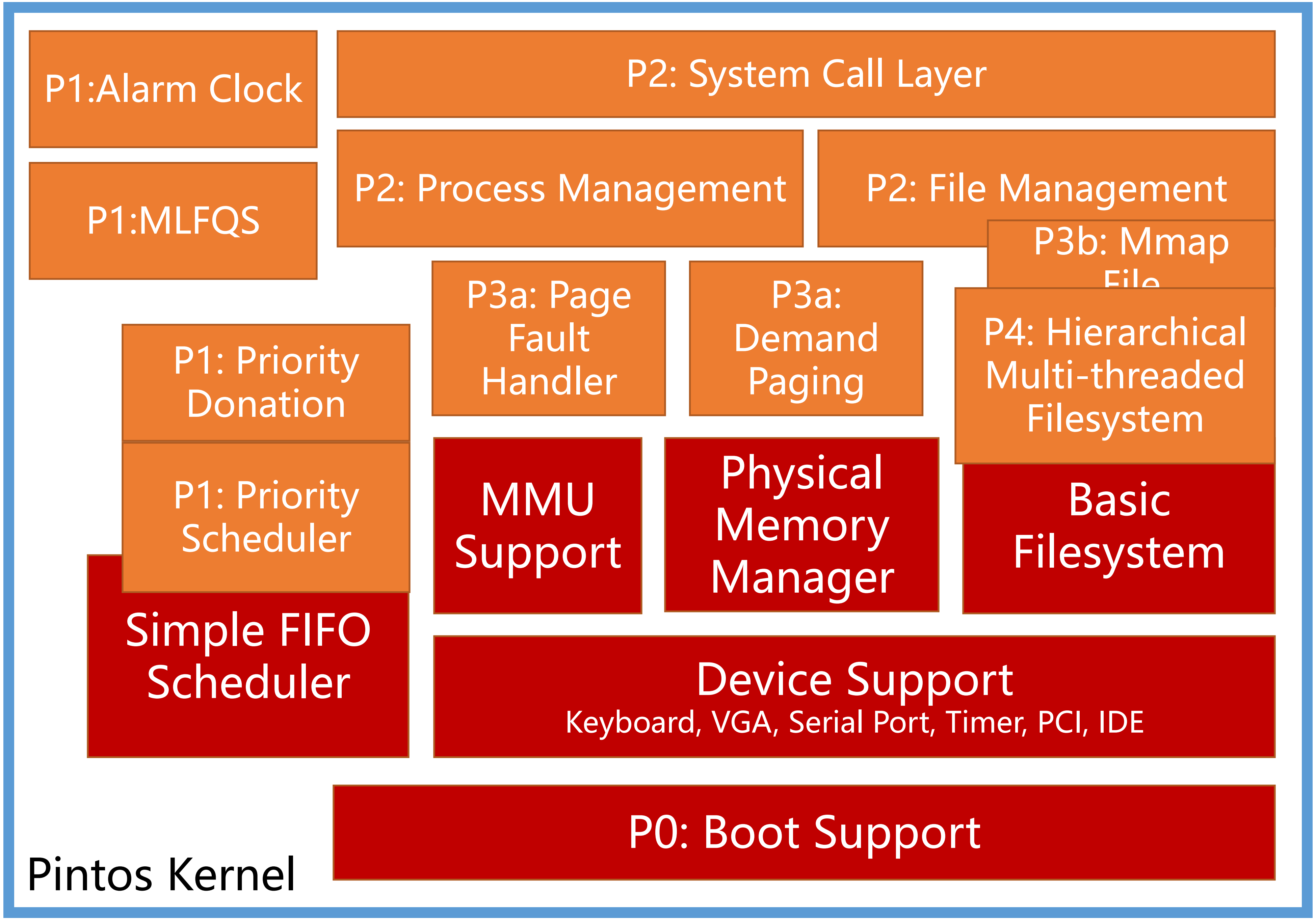
- IA32 architecture : CISC ISA, historical legacy

Pintos → Tacos

- time consuming : 100 hours +++

optional lab4, long long long lab document, per-lab TA session

Q : So … what will you do?



P1:Alarm Clock

P2: System Call Layer

P1:MLFQS

P2: Process Management

P2: File Management

P3b: Mmap File

P3a: Page Fault Handler

P3a: Demand Paging

P4: Hierarchical Multi-threaded Filesystem

P1: Priority Donation

P1: Priority Scheduler

MMU Support

Physical Memory Manager

Basic Filesystem

Simple FIFO Scheduler

Device Support

Keyboard, VGA, Serial Port, Timer, PCI, IDE

P0: Boot Support

Pintos Kernel

P0: Getting Real

P1: Threading

P2: User Programs

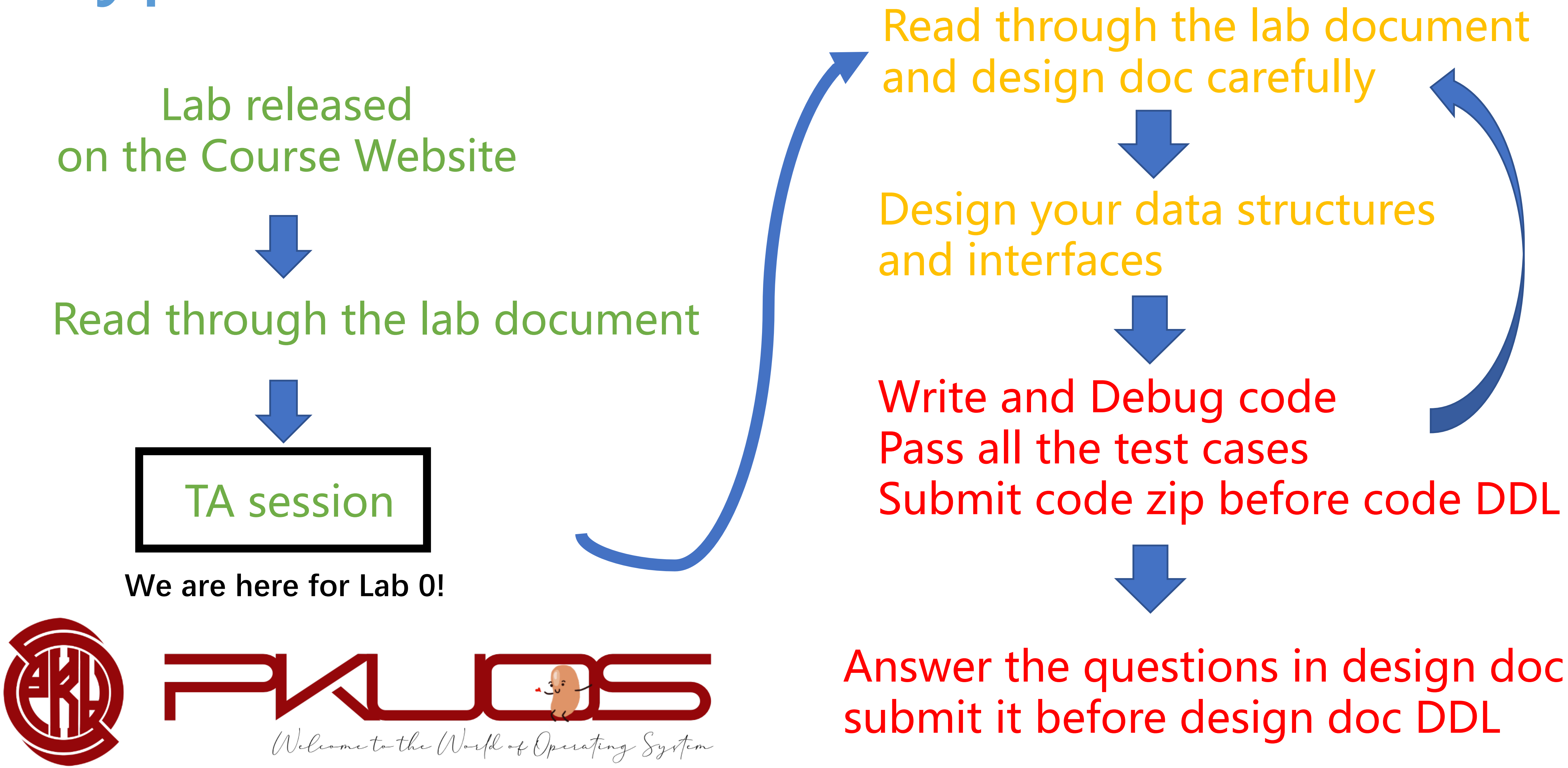
P3: Virtual Memory

P4: File System

🟧 Students Create

🟥 Support Code

Typical workflow:



Lab released
on the Course Website

Read through the lab document

TA session

We are here for Lab 0!

Read through the lab document
and design doc carefully

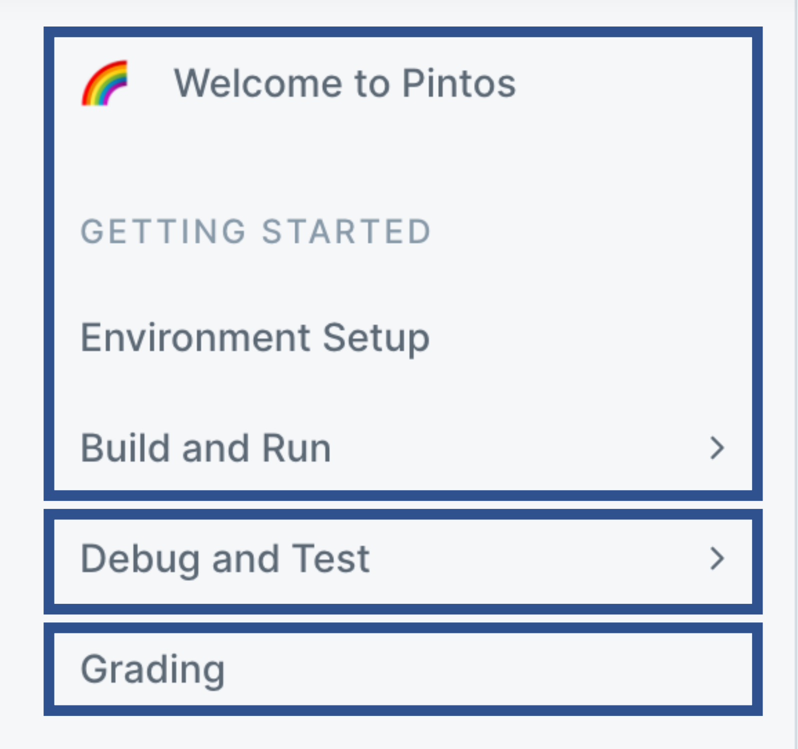
Design your data structures
and interfaces

Write and Debug code
Pass all the test cases
Submit code zip before code DDL

Answer the questions in design doc
submit it before design doc DDL

Q :How to survive?

PintosBook long, but helpful



🌈 Welcome to Pintos

GETTING STARTED

Environment Setup

Build and Run >

Debug and Test >

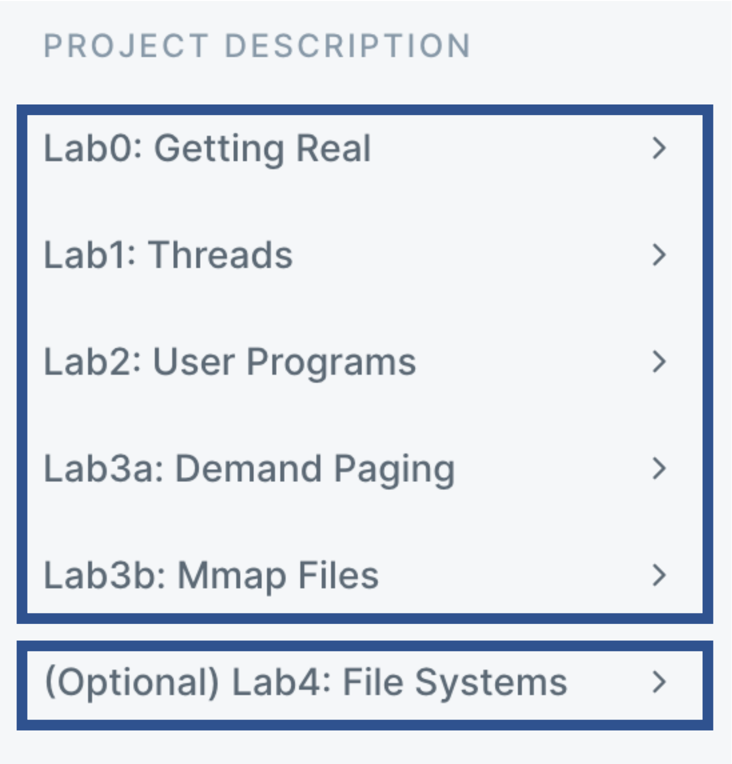
Grading

Set up your local development environment.

Look through it and look back if needed.

Important, read it carefully.

Q :How to survive? PintosBook



PROJECT DESCRIPTION

- Lab0: Getting Real
- Lab1: Threads
- Lab2: User Programs
- Lab3a: Demand Paging
- Lab3b: Mmap Files
- (Optional) Lab4: File Systems

Look through it before each TA Session.

Read it carefully during implementation.

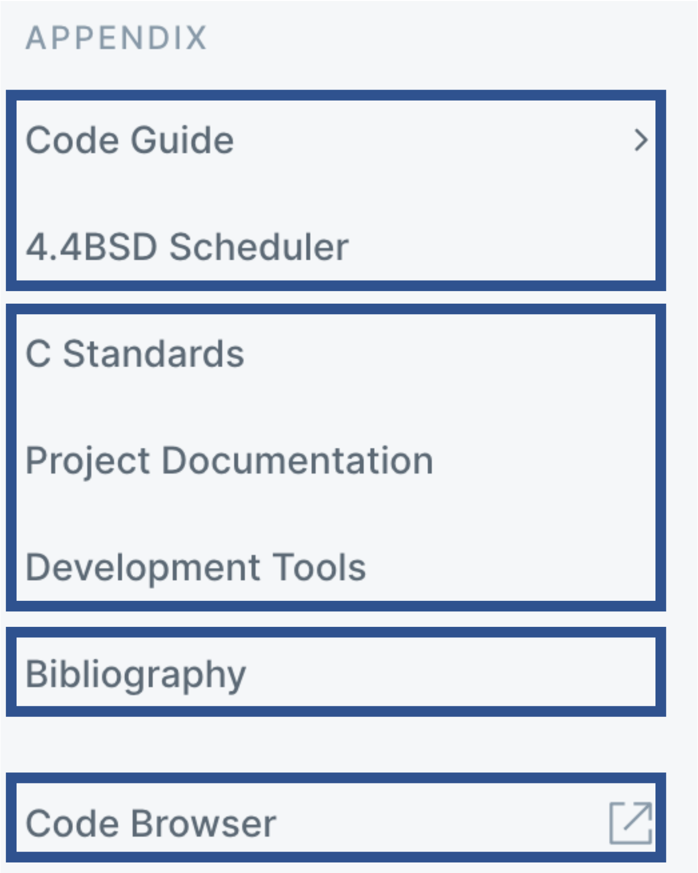
Optional but rewarding Lab4.

PKUOS

Welcome to the World of Operating System

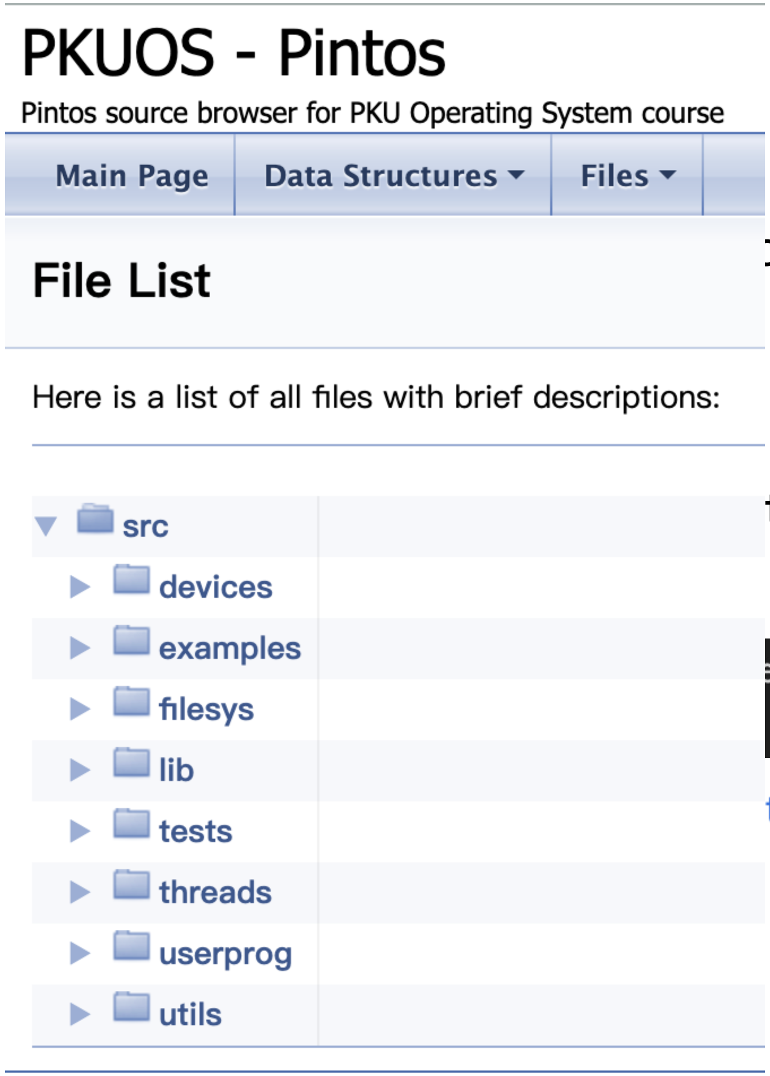
Q :How to survive?

PintosBook



APPENDIX

- Code Guide ›
- 4.4BSD Scheduler
- C Standards
- Project Documentation
- Development Tools
- Bibliography
- Code Browser



PKUOS - Pintos

Pintos source browser for PKU Operating System course

File List

Here is a list of all files with brief descriptions:

- ▼ src
  - ▶ devices
  - ▶ examples
  - ▶ filesys
  - ▶ lib
  - ▶ tests
  - ▶ threads
  - ▶ userprog
  - ▶ utils

cts going.

ters.

```
e [IntrList].
```

t List, 2000.

PKUOS

Welcome to the World of Operating System

Q :How to survive? Your kind TA

Learn to ask questions.

Do not be shy, ask in class, in office hour or in the Piazza.

Office hour: Friday 10:00-11:00 a.m., Yan Yuan Building 818
Piazza: https://piazza.com/pku.edu.cn/spring2025/04834490
(see course page)

But … … your TAs are not your ~~personal assistants (or Mr.Deepseek)~~.

- “My program crashed.”
- “What does this error mean?”
- “I failed xxx testcase.”
- “My computer can not boot.”

Think twice, Ask once.

- How to ask questions the smart way.

- RTFM

- STFW

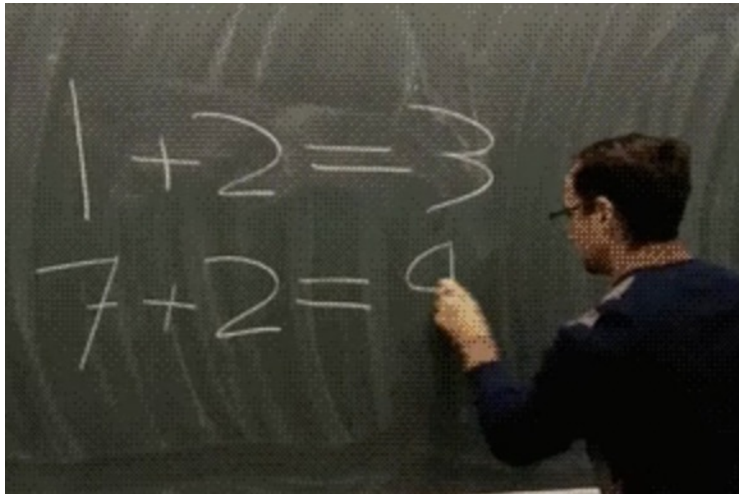
Think twice, Ask once.

- "I encounter xxx under xxx condition."
- "Google says xxx, StackOverflow says xxx, Document says xxx, but yyy."
- "Hey, TA, I found xxx and I think you do not know about it !"

Q :How to survive?

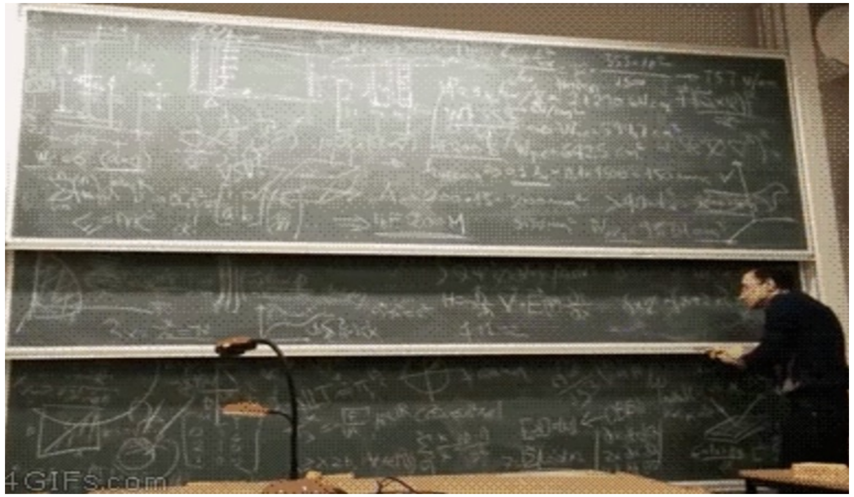
Good habits awkward, but helpful

Use Version Control tool —— Git



Newly written code

A week later



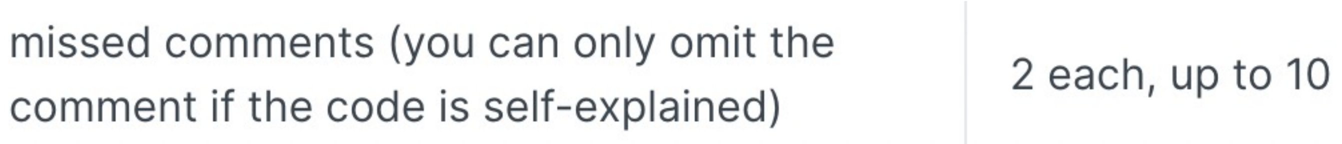
The same code

[How to write good commit message.]

Q :How to survive? Good habits

Write concise but good comments.

- ➢ Summarize the function in one sentence first.
- ➢ Pre-condition: input constraints (You may ASSERT these constraints)
- ➢ Post-condition: return value, exception (kernel panic)



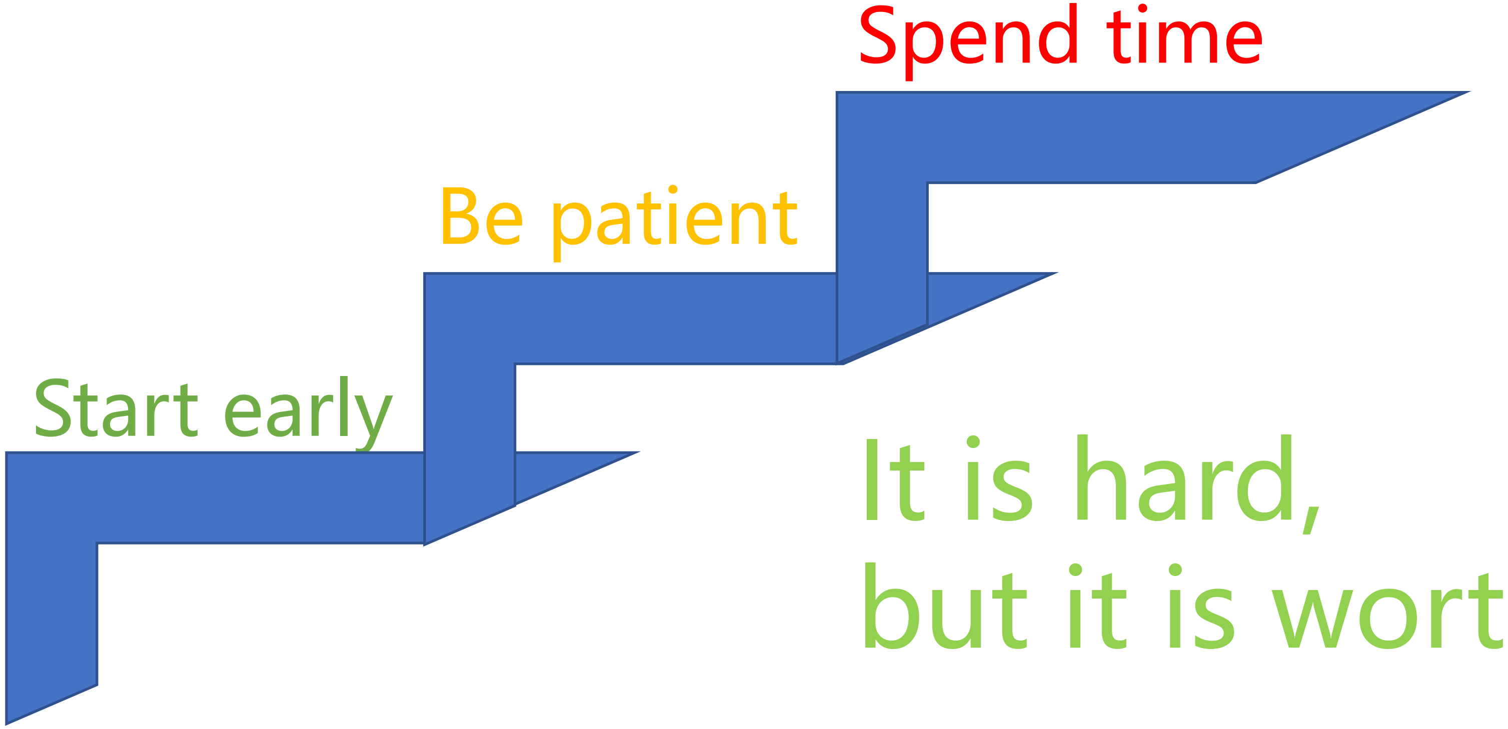
missed comments (you can only omit the comment if the code is self-explained)	2 each, up to 10

Q :How to survive? Good habits

Module and Abstraction.

➢ A function should (only) do one thing	clean
➢ A function more than 100 LOC	warning
➢ A function more than 200 LOC	Something may go wrong

Q :How to survive?



Start early

Be patient

Spend time

It is hard,
but it is worth it.

Lab0 FAQs

In this Lab, you will be...

- Walking through the **booting** of Pintos

- Try your hands on **debugging** Pintos

- Write your **first line of code** in Pintos: a tiny shell

OS Booting sounds overwhelming?

- All essential information are provided in the PintosBook
- You don't need to master all details; you practice how you learn from new information!

Cutting through the **confusing** jargons...

Docker ENV

You will (assuming you use Docker for running Pintos):

- develop Pintos in a ubuntu:18.04 container (regardless of your host)
- cross-compile Pintos into i386 (Intel 80386) binaries (IA32, 4GB)
  This architecture is hopefully familiar - it was discussed in ICS.
- execute Pintos with either QEMU or Bochs, kernel emumators
  - ...which come with the BIOS (Basic Input/Output System) firmware that loads a custom OS **bootloader**

Pintos

loader.S

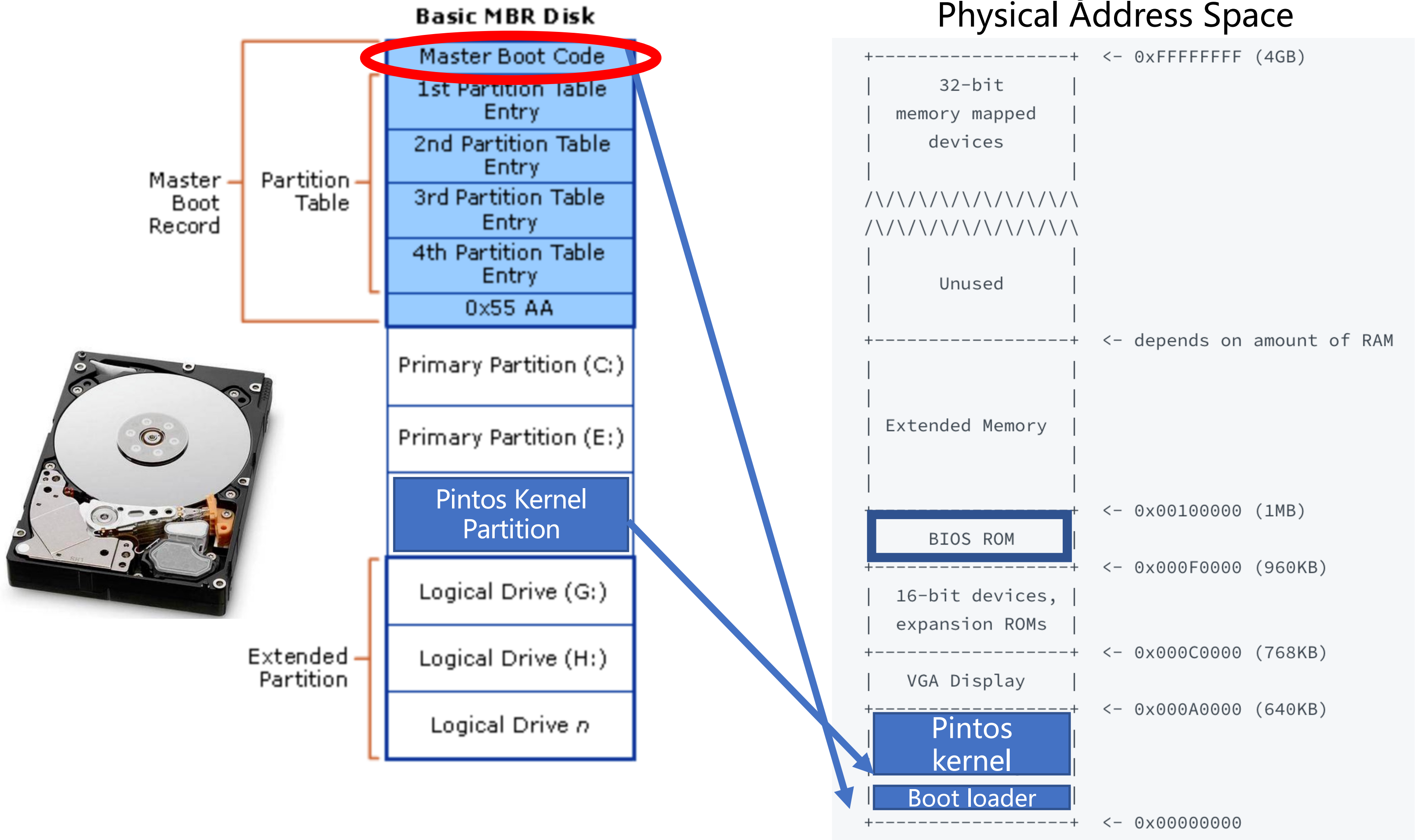
- ...which, as part of Pintos, locates and loads the actual **kernel**

start.S

- ...which is nothing but an i386 ELF executable, with .text, .data, and an entry point
  - ...which switches from real mode to protected mode, and calls pintos_init()

Booting Pintos

This MBR code is usually referred to as a boot loader.



```
+------------------+  <- 0xFFFFFFFF (4GB)
|      32-bit      |
|  memory mapped   |
|     devices      |
|                  |
/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\
|                  |
|      Unused      |
|                  |
+------------------+  <- depends on amount of RAM
|                  |
|                  |
| Extended Memory  |
|                  |
|                  |
+------------------+  <- 0x00100000 (1MB)
|     BIOS ROM     |
+------------------+  <- 0x000F0000 (960KB)
|  16-bit devices, |
|  expansion ROMs  |
+------------------+  <- 0x000C0000 (768KB)
|   VGA Display    |
+------------------+  <- 0x000A0000 (640KB)
|                  |
|                  |
|                  |
+------------------+  <- 0x00000000
```

Hard-wired by the hardware

The real-world booting process can be much more complicated

GRUB (GRand Unified Bootloader),
UEFI (Unified Extensible Firmware Interface), ...

X86 Mode (history legacy)

X86 Real Mode → Enabled in start.S → X86 Protected Mode

X86 Real Mode

- 16-bit Instructions and Registers

  AX, BX, CX, DX, SI, DI, BP, SP

- 20-bit Memory Address Space (Up to 1MB)

  16-bit segment registers

  CS, DS, SS, ES, FS, GS

  PAddr = SEG << 4 + Operand

X86 Protected Mode

- 32-bit Instructions and Registers

  EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP

- 32-bit Memory Address Space (Up to 4GB)

  Reserved segment registers, but for protection

  Address translation enabled

And if you really want to know every detail…

- Dig through Makefiles for how the Bootloader (loader.S) and the kernel itself is linked and run

- src/utils/pintos is how you will be running Pintos, which is actually a Perl script that you can try to read and understand

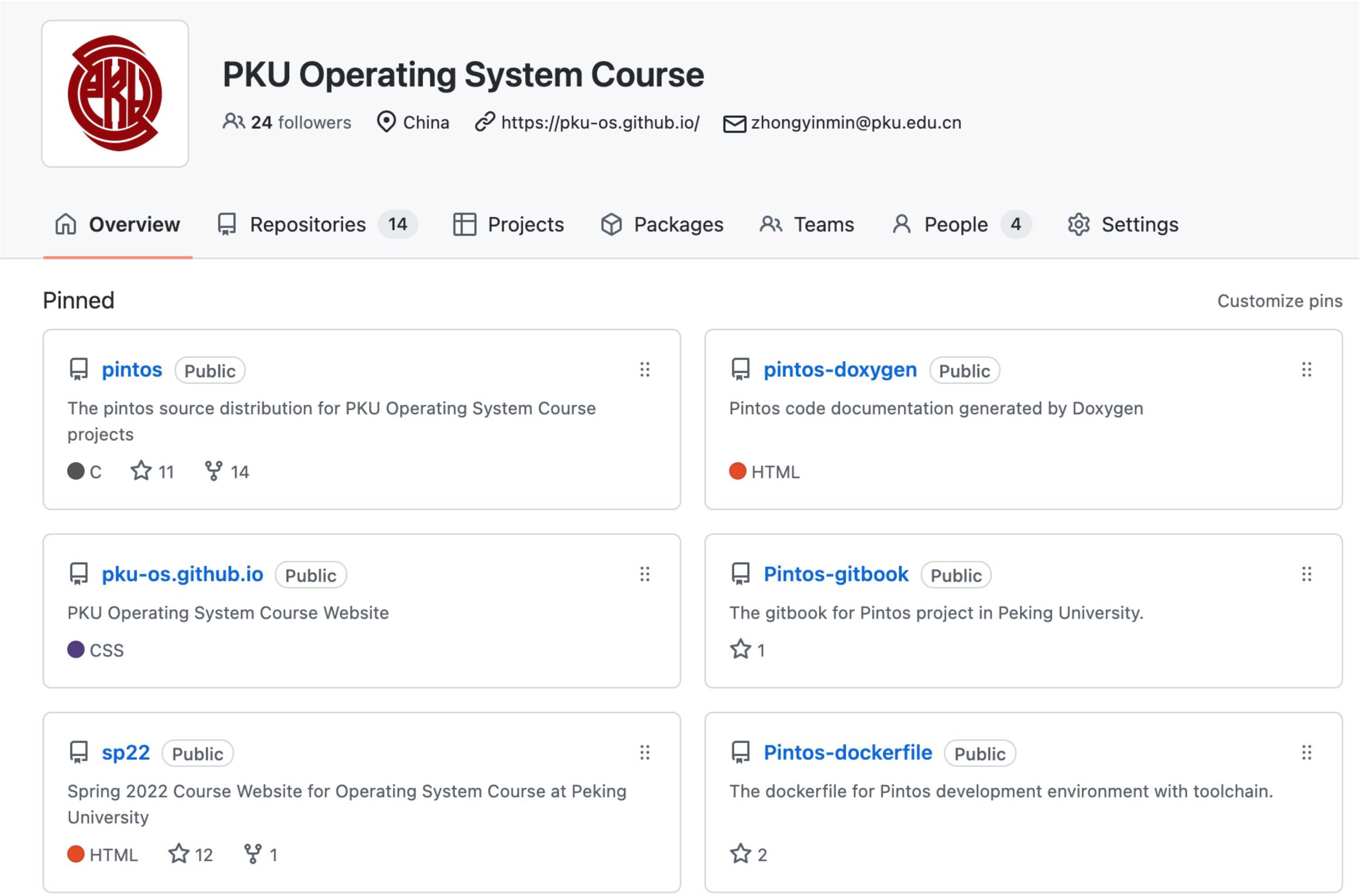
Conclusion

- Why Pintos?
  - Design and Implementation
  - Read, Design, Write, Debug the code
- What will you do in the projects?
  - Projects Map
  - Typical workflow
- How to survive the projects?
  - PintosBook
  - Ask questions
  - Good habits
  - Good attitude
- Lab0 FAQs: Booting Pintos, X86 mode

PKUOS

Welcome to the World of Operating System

https://github.com/PKU-OS



PKU Operating System Course

24 followers · China · https://pku-os.github.io/ · zhongyinmin@pku.edu.cn

Pinned

pintos Public
The pintos source distribution for PKU Operating System Course projects
● C ☆ 11 ⑂ 14

pintos-doxygen Public
Pintos code documentation generated by Doxygen
● HTML

pku-os.github.io Public
PKU Operating System Course Website
● CSS

Pintos-gitbook Public
The gitbook for Pintos project in Peking University.
☆ 1

sp22 Public
Spring 2022 Course Website for Operating System Course at Peking University
● HTML ☆ 12 ⑂ 1

Pintos-dockerfile Public
The dockerfile for Pintos development environment with toolchain.
☆ 2

Learn it,
Master it,
Love it,
and **Join** us.

Email: echostone@stu.pku.edu.cn
GitHub: EchoStone1101