

Operating System Labs

Yuanbin Wu
cs@ecnu

Operating System Labs

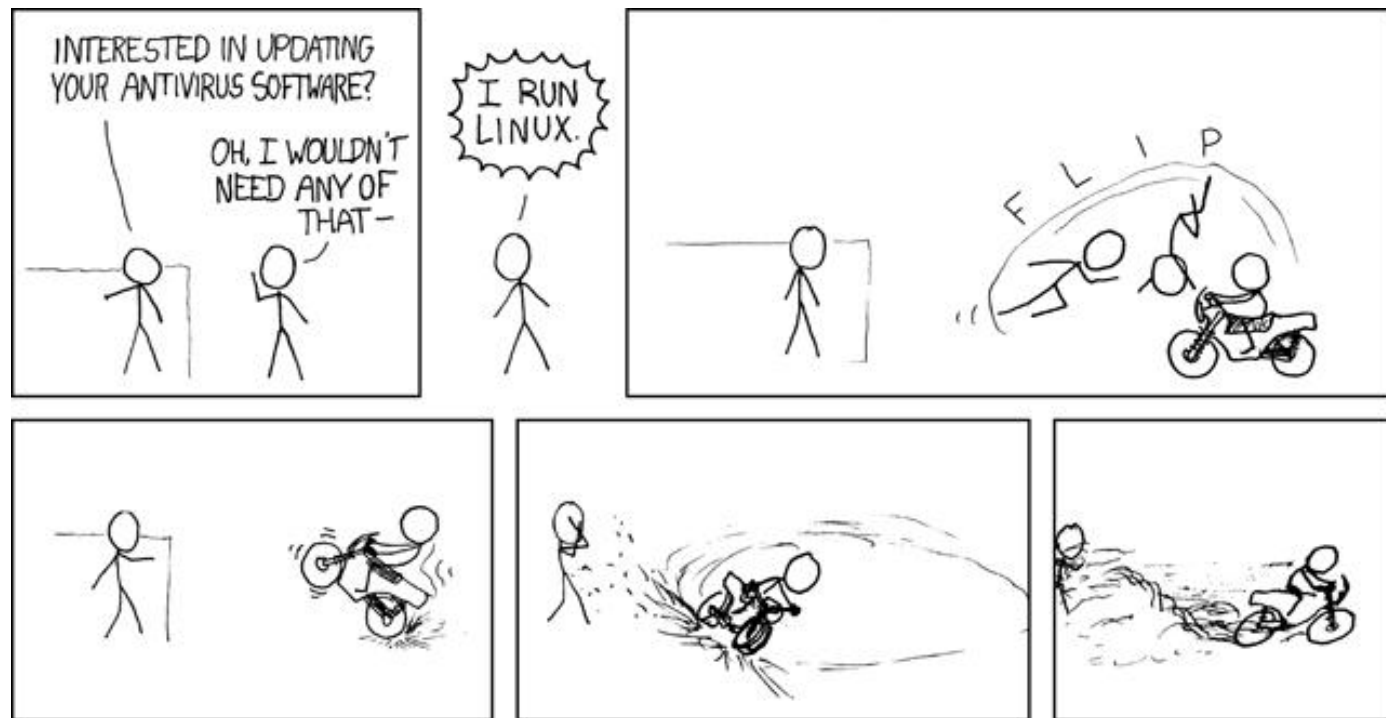
- Introduction to Unix (*nix)
- Course Overview

Operating System Labs

- Introduction to Unix (*nix)
- Course Overview

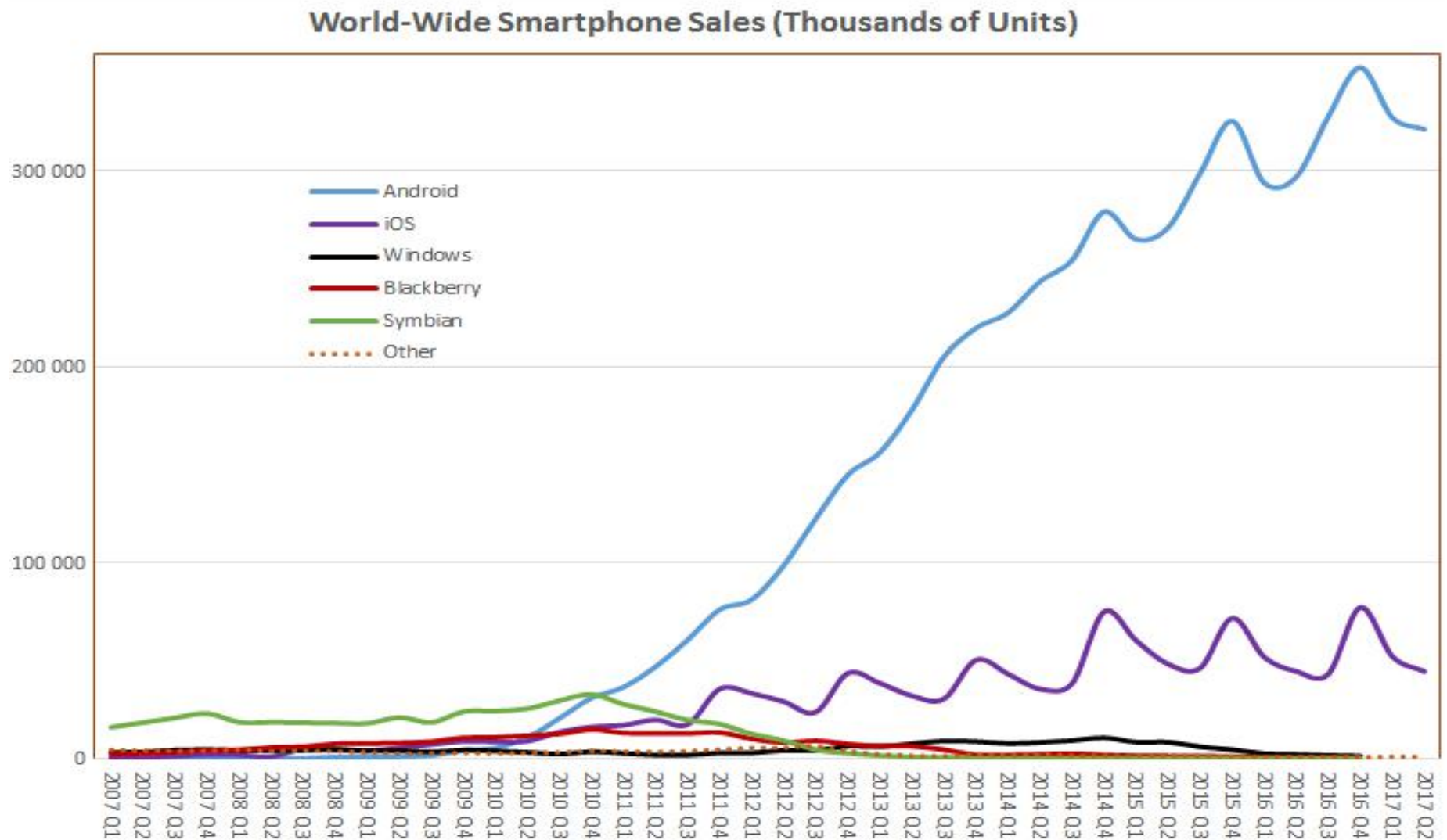
Unix / *nix

- What
 - A family of operating systems
 - Widely used
 - A cool thing



Unix / *nix

- Smartphone



Unix / *nix

- Smartphone



Unix / *nix

- Web Server

Source	Date	Unix, Unix-like				Microsoft Windows	References
		All	Linux	BSD	Unknown		
W3Techs	Feb 2015	67.8%	35.9%	0.95%	30.9%	32.3%	[54] [55]
Security Space	Feb 2014	<79.3%	N/A				[56] [57]
W3Cook	May 2015	98.3%	96.6%	1.7%	0%	1.7%	[58]

Unix / *nix

Category	Source	Date	Linux	BSD and other Unix	Windows	In-house	Other	
Desktop, laptop, netbook	Net Applications ^[68]	Dec 2014	1.34% (Ubuntu, etc.)	7.21% (OS X)	91.45% (7, 8, XP, Vista)			
Smartphone, tablet, handheld game console, smart TV, Wearable computer	StatCounter Global Stats ^[69]	Dec 2014	53.86% (Android)	31.10% (iOS)	1.87% (WP8, RT)		13.17%	
Server (web)	W3Techs ^[70]	Sep 2014	36.72% (Debian, Ubuntu, CentOS, RHEL, Gentoo)	30.18% (FreeBSD, HP-UX, Solaris, OS X Server)	33.10% (W2K3, W2K8, W2K12)			
Supercomputer	TOP500 ^[67]	Nov 2014	97.0% (Custom)	2.4% (AIX)	0.2%		0.2%	
Mainframe	Gartner ^[64]	Dec 2009	28% (SLES, RHEL)	72% (z/OS) UNIX System Services				
Gaming console	Nintendo, Sony, Microsoft, Valve Corporation	Jun 2015	<div>Device shipments, 2015</div> <div><div>Android</div><div></div><div>54.16%</div></div>					
Embedded	UBM Electronics ^[71]	Mar 2015	<div>iOS/macOS</div> <div></div> <div>12.37%</div>					41.1%
			<div>Windows</div> <div></div> <div>11.79%</div>					
Real time	NewTechPress ^[72]	Nov 2015	<div>Other</div> <div></div> <div>21.66%</div>					24.8%

OS Device Shipments, [Gartner](#)^[23]

According to [Gartner](#), the following is the worldwide device shipments (referring to [wholesale](#)) by operating system, which includes smartphones, [tablets](#), [laptops](#) and [PCs](#) together.

Unix / *nix

- As your desktop OS (work with it)
 - Coding
 - Web
 - Text processing
 - Multimedia
 - Shells
 - a working space
- It's “Free”

Unix / *nix



Unix / *nix

- For studying general OS concepts
 - Open source
 - High quality documents (freely available)
 - Great Community

Unix / *nix

- History

- Multics: mid-1960, MIT+GE+Bell Labs
 - “Multiplexed Information and Computing Service”, allowing multiple users to access a mainframe simultaneously
 - Segmentation, Dynamic linking
 - Complexity
 - Failed

Unix / *nix

- History

- Unics: 1969, Ken Thompson, Dennis Ritchie
 - A game called *Space Travel*
 - Smaller than Multics
 - Using PDP-7 assemble language
 - Hierarchical file system, process, device files, command-line interpreter
 - Single task
 - “Uniplexed Information and Computing Service”
- Core concepts of Unix

Unix / *nix

- History
 - C programming language, 1972, Dennis Ritchie
 - Rewrite unix for PDP-11

Unix / *nix



THE C PROGRAMMING LANGUAGE

GNU

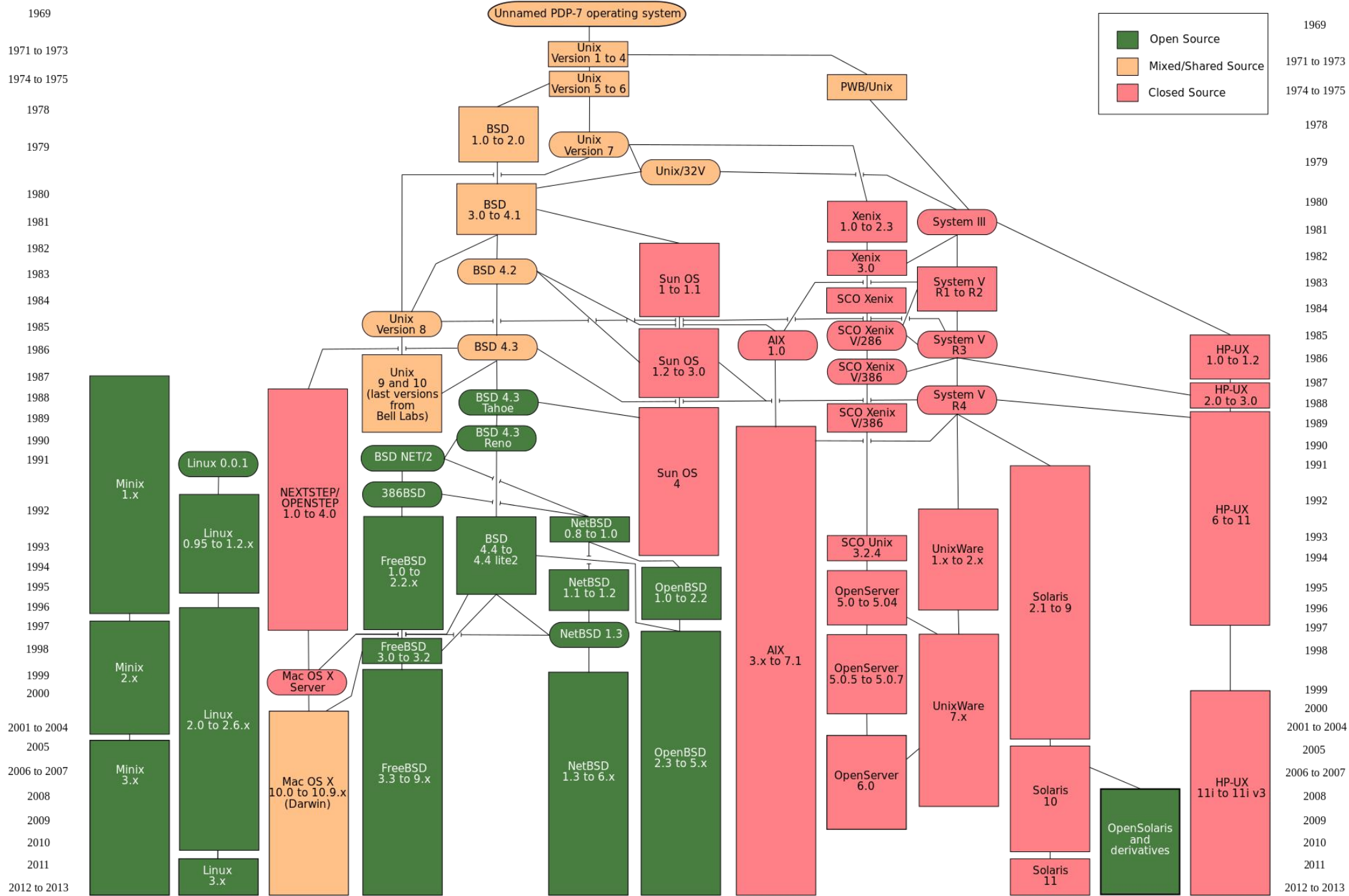
- History
 - GNU Project, 1983, Richard Stallman
 - GNU: GNU is Not Unix
 - Unix-like
 - Free software, contain no Unix code
 - GNU software
 - gcc (GNU C compiler)
 - gdb (GNU debugger)
 - Emacs
 - GNU kernel (GNU Hurd)
 - Free Software Foundation
 - Free Software License
 - GNU General Public License (GPL)

Unix

- Unix standardization
 - ISO C
 - Standard for the C programming language
 - POSIX
 - IEEE Portable Operating System Interface
 - SUS
 - Single Unix Specification

Unix

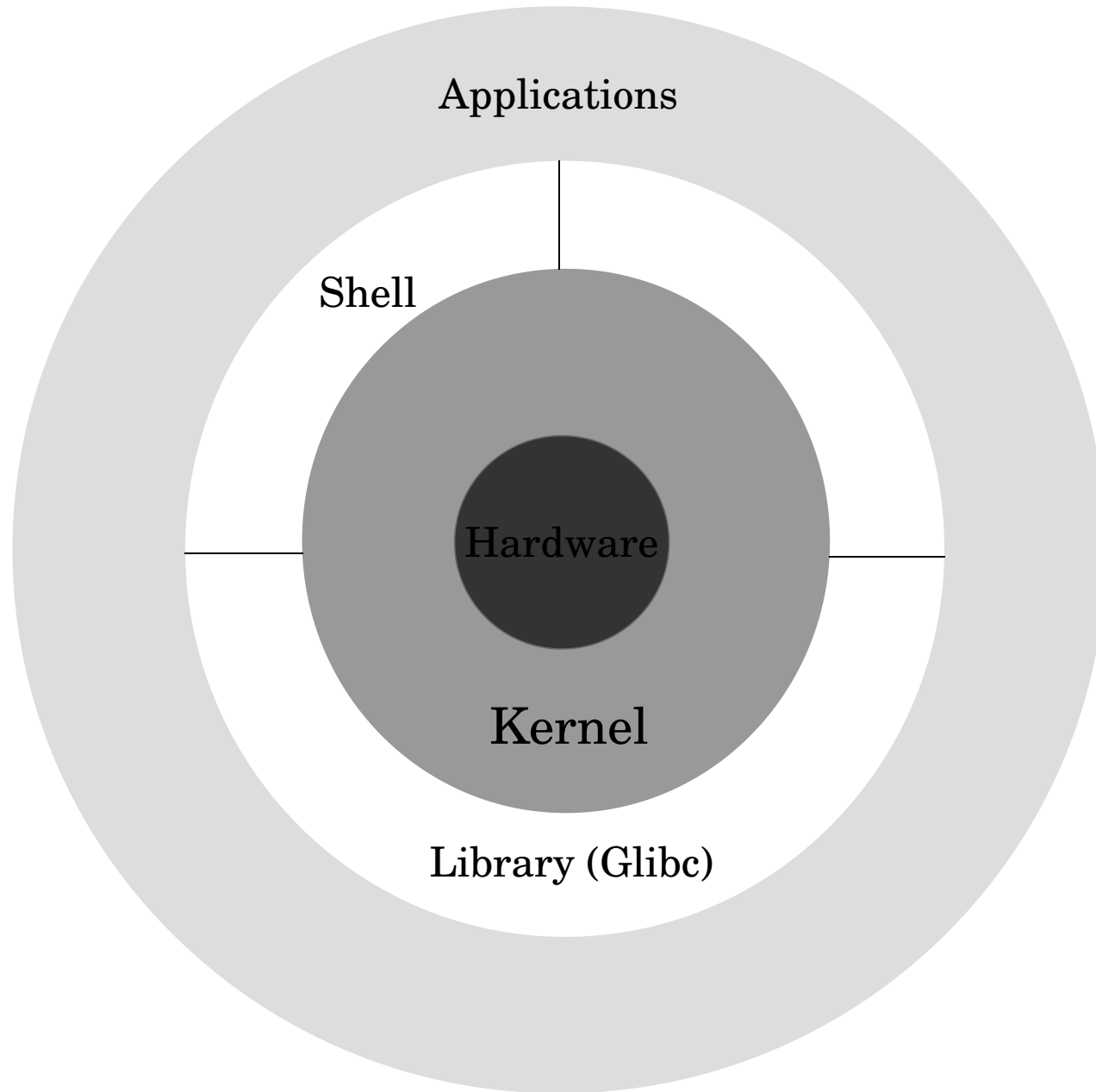
- Unix implementations
 - Unix v6, v7 (Bell lab)
 - FreeBSD (U.C. Berkeley)
 - Sun OS/Solaris (Sun)
 - System V (AT&T)
 - OS X (Apple)
 - Linux, 1991, Linus Torvalds
 - A (free) kernel with support of GNU packages
 - distributions
 - Ubuntu, Debian, CentOS, Fedora, Gentoo, ArchLinux
 - Android



Operating System

- OS in the eyes of users
 - Can I run XX software?
- OS in the eyes of CS students
 - Process, thread, paging, file system, ...
 - Can I write XX software on it to make some money/change the world?

Unix Operating System



Unix Operating System

- Login
 - User name
 - Password
- File and Directory
 - Hierarchical structure
 - /home/ybwu/Documents/myfile
 - Root directory: “/”

Unix Operating System

- Input and Output
 - Human-machine interaction
 - Keyboards
 - Monitors
- Files!

Unix Operating System

- “Everything is a file”
 - Documents
 - Directories
 - Hard-drives
 - Keyboards
 - Printers
 - /proc
- The same API: open, read, write, close

Unix Operating System

- File Descriptor
 - unsigned int
 - Allocate when open a file
 - Revoke when close a file
 - read() / write()

```
int fd = open("foo", "r");  
read(fd, buffer, size);  
close(fd);
```

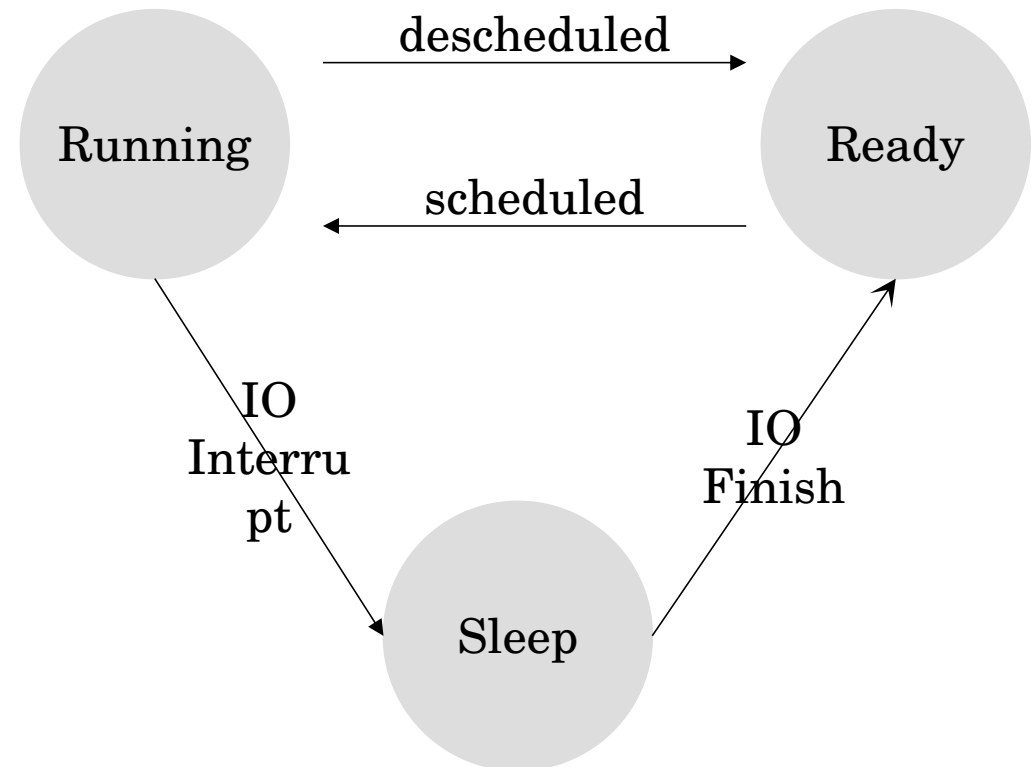
Unix Operating System

- File Descriptor
 - Standard input, output, error
 - 3 file descriptors
 - Automatically allocated for every process

- `read(STDIN_FILENO, buffer, size);`
- `write(STDOUT_FILENO, buffer, size);`
- `write(STDERR_FILENO, buffer, size);`

Unix Operating System

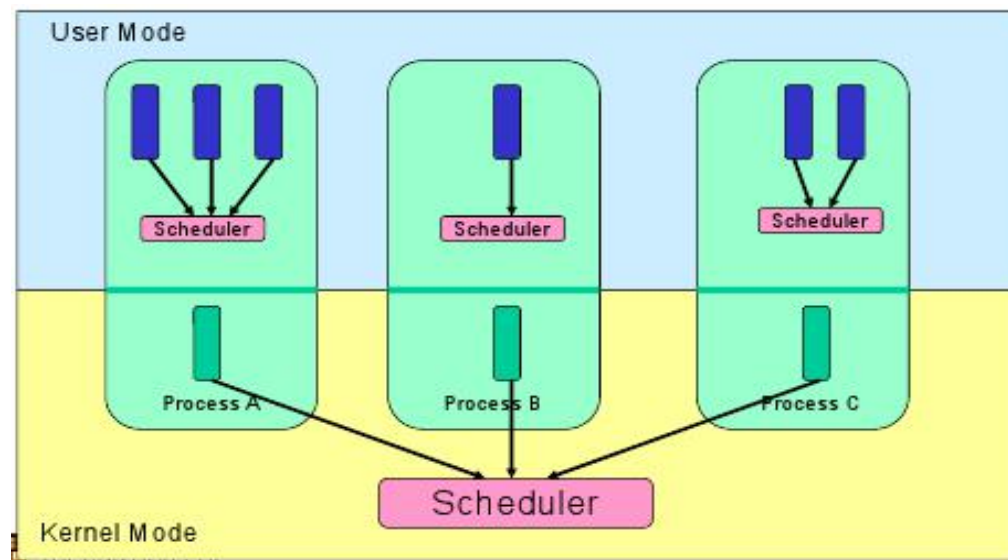
- Process
 - Process ID (PID)
 - Process status
 - ready, run, sleep



Unix Operating System

- Thread

- Processes that share same address spaces, file descriptors, ...
- Kernel thread
- User thread (fig. below)



Unix Operating System

- Communications among processes
- Example: Signal
 - Tell a process that something has happened
 - Example
 - pressing Ctrl+C generate a signal to terminate current process

Unix Operating System

- Memory management
 - Segmentation
 - Paging
- File system
 - Inode

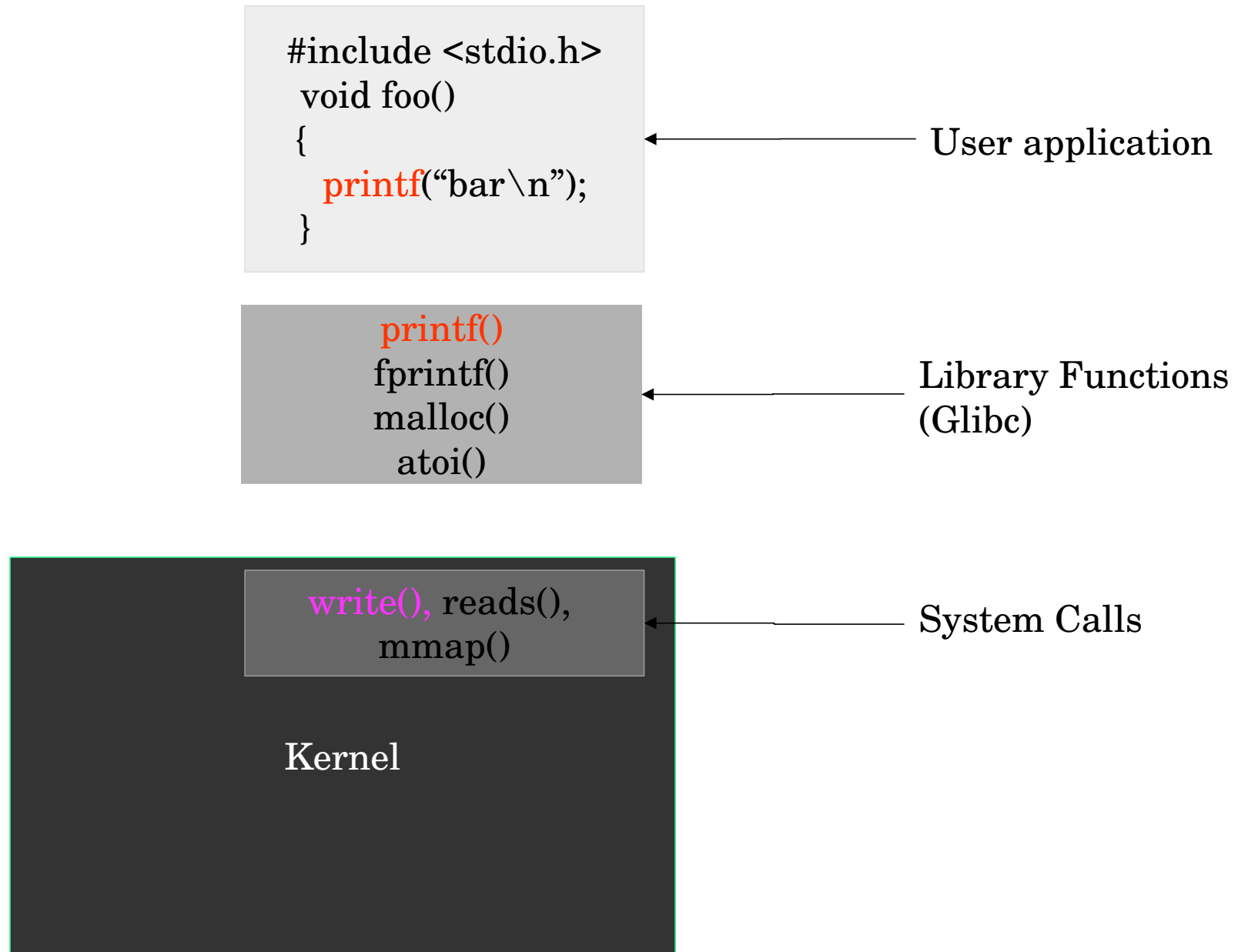
Unix Operating System

- Handling Errors
 - Not only report error, also provide detail info.
 - Variable: **errno**
 - Function: void **perror**(char* msg);
 - Print msg
 - Print error message string corresponding to the current errno

Unix Operating System

- System Call and Library Function
 - System Call:
 - Provided by kernel
 - Doing restricted operations with hardware
 - User mode, kernel mode
 - Library Function
 - Provided by user mode software developer
 - Some functions reused many times

Unix Operating System



Unix Operating System

- Summary
 - terms
 - File descriptor, stdin, stdout, stderr
 - Process, thread, Pid
 - errno, perror(),
 - Signal: Ctrl + C
 - System Call
 - Library Functions

Operating System Labs

- Introduction to *nix
- Course Overview

Course Overview

- Objectives
 - Reviewing core concepts of OS
 - Having some fun on coding
- How
 - Reading
 - Coding
 - Presentation

Course Overview

- In this semester:
 - 5 projects
 - each one has two parts (part a, part b)
 - Oral presentations
 - Course website:
 - <https://github.com/ybwu/ecnu-oslab-up/blob/main/23-Fall/index.md>

Course Overview

- Project 0 (part a)
 - To get familiar with Linux
 - Shell command
 - cd, ls, mkdir, rm, ...
 - Dev environment
 - gcc, gdb
 - Version control (and assignment submission)
 - git

Course Overview

- Project 0 (part b)
 - A simple key-value store
 - Warm up with Linux programming
 - I/O system call or `stdio.h`

Course Overview

- Project 1 (part a)
 - Implement your own shell
 - Linux process API
 - Redirect
 - Pipe

Course Overview

- Project 1 (part b)
 - xv6
 - xv6 is a modern re-implementation of Sixth Edition Unix in ANSI C for multiprocessor x86 systems.
 - Implemented by MIT PDOS
<https://pdos.csail.mit.edu/6.828/2019/xv6.html>
 - Adding a system call for xv6
 - Getting familiar with xv6
 - How a system call is handled

Course Overview

- Project 2 (part a)
 - Implement your own malloc() / free()
 - Dynamic memory allocation
 - The pointer of C programming language

Course Overview

- Project 2 (part b)
 - xv6 scheduler
 - Implement a multi-level feedback queue scheduler for xv6
 - Getting familiar with context switch, co-routines,...

Course Overview

- Project 3 (part a)
 - Implement your own lock
 - Introduction to concurrency
 - Linux pthread API
 - Thread safe data structures

Course Overview

- Project 3 (part b)
 - xv6 virtual memory
 - Adding a NULL pointer handler
 - Adjusting the arrangement of xv6 address space

Course Overview

- Project 4 (part a)
 - Implement a file defragmentor
 - Reorganize file blocks
 - basic concepts of file system

Course Overview

- Project 4 (part b)
 - xv6 kernel thread
 - supporting kernel threads in xv6
 - clone(), join(),...

Course Overview

- Projects
 - P0, P1: individual assignments
 - P2, P3, P4: group assignments (three member).
- Grading
 - The quality of your projects (code, documentation)
- Oral presentation
 - P2, P3, P4
- General advice
 - Start early
 - Build your projects incrementally
 - Bonus points for good development process

Course Overview

- How
 - Reading
 - Coding
 - Presentation

Course Overview

- Reading is important
 - You may spend >50% of your time on reading materials.



Course Overview

- Reading

- The main text book:

- *Operating Systems: Three Easy Pieces*, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
 - <http://pages.cs.wisc.edu/~remzi/OSTEP/>

Course Overview

- Reading
 - Reference for Unix programming:
 - *Advanced Programming in the UNIX Environment*, by W. Richard Stevens, Stephen A. Rago
 - Reference for C programming:
 - *The C Programming Language*, by Brian W Kernighan, Dennis M. Ritchie
 - Reference for Linux kernel:
 - *Linux Kernel Development*, by Robert Love

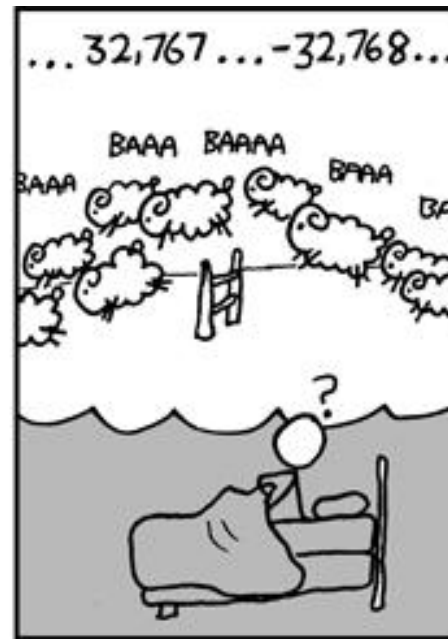
Course Overview

- Reading
 - RTM
 - “Read The Manual”
 - Xv6 source code/textbook
 - We will use an old version of xv6 in projects

Course Overview

- Coding
 - Using C, no C++, no Java ...
 - Compile with gcc
 - Debug with gdb
 - Maybe without IDE
- Please make your code
 - Well structured
 - Clean
 - Easy to read



Course Overview



Course Overview

- Presentation
 - you will present one of your group assignments
 - About
 - What have you done
 - How to accomplish them
 - Your favorite parts
 - What did you learn
 - ...

Operating System Labs

Month	Sep.			Oct.					Nov.				Dec.				Jan.	
Day	11	18	25	2	9	16	23	30	6	13	20	27	4	11	18	25	1	2
Week	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18
P0	L	L																
P1			L		L													
P2						L			L									
P3										L		O						
P4														L	O		O 	
L: lectures O: oral tests.																		

Tentative

Course Overview

- Policies
 - Plagiarism policy
 - Late policy

Course Overview

- Plagiarism policy
 - What is OK
 - Discuss programming specifications
 - What is the meaning of “redirection”
 - Discuss reading materials
 - What are the differences between exec functions?
 - Discuss implementation strategies
 - How to make the lock faster?

Course Overview

- Plagiarism policy
 - What is NOT OK
 - Copy codes/docs from someone
 - Beg someone to write a copy for you

Course Overview

- If we discover any improper code sharing
 - **ALL** participants will loss **ALL** credits of the project
- No Cheating!

Course Overview

- Late policy
 - For P0, P1
 - Late handins are NOT accepted.
 - For P2, P3, P4
 - Your group will have 3 “late days”.
 - You need to email TAs at least 1 hour before the dateline.
 - If all your 3 “late days” are used, late handins will not be accepted.
- Start Early!

Course Overview

- Submit your projects
 - github classroom (<https://classroom.github.com/>)
 - submissions as "git push"
- What is git?
 - a version control software

Course Overview

- More on grading
 - TAs grade your project based on the last commit before the deadline
 - We encourage "good" developing processes which
 - start early
 - build projects incrementally
 - Bonus points ($\approx 5\%$) for
 - evenly distributed commits
 - meaningful commit messages

Course Overview

- Contact

- Instructor

- 吴苑斌, ybwu@cs.ecnu.edu.cn
 - 713 Science Building B

- TAs

- 杨润东, 1353097778@qq.com
 - 缪岱烨, 2825467993@qq.com

- emails/issues are welcome

- Office hour

- 13:00-15:00, Wed.

Course Overview

- Project 0 due
 - 21:00 Sep. 24th

Course Overview

- Next Week:
 - Lectures :教书院 109