

Operating System Labs

Yuanbin Wu
cs@ecnu

Operating System Labs

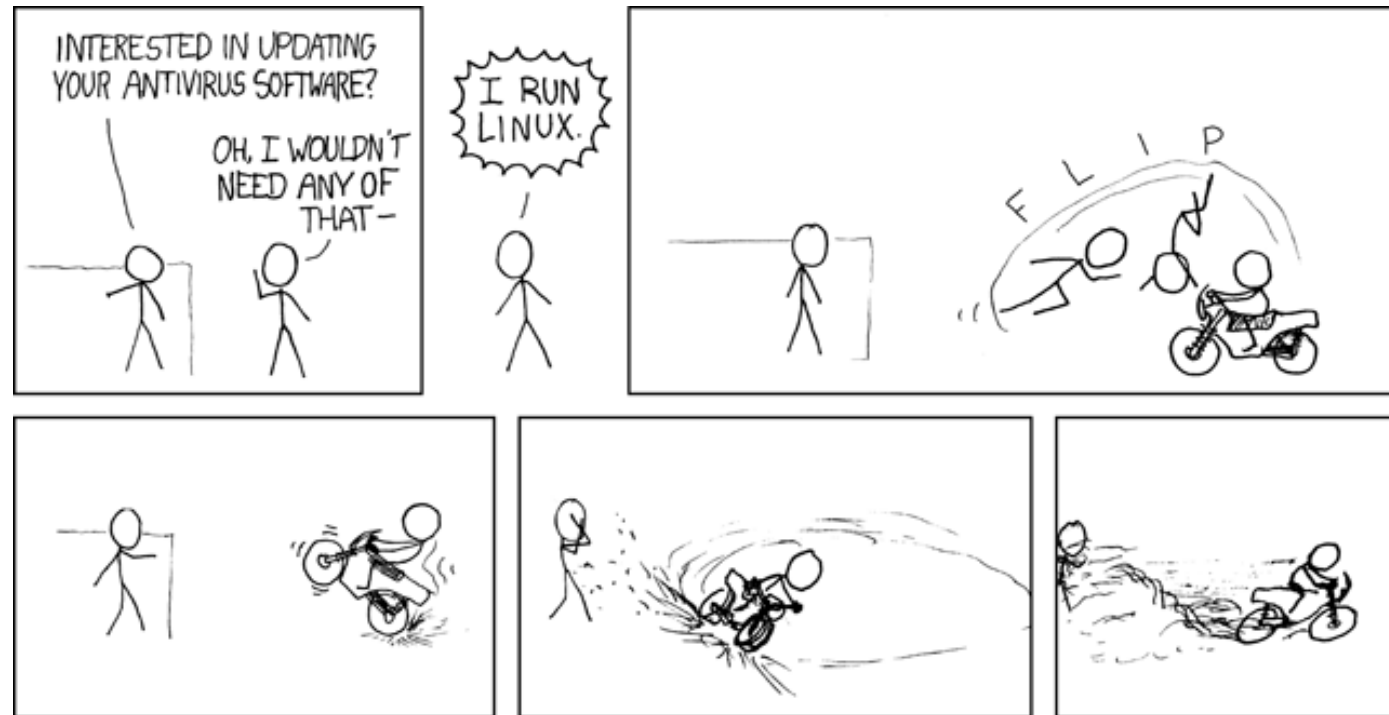
- Introduction to Unix (*nix)
- Course Overview

Operating System Labs

- Introduction to Unix (*nix)
- Course Overview

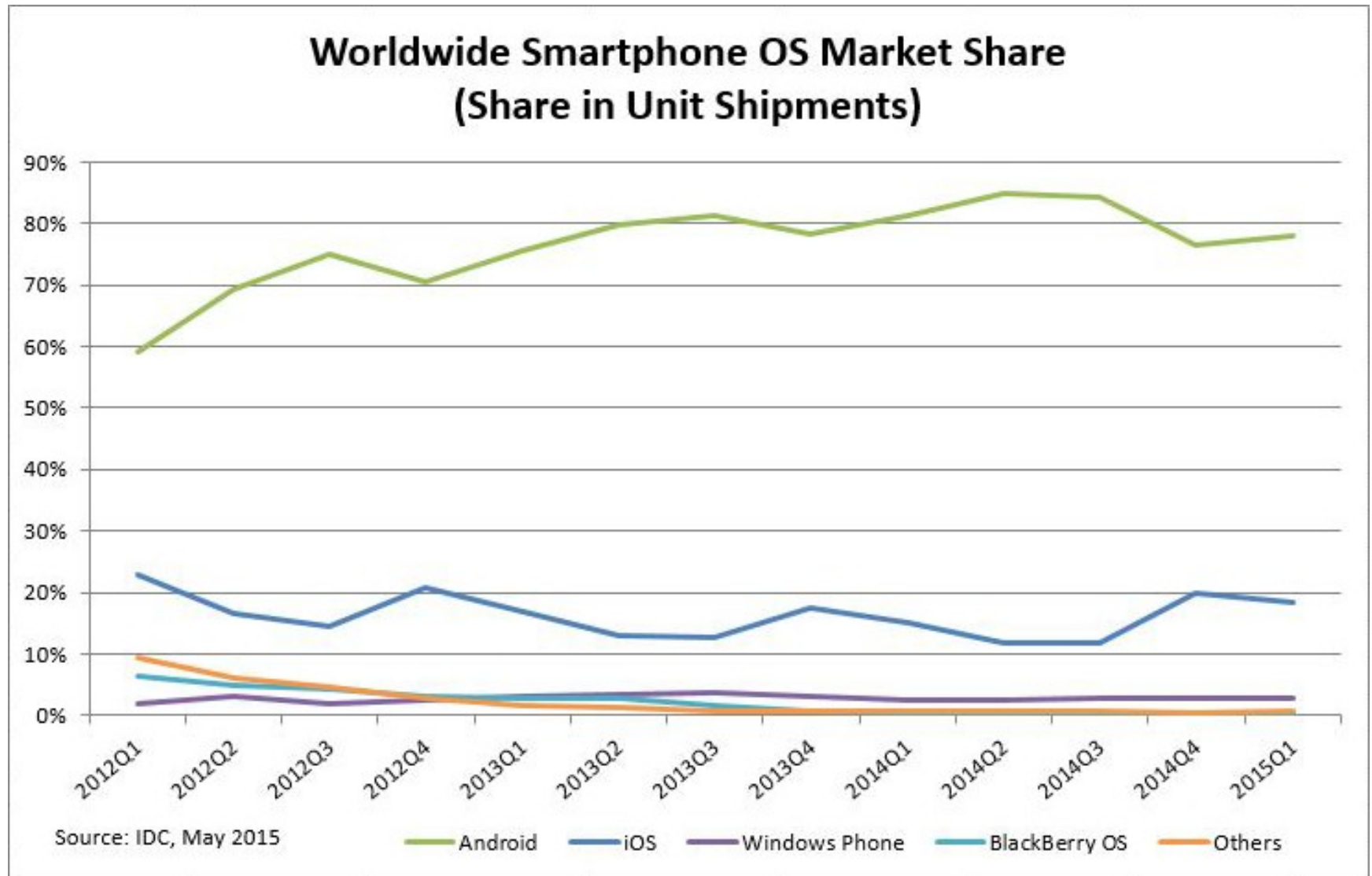
Unix / *nix

- What
 - An family of operating systems
 - Maybe the best operating system
 - A cool thing



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			>20.7%	[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 2008	28% (SLES, RHEL)	72% (z/OS) UNIX System Services			
Gaming console	Nintendo, Sony, Microsoft, Valve Corporation	Jun 2013	0% (SteamOS)	29.6% (PS3)	29.5% (Xbox 360)	40.9% (Wii)	
Embedded	UBM Electronics ^[71]	Mar 2012	29.44% (Android, Other)	4.29% (QNX)	11.65% (WCE 7)	13.5%	41.1%
Real time	NewTechPress ^[72]	Nov 2011	19.3% (Android)		35.8% (XPE, WCE)	20.1%	24.8%

Worldwide Device Shipments by Operating System

Source	Year	Android	iOS/OS X	Windows	Others
Gartner ^[2]	2014	48.61%	11.04%	14.0%	26.34%
Gartner ^[3]	2013	38.51%	10.12%	13.98%	37.41%
Gartner ^[4]	2012	22.8%	9.6%	15.62%	51.98%

Unix / *nix

- As your daily OS (work with it)
 - Coding
 - Web
 - Text processing (slides, documents)
 - Multimedia
 - The Shell

Unix / *nix

- The Shell
 - Your main working space

Unix / *nix



Unix / *nix

- As an example for studying OS
 - Open source
 - High quality documents are freely available
 - Great community

Unix / *nix

- History

- Multics: mid-1960, MIT+GE+Bell Labs
 - Segmentation, Dynamic linking
 - Complexity
- Unics: 1969, Ken Thompson
 - Using PDP-7 assemble language
- C programming language, 1972, Dennis Ritchie
 - Rewrite unix for PDP-11

Unix / *nix



THE

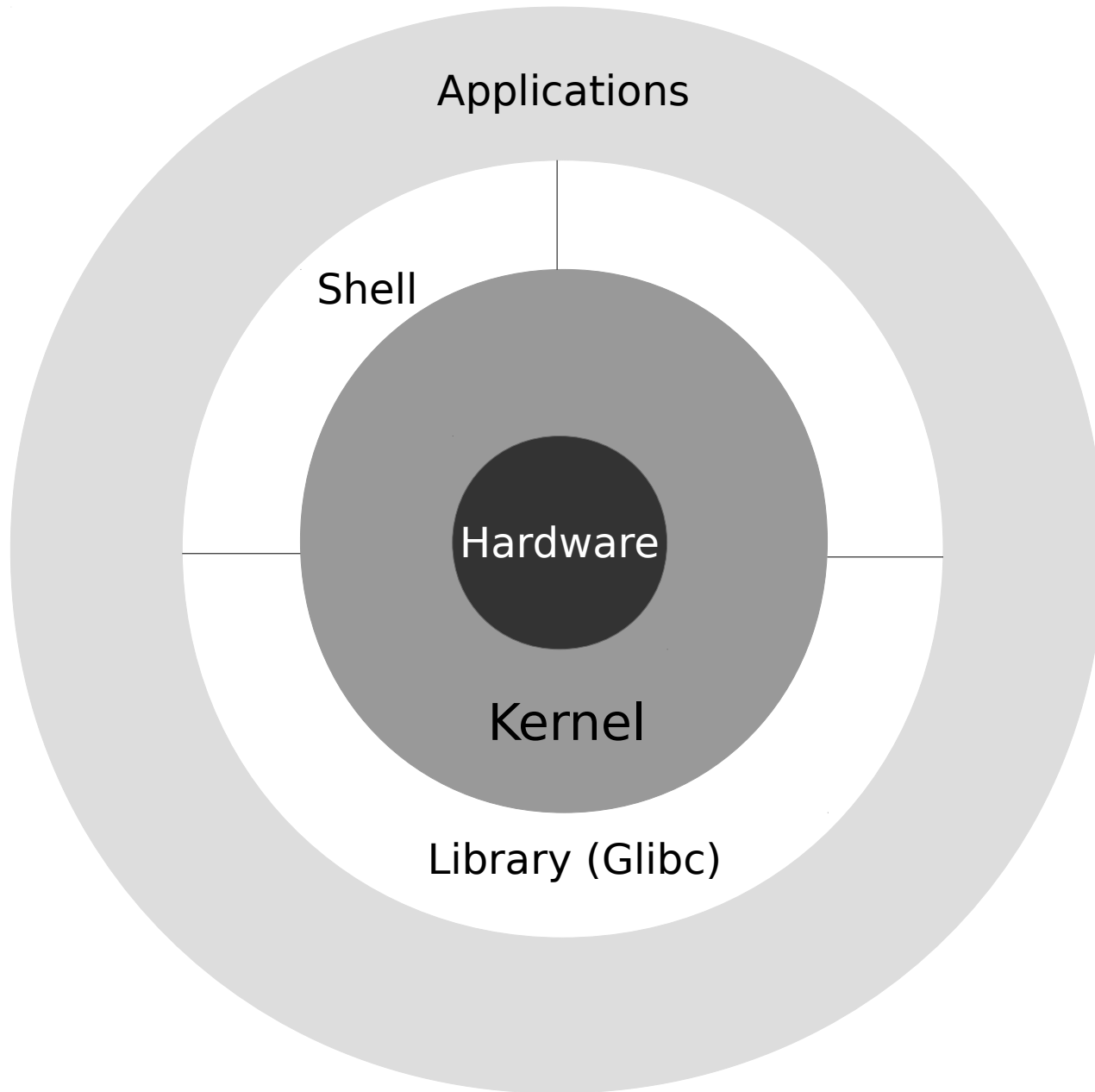
C

PROGRAMMING
LANGUAGE

Unix

- History
 - GNU Project, 1983, Richard Stallman
 - Free software
 - Linux, 1991, Linus Torvalds
 - A (free) kernel with support of GNU packages
 - Many distribution
 - Ubuntu, Debian, SentOS, Federa, Gentoo, ArchLinux
 - Android
 - FreeBSD, OS X, ...

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
 - 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

- Input and Output
 - Standard input, output, error
 - 3 file descriptors
 - Automatic allocated for every process

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

Unix Operating System

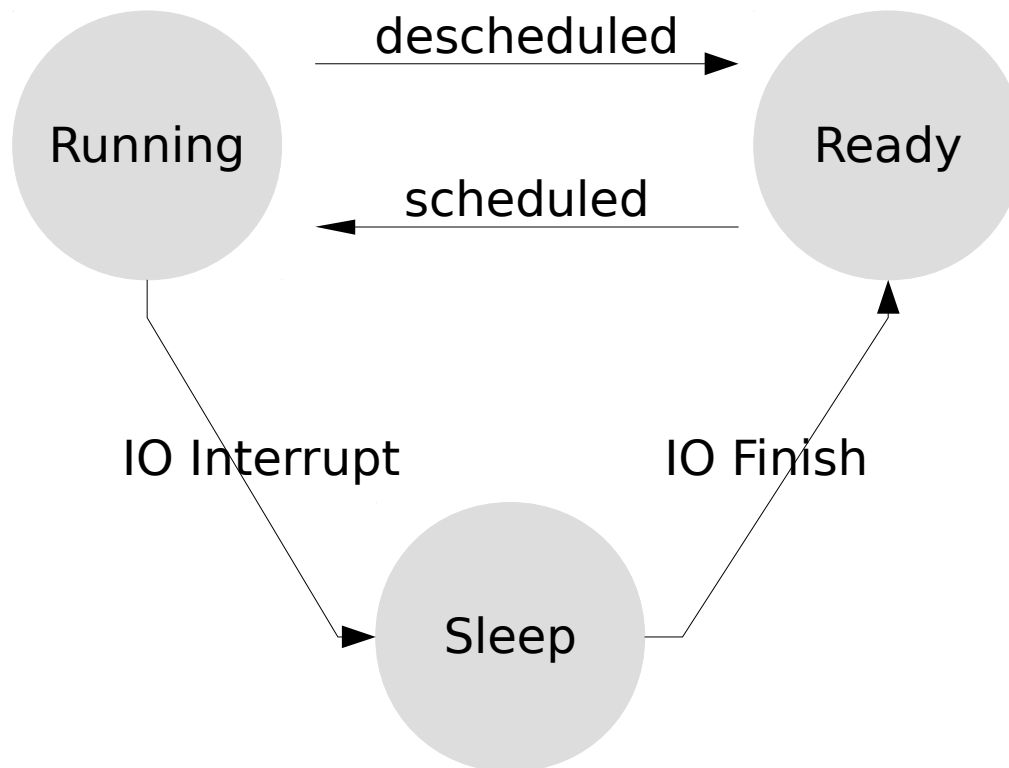
- “Everything is a file”
 - Documents
 - Directories
 - Hard-drives
 - Keyboards
 - Printers
 - /proc
- The same API: open, read, write, close

Unix Operating System

- Process

- Process ID (PID)

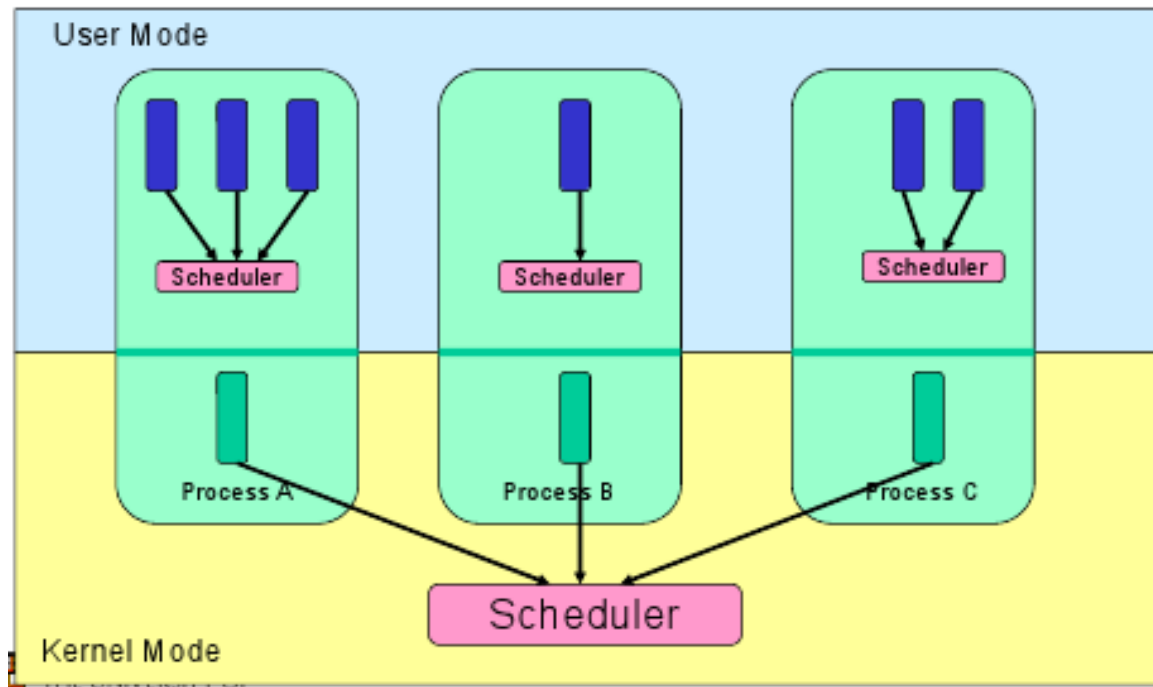
- Process status: ready, run, sleep



Unix Operating System

- Thread

- Processes that share same address spaces, file descriptors, ...
- Kernel thread / User thread



Unix Operating System

- Handle 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

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

Unix Operating System

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

Unix Operating System

```
#include <stdio.h>
void foo()
{
    printf("bar\n");
}
```

← User application

```
printf()
fprintf()
malloc()
atoi()
```

← Library Functions
(Glibc)

```
write(), reads(),
mmap()
```

← System Calls

Kernel

Unix Operating System

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

Operating System Labs

- Introduction to *nix
- Course Overview

Course Overview

- Objectives
 - To get some practical knowledge on OS
 - To discover the fun of programming
 - To feel the beauty of computer science
- How
 - Reading
 - Coding
 - Presentation

Course Overview

- In this semester:
 - 6 projects
 - 3 of them need oral presentations
 - Course website:
<http://ybwu.org/ecnu-oslabs/15-Fall/index.html>

Course Overview

- Project 0
 - To get familiar with Linux
 - Shell command
 - cd, ls, mkdir, rm, ...
 - Dev environment
 - gcc, gdb

Course Overview

- Project 1
 - Sorting
 - Warm up with linux programming
 - I/O system call

Course Overview

- Project 2
 - Implement your own shell
 - Linux process API
 - Redirect
 - Pipe

Course Overview

- Project 3
 - Implement your own malloc() / free()
 - Dynamic memory allocation
 - The pointer of C

Course Overview

- Project 4
 - Implement your own lock
 - Introduction to concurrency
 - Linux pthread API
 - Thread safe data structures

Course Overview

- Project 5
 - Implement a file defragmentor
 - Reorganize file blocks
 - Familiar with basic concepts of file system

Course Overview

- Projects
 - P0, P1, P2 should be completed individually
 - P3, P4, P5 should be completed in groups of three.
- Grading
 - The quality of your projects
 - Presentation
- General advice
 - Start early
 - Build your projects incrementally

Course Overview

- How
 - Reading
 - Coding
 - Presentation

Course Overview

- Reading

- The main 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
 - RTFM
 - “Read The Manual”

Course Overview

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

Course Overview



Course Overview

- Presentation
 - Everyone need to make presentation at least once (for one project)
 - About
 - What've you done?
 - How did you accomplish them?
 - Which are your favorate parts in your implementation?
 - What did you learn from the project?

Operating System Labs

	9.7	14	21	28	10.5	12	19	26	11.2	9	16	23	30	12.7	14	21	28
	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17
P0	L																
P1			L														
P2					L		L										
P3								L									
P4											L	▼	L				
P5															L	▼	▼

P3 oral

P4 oral

P5 oral

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, P2
 - Late handins are NOT accepted.
 - For P3, P4, P5
 - Your group will have 3 “late days”.
 - You need to email TA 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

- Contact

- Instructor:

- Yuanbin Wu, ybwu@cs.ecnu.edu.cn
 - 610 Information Building

- TA:

- Chengchao Huang, ecnucchuang@126.com
 - Changzhi Sun, 51141201057@ecnu.cn

- Office hour

- Wed 16:30 – 17:30 @ 4-211

- Advanced Machine Learning (Wed. 15:00 – 16:30, 4-211)