Operating System Labs

Yuanbin Wu
cs@ecnu

Operating System Labs

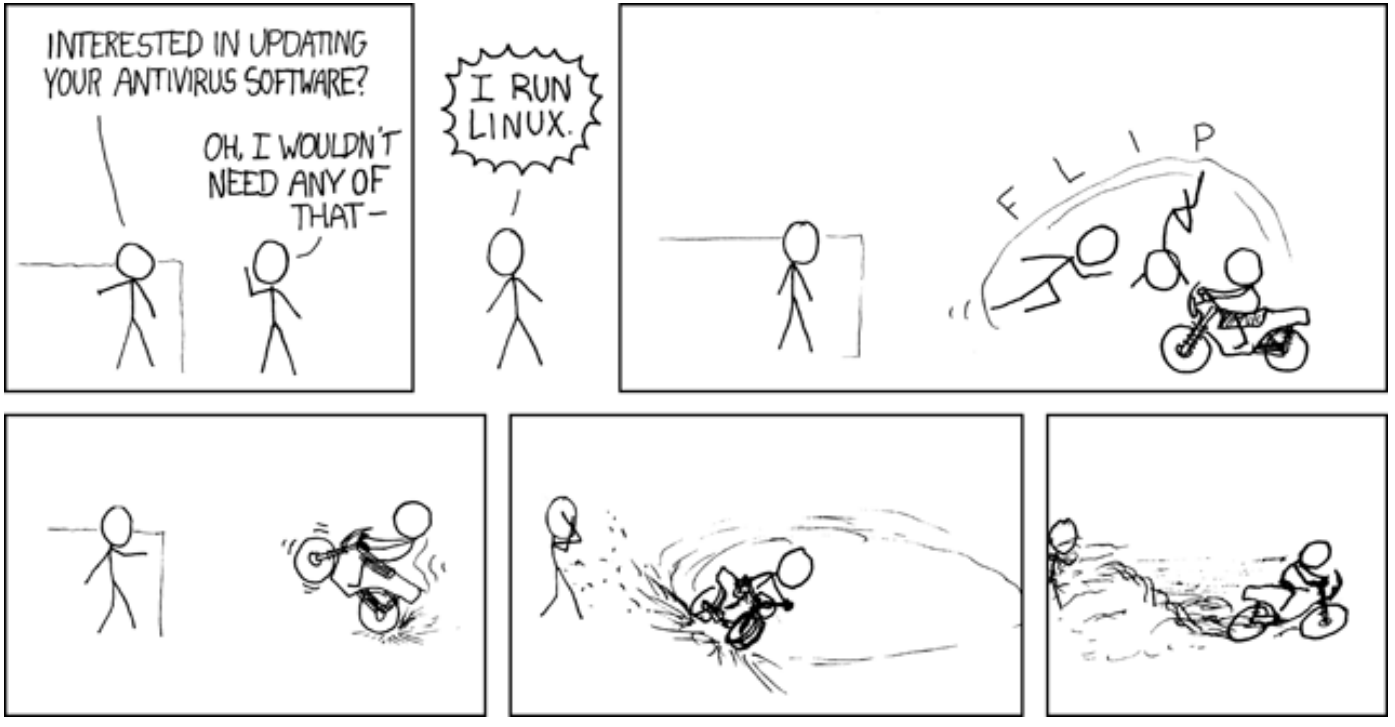
- Introduction to Unix (*nix)
- Course Overview

Operating System Labs

- Introduction to Unix (*nix)
- Course Overview

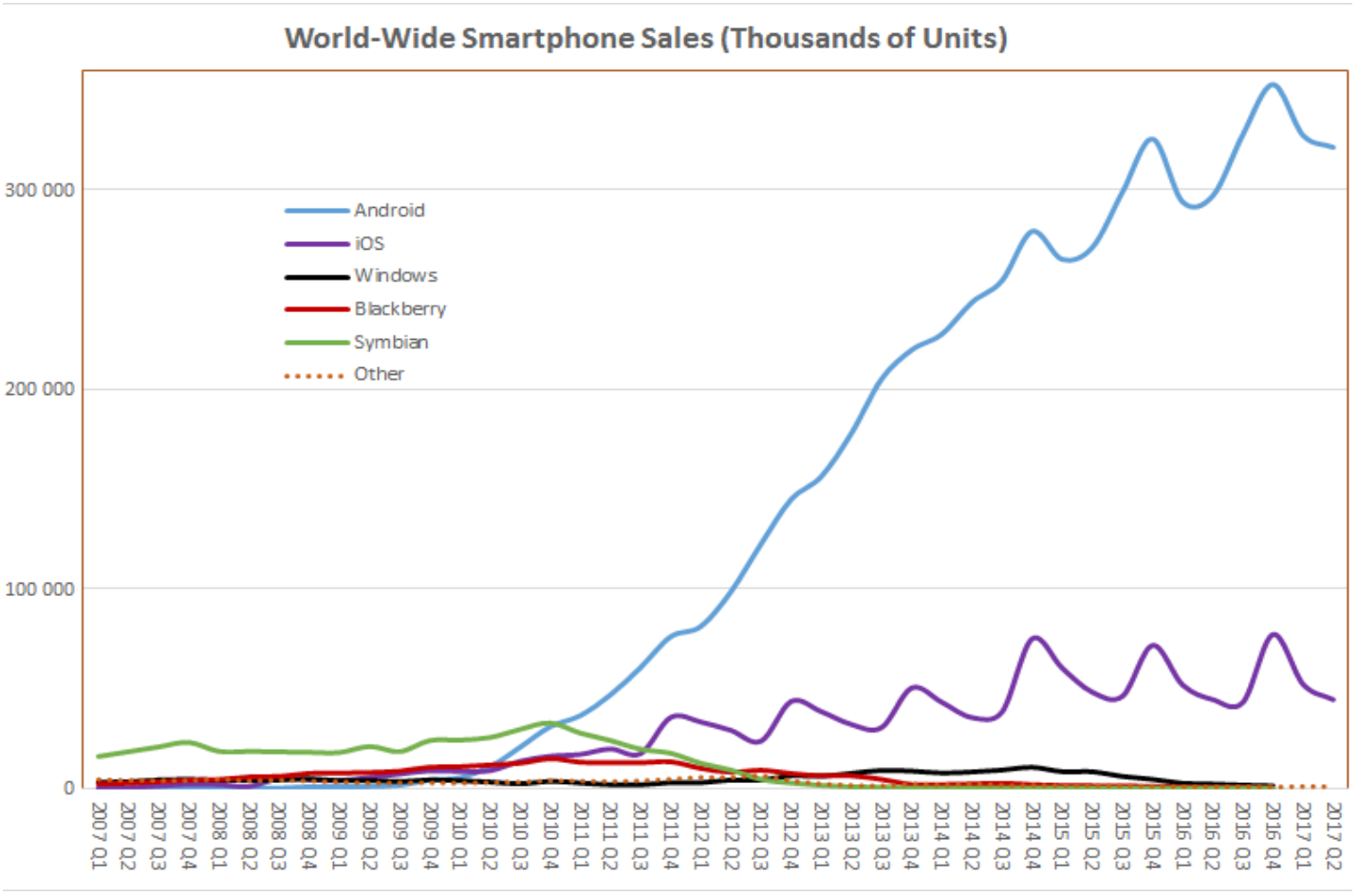
Unix / *nix

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



Unix / *nix

- Smartphone



Unix / *nix

- Smartphone

Unix / *nix

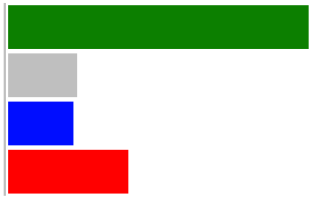
- Web Server

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			>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 200	28% (SLES, RHEL)	72% (z/OS) UNIX System Services			
Gaming console	Nintendo, Sony, Microsoft, Valve Corporation	Jur 20					
Embedded	UBM Electronics[71]	Ma 20					41.1%
Real time	NewTechPress[72]	No 20					24.8%

Device shipments, 2015



Android	54.16%
iOS/macOS	12.37%
Windows	11.79%
Other	21.66%

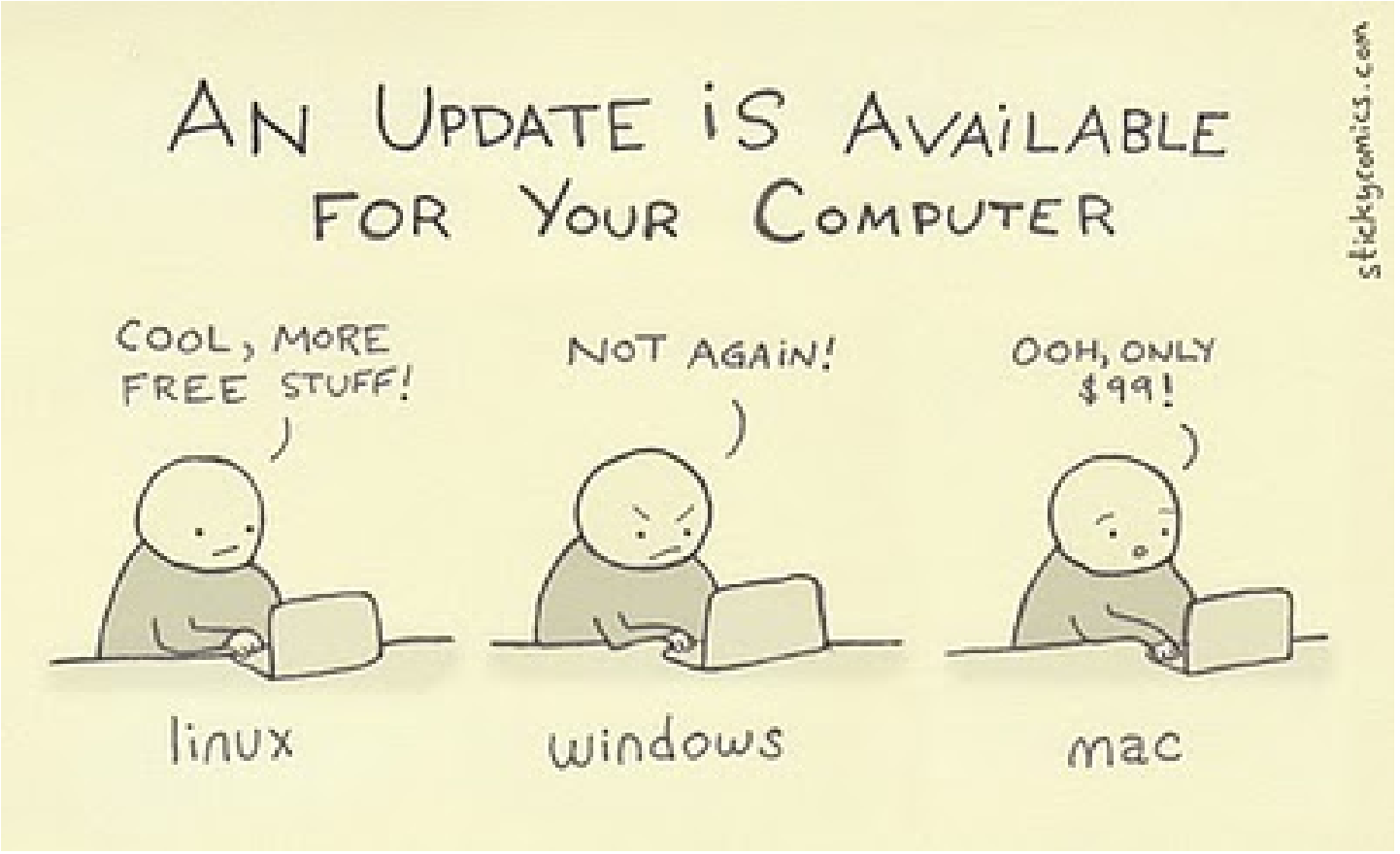
OS Device Shipments, Gartner[23]

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


AN UPDATE IS AVAILABLE FOR YOUR COMPUTER
COOL, MORE FREE STUFF!
NOT AGAIN!
OOH, ONLY $99!
linux
windows
mac
stickycomics.com

Unix / *nix

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

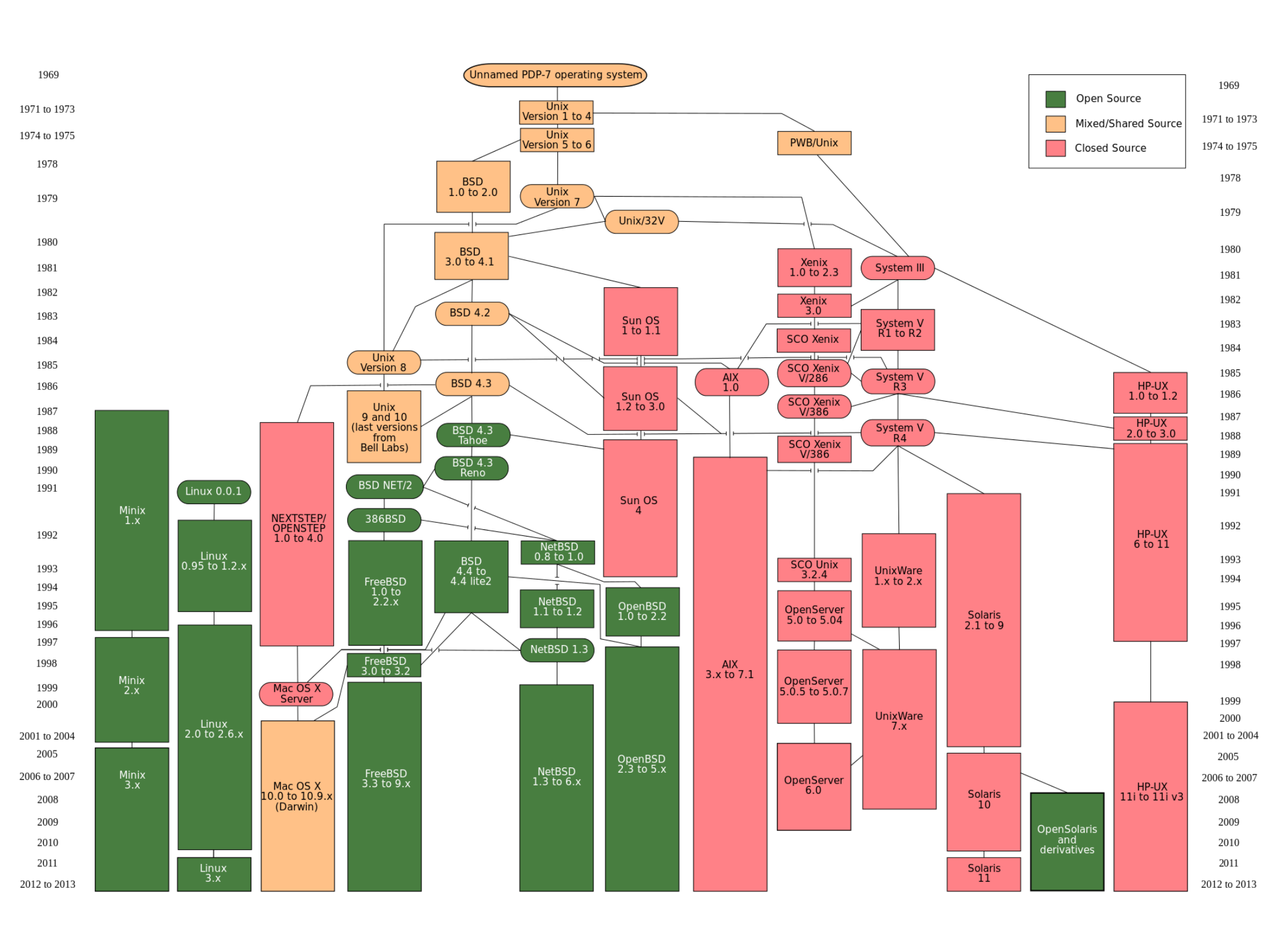
- History
  - GNU Project, 1983, Richard Stallman
    - GNU: GNU is Not Unix
    - Unix-like
    - Free software, contain no Unix code
    - GNU software
      - gcc (GNU C compiler)
      - gdb (GNU debugger)
      - Emacs
    - Free Software Foundation
    - Free Software License
      - GNU General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

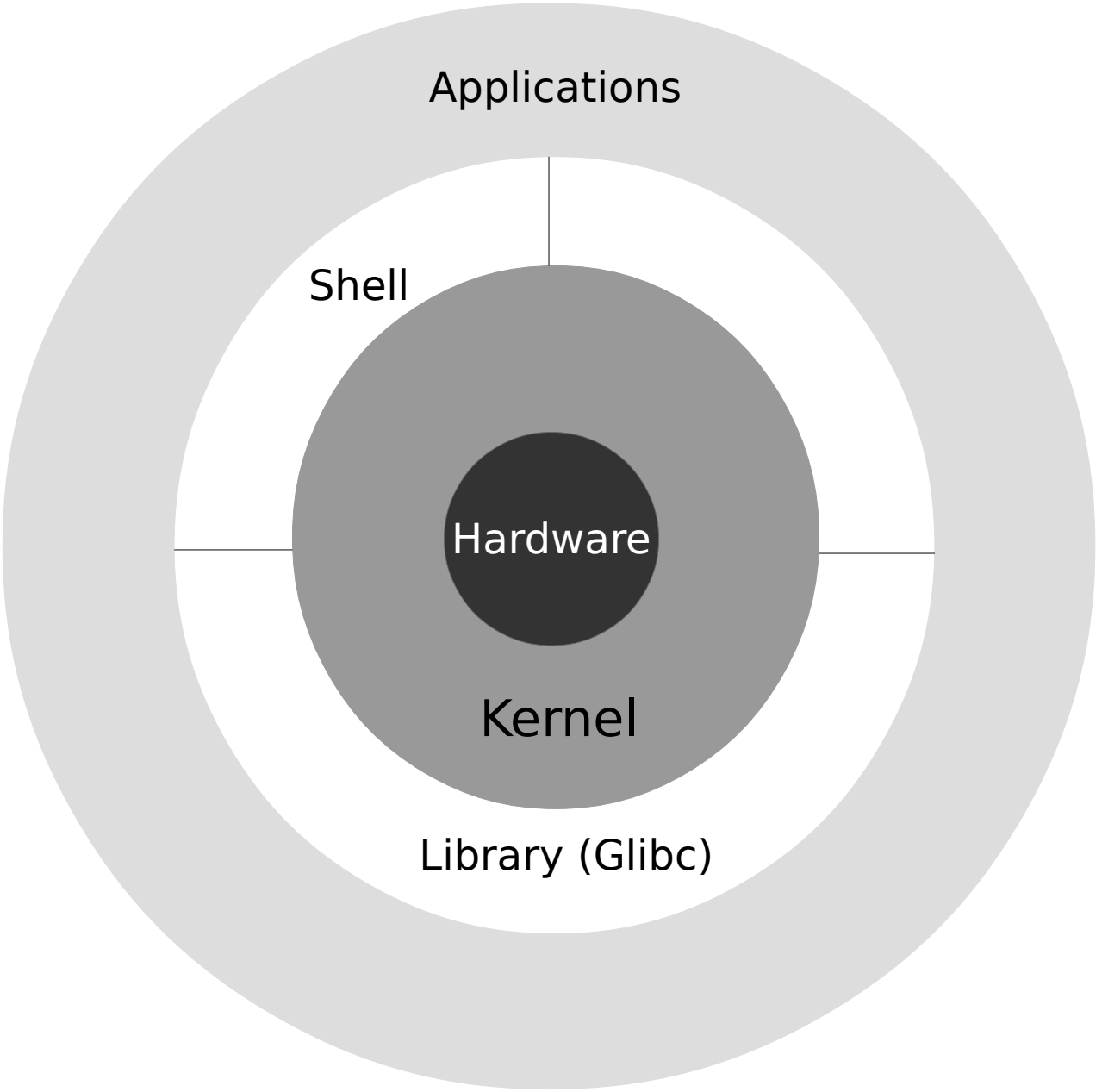
- Unix implementations
  - Unix v6, v7 (Bell lab)
  - FreeBSD (U.C. Berkeley)
  - Sun OS/Solaris (Sun)
  - System V (AT&T)
  - OS X (Apple)
  - Linux, 1991, Linus Torvalds
    - A (free) kernel with support of GNU packages
    - distributions
      - Ubuntu, Debian, CentOS, Federa, Gentoo, ArchLinux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


1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013
Open Source
Mixed/Shared Source
Closed Source
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
HP-UX 1.0 to 1.2
BSD 4.3
Sun OS 1.2 to 3.0
System V R3
SCO Xenix V/386
Unix 9 and 10 (last versions from Bell Labs)
HP-UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
SCO Xenix V/386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
BSD NET/2
Sun OS 4
NEXTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
HP-UX 6 to 11
SCO Unix 3.2.4
UnixWare 1.x to 2.x
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
NetBSD 0.8 to 1.0
OpenBSD 1.0 to 2.2
NetBSD 1.1 to 1.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
AIX 3.x to 7.1
Minix 2.x
NetBSD 1.3
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x
Linux 2.0 to 2.6.x
Mac OS X Server
OpenBSD 2.3 to 5.x
OpenServer 6.0
Solaris 10
OpenSolaris and derivatives
Minix 3.x
Mac OS X 10.0 to 10.9.x (Darwin)
FreeBSD 3.3 to 9.x
NetBSD 1.3 to 6.x
HP-UX 11i to 11i v3
Linux 3.x
Solaris 11

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 Input and File Output
  - File operations
  - 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);
```

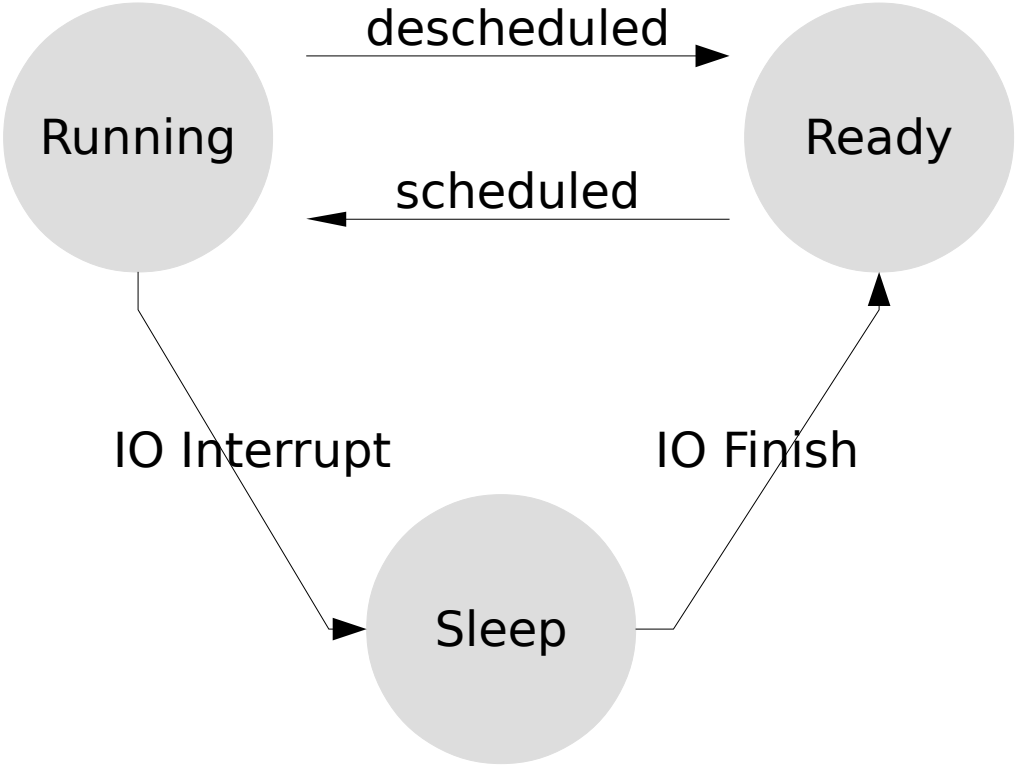
Unix Operating System

- File Input and File 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);
```

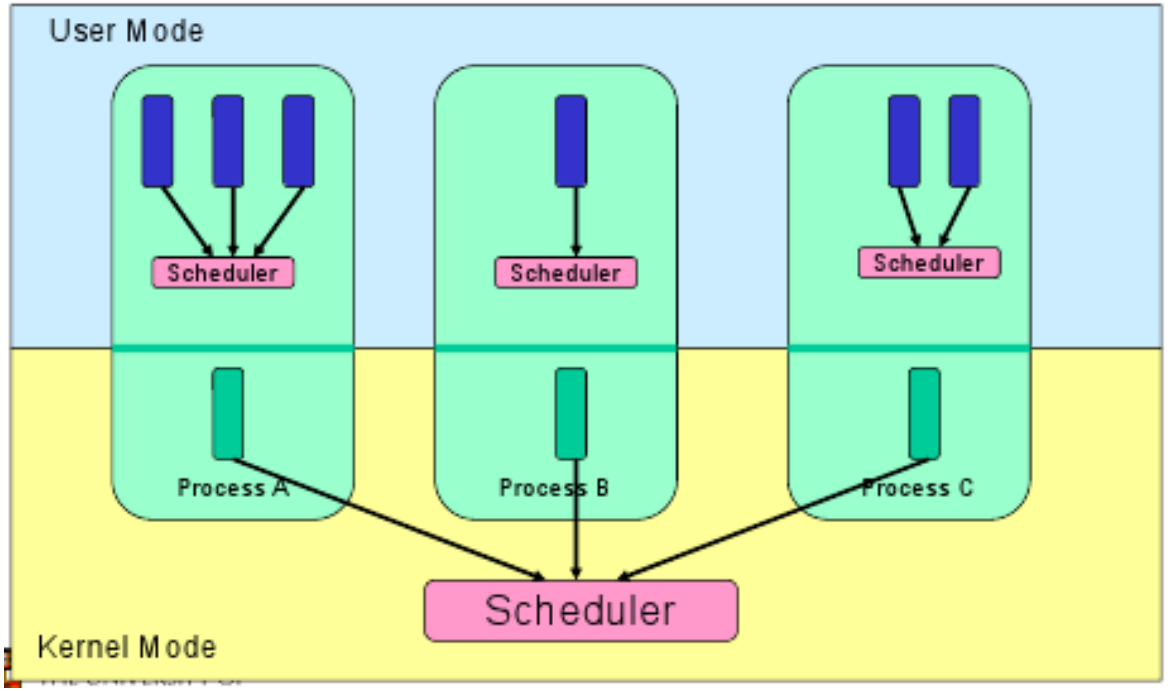
Unix Operating System

- 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

- Communications of 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

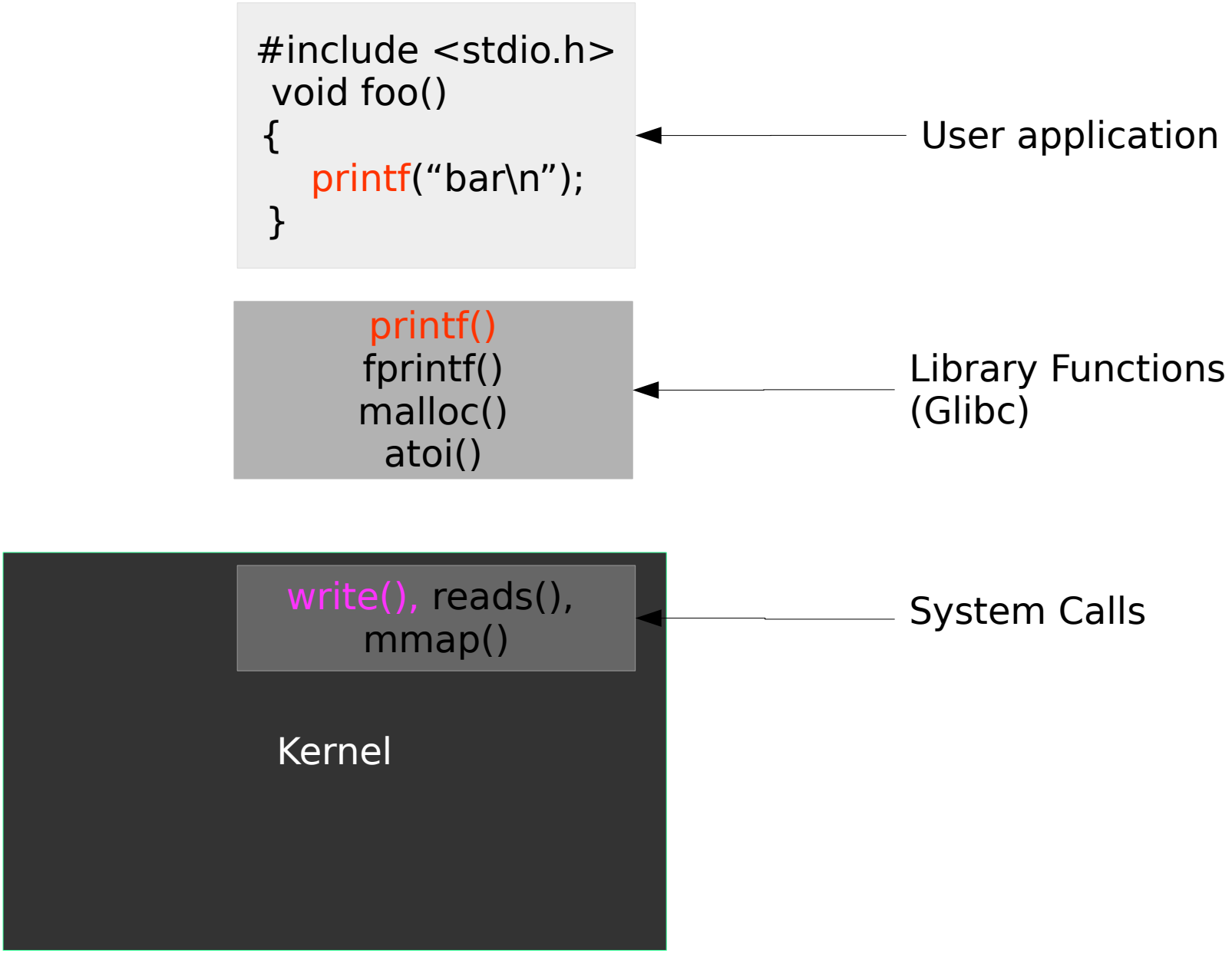
Unix Operating System

- 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)
  - Bonus points to oral presentations
  - Course website:

    http://ybwu.org/ecnu-oslabs/index.html

Course Overview

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

Course Overview

- Project 0 (part b)
  - Sorting
  - Warm up with Linux programming
  - I/O system call

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: single
  - P3, P4, P5: groups of three.
- Grading
  - The quality of your projects (code, documentation)
- Optional: oral presentation (with bonus points)
  - P3, P4, P5
- General advice
  - Start early
  - Build your projects incrementally

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

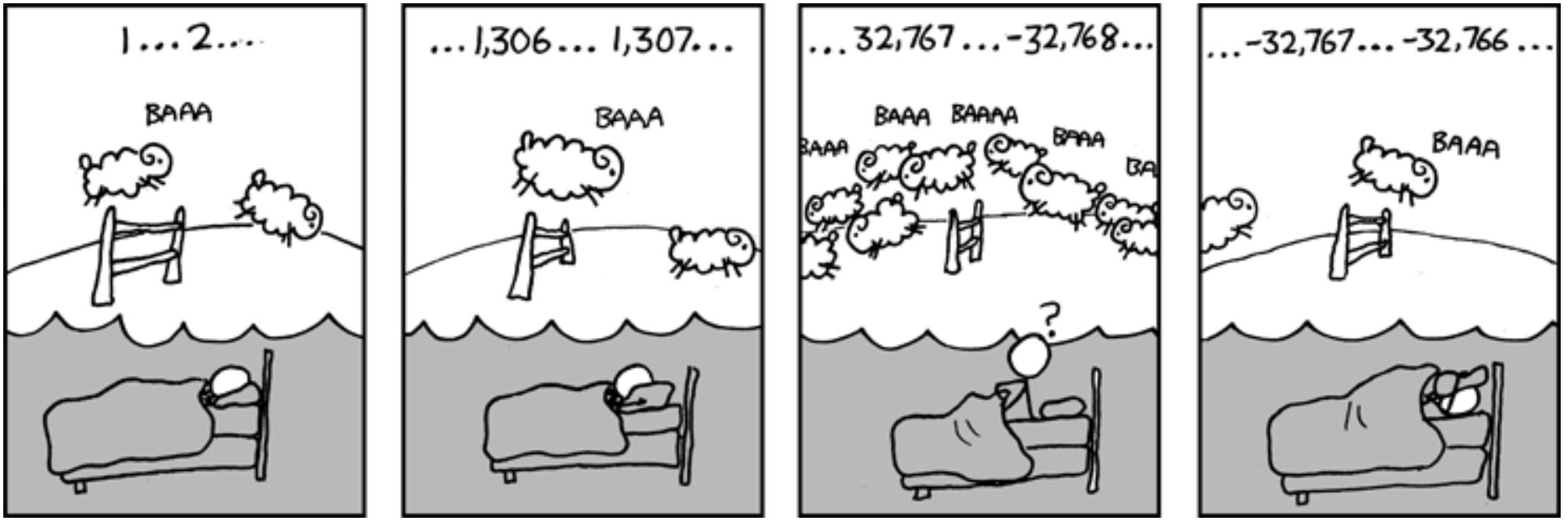
Course Overview

- Reading
  - RTFM
  - “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
- Make your code
  - Well structured
  - Clean
  - Easy to read

Course Overview



Course Overview

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

Operating System Labs

	9.14	21	28	10.5	12	19	26	11.2	9	16	23	30	12.7	14	28	1.4	11	18
	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	L	oral						
P4														L	oral			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
    - Late handins are NOT accepted.
  - For P2, P3, P4
    - 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:
    - 吴苑斌, ybwu@cs.ecnu.edu.cn
    - 911 Science Building B
  - TA:
    - 高怡， 296804129@qq.com
    - 梁梦雯, moonliang_vvv@163.com
- Office hour
  - TBD

Course Overview

- Project 0 due
  - 21:00 Sep. 27th
- Submit your project to
  - oslab2020@163.com
  - see course websites for more requirements

Course Overview

- Next Week:
  - Lectures : 教书院 105
  - Labs: 理科楼 B519