

W4118 Operating Systems



Junfeng Yang

Bad News

- ❑ This is a DIFFICULT course
 - “Most difficult” rated by CS alumni
- ❑ Unfamiliar low-level systems programming
 - C and Assembly
 - No abstraction, close to hardware
- ❑ Intense
 - Should be 6 units instead of 3 ...
 - Most of those struggling in CS lounge or CLIC lab late or possibly overnight were OS students
- ❑ And you have to climb up 7 floors for each lecture

Good News I

- Not interested in learning OS and low-level systems programming? **Don't take this course!**
 - Waive if you have taken similar course
 - Personalized track

Good News II

- Heavy, but worth it
 - "Most useful after graduating" rated by alumni
- Works hard → good grade

Why study OS?

- OS = arguably the most fundamental software
 - We do almost everything with computers through OS
- By studying OS, you will
 - Gain a good understanding of OS
 - Learn some **portable** tricks
 - Gain a good understanding of **the big picture**
 - How do hardware, programming language, compiler, algorithms, OS work together?

Possibly

- Get **a job** at Google/Microsoft/VMware/...
- Get started in systems research
- Apply OS ideas to your area
- ...

What will we learn?

□ OS concepts

- What does an OS do?
 - Abstract hardware: processes, threads, files
 - Manage resources: CPU scheduling, memory management, file systems

□ OS implementations

- How does an OS do these in general?
- How does xv6, an implementation of Unix 6th edition on x86, do these?
 - Complete, real code

What will we learn? (cont.)

- Hands on OS programming experience in xv6
 - **Best** way: learning by doing
 - Six programming assignments; five are **kernel programming assignments**
 - **Practical programming skills**
 - How to understand code
 - How to modify
 - How to debug
 - How to work with others
 - ...

xv6 overview

- ❑ Implementation of Unix 6th Edition on x86
- ❑ A subset of Unix system calls
 - `fork`, `exec`, `read`, `write`, `pipe`, ...
- ❑ Runs with multiple processors/multicore
- ❑ User-mode programs (can do some real stuff)
 - `mkdir`, `rm`, ...
- ❑ Can boot on real machine

Understanding xv6

□ Lectures + study code on your own + programming assignments

□ Resources:

<http://www.cs.columbia.edu/~junfeng/os/resources.html>

- gcc inline assembly
- Intel programming manual
- QEMU monitor commands
- gdb commands
- PC hardware programming

xv6 files

- ❑ **Generic:** `asm.h` (segmentation), `mmu.h`, `x86.h` (inline assembly), `elf.h` (ELF format), `types.h`, `param.h` (kernel constants), `string.c`
- ❑ **Boot:** `bootasm.S`, `bootother.S`, `bootmain.c`, `main.c`
- ❑ **Process and virtual memory:** `proc.h`, `proc.c`, `vm.c`, `zombie.c`, `pipe.c`, `exec.c`, `kalloc.c`, `sysproc.c`, `swtch.S`, `initcode.S`
- ❑ **System call and interrupt:** `syscall.h`, `traps.h`, `trap.c`, `syscall.c`, `trapasm.S`, `vector.S`
- ❑ **Synchronization and multicore:** `spinlock.h`, `mp.h`, `spinlock.c`, `mp.c`
- ❑ **Disk and file system:** `defs.h`, `fs.h`, `stat.h`, `file.h`, `buf.h`, `fcntl.h`, `bio.c`, `fs.c`, `file.c`, `sysfile.c`
- ❑ **Device:** `kbd.h`, `kbd.c`, `timer.c`, `lapic.c`, `picirq.c`, `uart.c`, `console.c`, `ide.c`, `ioapic.c`
- ❑ **User-mode programs:** `user.h`, `sh.c`, `forktest.c`, `wc.c`, `kill.c`, `cat.c`, `grep.c`, `stressfs.c`, `ln.c`, `ulib.c`, `echo.c`, `init.c`, `ls.c`, `printf.c`, `umalloc.c`, `usertests.c`, `mkdir.c`, `rm.c`, `usys.S`
- ❑ **Other:** `mkfs.c`, `Makefile`

My background

- ❑ Research area: systems
 - Publish in systems conferences
 - e.g., OSDI, SOSP, NSDI
- ❑ Research wise, practical kind of guy; believe only in stuff that works and is useful
- ❑ System reliability research for N years
 - Systems research shifted from pure performance to reliability starting around 2000
 - I was fortunate to be at the cutting edge of this shift
 - Hacked Linux & Windows, found some of the worst bugs
 - Current focus: concurrency
- ❑ Cool projects available for interested students
 - <http://rcs.cs.columbia.edu/student-projects.html>

Some of my previous results

- Built several effective bug-finding tools
 - One transferred to Microsoft SQL Azure
- Found 100+ serious bugs
 - Security holes: write arbitrary memory
 - Data loss errors: lose entire file system data
 - Errors in commercial data center systems: stuck w/o progress
- Serious enough that developers immediately worked on fixes
 - google "lkml junfeng"
- Reported at news website (e.g. lwn.net)

Basic Course Info

- Course website:

<http://www.cs.columbia.edu/~junfeng/os/>

- Next: tour of course website

Homework 1

- ❑ Written: basic OS concepts
- ❑ Programming: warm up, sanity test
 - Part A: set up xv6 and qemu
 - Part B: simple shell
- ❑ Apply CS account
- ❑ Look for teammates