

Outline

- **Last time**
 - General discussion of Operating Systems
 - Course logistics
- **This time**
 - More intro, History (Chapters 1 and 2)
 - Assignment #1 out (later today, due Thurs Sept 13 11am)
- **Next time**
 - Hardware review (Chapter 2)
 - Chapter 3: OS structures
- **Next next time (Thur night / Fri afternoon)**
 - More Chapter 3: OS structures
- **Group question of the day:**
 - You're a SW developer for a customer: why might you choose one OS vs. another?

CS414: Operating Systems

Before we start

- **Make-up class this week (for last Tuesday)**
 - 6:00-7:15 Thurs OLS 120
 - 3-4:15 Fri OLS 120

CS414: Operating Systems

Before we start – results of survey

	Written	Run	No
Win	46	12	
Linux/UNIX	17	23	18
MacOS	2	32	23

- **Little experience with emacs/vi, C**
- **Broad interest in handhelds** ↔ **supercomputers**
 - Maybe not so much handhelds
- **Equal interest in Linux and Windows**
 - Programming for, Administering, Design and implementation of
- **"very interested" in networking**
- **Use own computers? Ave: 5.69 (of 57 responses)**
 - 27 / 60-ish seem to want to use their own computers (7+); 21 seem to NOT want to use their own computers (3-)
- **Student comments...**

CS414: Operating Systems

Before we start – survey comments

- "experience with parallel computing?"
- "use plenty of examples"
- "be aware of skill levels"
- "candy"
- "class involvement"
- "PS3"
- "why don't we learn about MacOS"?
- "Chocolate"
- "please provide extensive tutoring for programming assignments"
- "I'm graduating after this semester"
- "break down the material please"
- "Please teach us C"
- "I hate Java. Would rather manage my own memory."
- "I would like to learn a little more about device drivers."
- "throw candy at people"
- "everyone needs linux exposure" (11)
- "nothing right now. First class was good."
- "5. can I do both?"
- "I've got a nasty schedule this semester. I will need some working with."

CS414: Operating Systems

Before we start

- **The "blue slides"...**
 - These materials are part of the *Windows Operating System Internals Curriculum Development Kit*, developed by David A. Solomon and Mark E. Russinovich with Andreas Polze
 - Microsoft has licensed these materials from David Solomon Expert Seminars, Inc. for distribution to academic organizations solely for use in academic environments (and not for commercial use)
 - **Copyright Notice (on the blue slides)**
 - © 2000-2005 David A. Solomon and Mark Russinovich
- **Note:** I might have made minor modifications on these slides

CS414: Operating Systems

From last time

- **This course**
 1. What an OS does ("theory")
 2. How an OS is implemented ("practical experience")
 3. How the OS impacts your user programs ("exploit the OS for your own purposes")
- **Multiple Views**
 - OS as juggler
 - provide illusion of "more machine" than actually exists (*virtual machine*)
 - OS as government
 - protect users from each other, allocate resources efficiently and fairly, provide safe and secure communication
 - OS as complex system
 - keeping OS design and implementation as simple as possible

CS414: Operating Systems

Effect of Changing Hardware/Software Costs on OS

	1983 (desktop)	2006 (my laptop)
MIPS	0.5	1600 x 2
Price/MIP	\$100,000	\$.21
memory	1 MByte	1.25 Gbyte
Network	10 Mbps	100 Mbps – 1 Gbps
Backing store (tape)	1 GByte	1 TeraByte (120G Disk)
Addressable bits	32	32 or 64

What other area has seen such orders of magnitude change? (Answer: none)

CS414: Operating Systems

Compare And Contrast OS design environments

	UNIX (1970s)	NT (1980/1990s)	?? (2000/2010s)
Address space	16b, swapping	32-bit, linear VM	64-bit, ??
CPU perf	KIPS	MIPS	GIPS
Synchr	IRQL	Test&Set, Cmpr&Swap	Transactional memory?
Memory size	KBytes	MBytes	GBytes
Hard concurrency	none	SMP	High-Multicore
Mass storage	Kbytes, slow seek	Mbytes, slow seeks	GBytes, no seeks TBytes
Distrib. computing	Tape	Client/server	Peer-to-peer

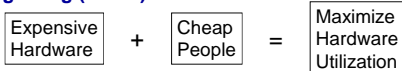
- Old OS designs can (of course) be ported, but
- How you would design an OS on blank paper?
 - How should the CPU system architecture evolve?

Microsoft

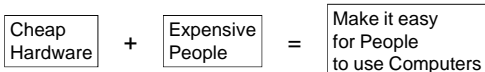
A Historical Perspective: Theme

•“Those who cannot remember the past are condemned to repeat it.” -- George Santayana (late 1800s)

•In the beginning (~1950)



• Now



CS414: Operating Systems

Problems make solutions make problems

•**Awkward Computers:** huge, expensive, expensive to run, expensive to maintain

•**Single-user, interactive:** programmers flick console switches, dump cards into card reader

•**Programmer given exclusive access**

•**Problem:** Code to manipulate external I/O devices is very complex and problematic

- Solution:** Build a subroutine library (device drivers); the library is loaded into the top of memory and stays there
- beginnings of an OS

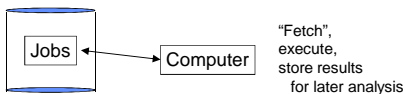
CS414: Operating Systems

Additional problems in the early days

•**Problem:** Computer idles while programmer sets things up

•**Solution:** Hire a specialized person to do setup (faster than programmer, still slower than the machine)

•**Another solution:** Build a batch monitor (stores jobs on a disk, have computer read them in one at a time and execute them without user intervention); debugging now done off-line (no more instant feedback)

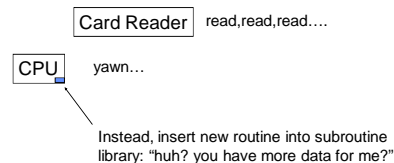


CS414: Operating Systems

Progress makes more problems

•**Problem:** At any given time, job is actively using either the CPU or an I/O device (rest of the machine is idle)

•**Solution:** Allow job to overlap computation and I/O (buffering and interrupt handling added to subroutine library)



CS414: Operating Systems

Yes, more problems

• **Problem:** One job generally can't keep both CPU and I/O devices busy

- **Solution:** multiprogramming – several jobs share the system
 - loading multiple programs into space-multiplexed memory while time-multiplexing the processor (e.g., interrupt-driven)

• **Problem:** Two executing programs can bash each other

- **Solution:** Protection and memory relocation

• **Problem:** “OS” becoming significant software system by itself

- **Solution:** ??? (if you can solve it..)

CS414: Operating Systems

Introduction of Timesharing

• **Issue/Problem:**

- Make computers easier to use, people more productive
- Batch output no longer practical, but we can't afford a computer on every desk!

• **Solution:** Interactive Timesharing (1970s)

- Interact with computer using terminal keyboard and display device
- User “logged on” and sees “virtual machine”
- Requires equitable processor sharing
- OS implements timesharing through multiprogramming

• **Problem:** Old batch schedulers run jobs to completion (or I/O ops); people need reasonable response time

- **Solution:** Preemptive scheduling!

CS414: Operating Systems

Gimme, Gimme, Gimme

• **Problem:** Users want to do more than one job while logged on

- **Solution:**
 - Process: program in execution
 - Multitasking: timesharing multiprogramming system that supports multiple processes per user

• **Problem:** People need to have data and programs around while they use the computer

- **Solution:** Create file systems (no more cards or tapes!)

• **Problem:** Boss logs on and gets lousy response time

- **Solution:** Prioritized scheduling, disk quotas, physical memory “quotas”

CS414: Operating Systems

Computers become cheaper

• **Problem:** Every timesharing system becomes overloaded

- **Solution:** Work at night, become addicted to caffeine
- Another Solution: Develop cheap computers, write OS in ROM for cheap computers without requiring isolation or sharing

• **Problem:** Minimal hardware + minimal OS (e.g., MS-DOS 1981) = minimal productivity

- **Solution:** Build cheaper memory, faster CPUs, support multitasking (replace Windows 3.1 with WinNT, WinXP, and Linux)

• **Problem:** User-as-isolated-entity doesn't work

- **Solution:** Networking becomes very important (Ethernet); make OS “sophisticated” again; (distributed OS---Goal: make collection of physically remote hardware appear as one big computer system)

CS414: Operating Systems

Operating Systems Evolution

