

CS161: Operating Systems

Prof. Matt Welsh
mdw@eecs.harvard.edu

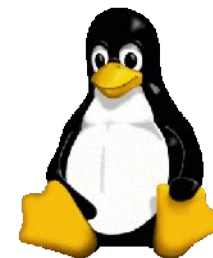


Lecture 1: Course Introduction
February 1, 2007

Welcome to CS161!

What is this course about?

- Operating Systems drive the inner workings of virtually every computer in the world today
- PCs, servers, iPods, cell phones, missile guidance systems, etc. all have an OS that dictate how they operate.
- The OS manages many aspects of how programs run, and how they interact with hardware and the outside world.



*** STOP: 0x0000001E (0x80000003,0x80106fc0,0x8025ea21,0xfd6829e8)
Unhandled Kernel exception c0000047 from fa8418b4 (8025ea21,fd6829e8)

Dll Base	Date Stamp	- Name	Dll Base	Date Stamp	- Name
80100000	2be154c9	- ntoskrnl.exe	80400000	2bc153b0	- hal.dll
80258000	2bd49628	- ncr710.sys	8025c000	2bd49688	- SCSIPORT.SYS
80267000	2bd49683	- scsidisk.sys	802a6000	2bd496b9	- Fastfat.sys
fa800000	2bd49666	- Floppy.SYS	fa810000	2bd496db	- Hpfs_Rec.SYS
fa820000	2bd49676	- Null.SYS	fa830000	2bd4965a	- Beep.SYS
fa840000	2bdaab00	- i8042prt.SYS	fa850000	2bd5a020	- SERMOUSE.SYS
fa860000	2bd4966f	- kbdclass.SYS	fa870000	2bd49671	- MOUCLASS.SYS
fa880000	2bd9c0be	- Videoprt.SYS	fa890000	2bd49638	- NCC1701E.SYS
fa8a0000	2bd4a4ce	- Vga.SYS	fa8b0000	2bd496d0	- Msfs.SYS
fa8c0000	2bd496c3	- Npfs.SYS	fa8e0000	2bd496c9	- Ntfs.SYS
fa940000	2bd496df	- NDIS.SYS	fa930000	2bd49707	- wlan.sys
fa970000	2bd49712	- TDI.SYS	fa950000	2bd5a7fb	- nbfs.sys
fa980000	2bd72406	- streams.sys	fa9b0000	2bd4975f	- ubnbf.sys
fa9c0000	2bd5bffd7	- usbser.sys	fa9d0000	2bd4971d	- netbios.sys
fa9e0000	2bd49678	- Parallel.sys	fa9f0000	2bd4969f	- serial.SYS
faa00000	2bd49739	- mup.sys	faa40000	2bd4971f	- SMBTRSUP.SYS
faa10000	2bd6f2a2	- srv.sys	faa50000	2bd4971a	- afd.sys
faa60000	2bd6fd80	- rdr.sys	faaa0000	2bd49735	- browser.sys

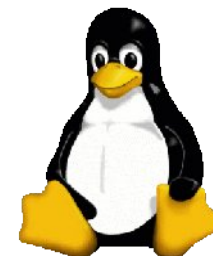
Address	dword	dump	Dll Base	- Name	
801afc20	80106fc0	80106fc0	00000000	00000000	80149905 : fa840000 - i8042prt.SYS
801afc24	80149905	80149905	ff8e6b8c	80129c2c	ff8e6b94 : 8025c000 - SCSIPORT.SYS
801afc2c	80129c2c	80129c2c	ff8e6b94	00000000	ff8e6b94 : 80100000 - ntoskrnl.exe
801afc34	801240f2	80124f02	ff8e6df4	ff8e6f60	ff8e6c58 : 80100000 - ntoskrnl.exe
801afc54	80124f16	80124f16	ff8e6f60	ff8e6c3c	8015ac7e : 80100000 - ntoskrnl.exe
801afc64	8015ac7e	8015ac7e	ff8e6df4	ff8e6f60	ff8e6c58 : 80100000 - ntoskrnl.exe
801afc70	80129bda	80129bda	00000000	80088000	80106fc0 : 80100000 - ntoskrnl.exe

Kernel Debugger Using: COM2 (Port 0x2f8, Baud Rate 19200)
Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option. If this message reappears,
contact your system administrator or technical support group.

Welcome to CS161!

What is this course about?

- Operating Systems drive the inner workings of virtually every computer in the world today
- PCs, servers, iPods, cell phones, missile guidance systems, etc. all have an OS that dictate how they operate.
- The OS manages many aspects of how programs run, and how they interact with hardware and the outside world.



Understanding the OS is essential for understanding:

- System performance and reliability
- Resource management
- Virtualization and abstraction
- Concurrency and parallelism
- Hardware interfaces and I/O



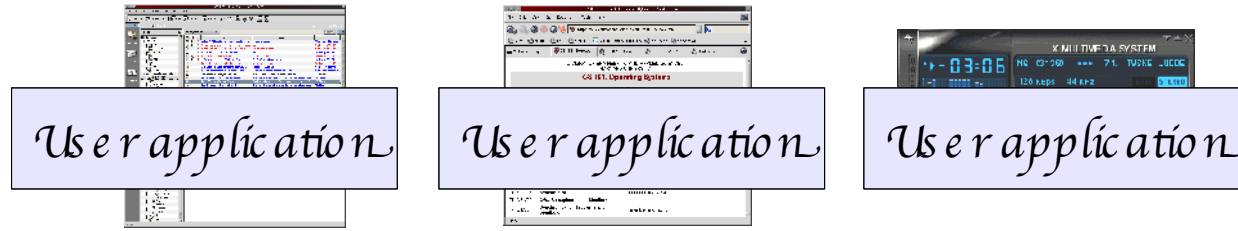
This course is about more than just “kernel internals”

- It is really about learning complex systems design.

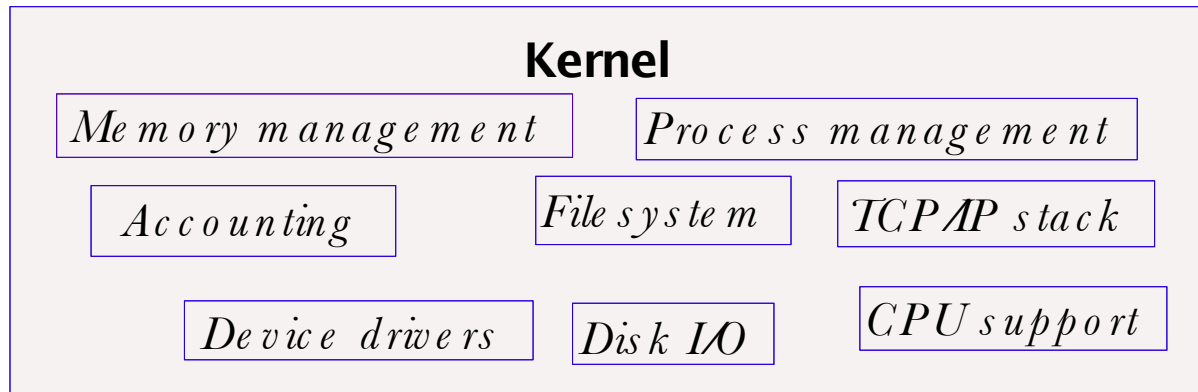


What is an operating system?

Software that provides an elaborate illusion to applications



Protection boundary

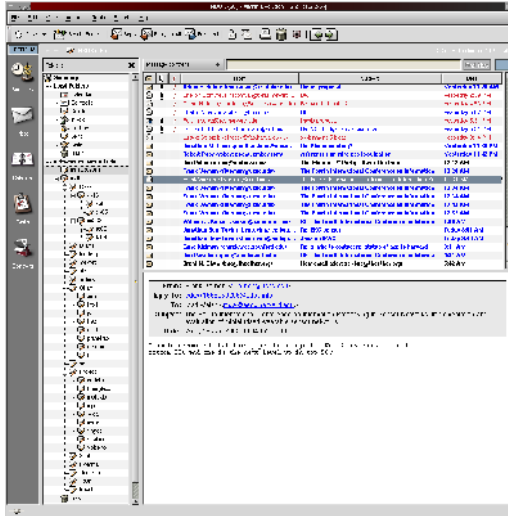


Hardware / software interface



One OS Function: Concurrency

Give every application the illusion of having its own CPU!



I think I have my own CPU

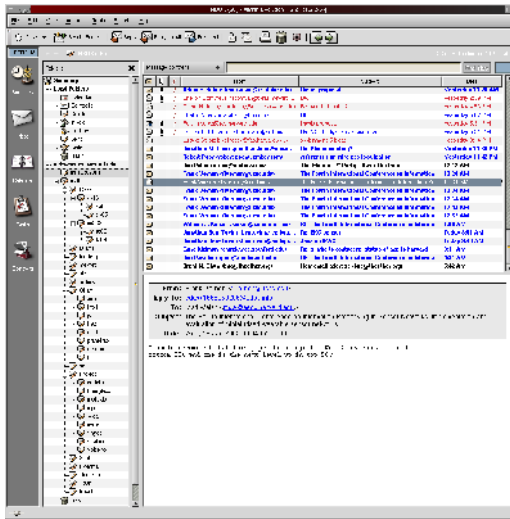
So do I



One OS Function: Concurrency

The OS *timeslices* each application on a single CPU

- Switches between applications extremely rapidly, i.e., 100 times/sec



Kernel Scheduler



time →

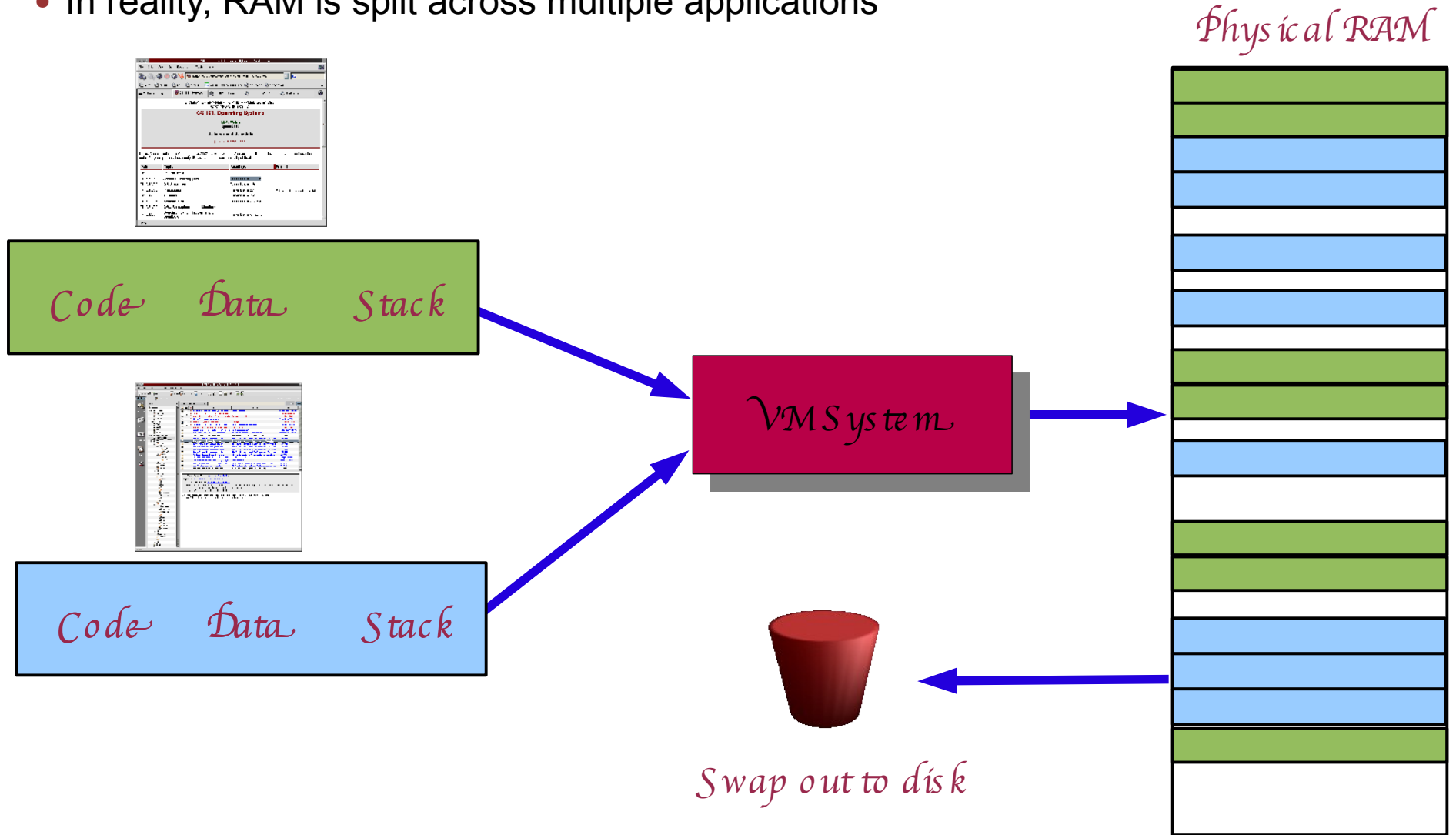
Timeslice on single CPU system



Another OS Function: Virtual Memory

Give every application the illusion of having infinite memory

- And, that it can access any memory address it likes!
- In reality, RAM is split across multiple applications



More OS Functions

Multiprocessor support

- Modern systems have multiple CPUs
- Can run multiple applications (or **threads** within applications) in parallel
- OS must ensure that memory and cache contents are consistent across CPUs

Filesystems

- Real disks have a hairy, sector-based access model
- User applications see flat files arranged in a hierarchical namespace

Network protocols

- Network interface hardware operates on the level of unreliable packets
- User apps see a (potentially reliable) byte-stream **socket**

Security and protection

- Prevent multiple apps from interfering with each other and with normal system operation

Why bother with an OS?

Not just to give Slashdot readers something to argue about...

What do you think?

-
-
-
-

Why bother with an OS?

Not just to give Slashdot readers something to argue about...

Abstract away messy details of hardware

- Give apps a nice clean view of the system
- Save programmers a lot of trouble when building applications
- Allow apps to be ported across a wide range of hardware platforms

Safety!

- Don't let applications run amok – keep them in a “sandbox”
- e.g., Access to unallocated memory address crashes only the program, not the whole system
 - `Segmentation fault – core dumped`

Efficiency

- Share one machine across many different apps: concurrent execution
- You would be surprised how much slack there is in a typical computer system

Why study operating systems?

Most people will never write one from scratch...

- *(Though if you do you stand to get incredibly rich)*
- Although more and more people are hacking them (e.g., Linux and BSD)
- You need to understand the “big picture” in order to hack the details
- Lots of examples of fundamentally broken Linux kernel drivers, probably written by people who never took CS161 :-)

This class is about much more than the kernel!

- Data structures, concurrency, performance, resource management, synchronization, networks, distributed systems, databases ...
- The ideas and skills you pick up in this class have broad applications
 - *And it doesn't hurt for those Microsoft interviews either*

This course is the basis for future work in other areas of systems

- Distributed systems, P2P, sensor nets, etc.

In the Beginning...



There was no OS – just libraries

- Computer only ran one program at a time, so no need for an OS

And then there were **batch systems**

- Programs printed on stacks of punchhole cards
- OS was resident in a portion of machine memory
- When previous program was finished, OS loaded next program to run



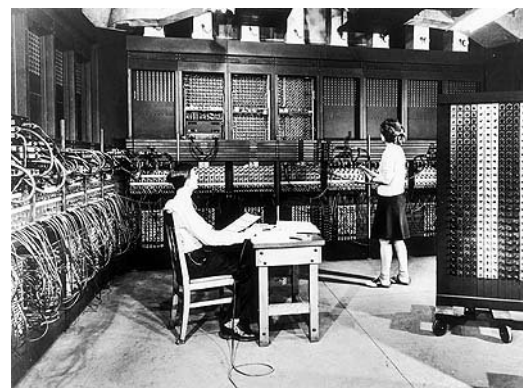
Disk spooling

- Disks were much faster than punchcards – read stack onto disk while previous program is running
- With multiple programs on disk, need to decide which to run next!
- But, CPU still idle while program accesses a peripheral (e.g., tape or disk!)

Harvard Mark I, 1944



ENIAC, 1945



*IBM 360,
1960's*

Multiprogramming

To increase system utilization, **multiprogramming** OS's were invented

- keeps multiple runnable jobs loaded in memory at once

Overlaps I/O of a job with computing of another

- While one job waits for I/O to complete, CPU runs instructions from another job
- To benefit, need **asynchronous** I/O devices
 - need some way to know when devices are done performing I/O
- Goal: optimize system throughput
 - perhaps at the cost of response time...



*Dennis Ritchie and Ken
Thompson at a PDP11, 1971*

Timesharing

To support interactive use, **timesharing** OS's were created

- multiple terminals connected to one machine
- each user has illusion of entire machine to him/herself
- optimize response time, perhaps at the cost of throughput

Timeslicing

- divide CPU fairly among the users
- if job is truly interactive (e.g. editor), then can switch between programs and users faster than users can generate load

MIT Multics (mid-1960's) was the first large timeshared system

- nearly all modern OS concepts can be traced back to Multics



Personal Computing

Totally changed the computing industry.

CP/M: First personal computer OS

- IBM needed OS for their PCs, CP/M behind schedule
- Bill Gates to the rescue: Bought 86-DOS and made MS-DOS
 - *DOS is basically a subroutine library!*

Many popular personal computers follow

- Apple, Commodore, TRS-80, TI 99/4, Atari, etc...
 - *Those were the days ...*

Apple I, 1976



Apple Lisa, 1983



IBM PC, 1981



*Commodore VC-20
(-my first computer)*



Bill Gates and Paul Allen, c.1975



Photo courtesy of Microsoft Archives.

Apple Introduces the First Low Cost Microcomputer System with a Video Terminal and 8K Bytes of RAM on a Single PC Card.

The Apple Computer. A truly complete microcomputer system on a single PC board. Based on the MOS Technology 6502 microprocessor, the Apple also has a built-in video terminal and sockets for 8K bytes of on-board RAM memory. With the addition of a keyboard and video monitor, you'll have an extremely powerful computer system that can be used for anything from developing programs to playing games or running BASIC.

Combining the computer, video terminal and dynamic memory on a single board has resulted in a large reduction in chip count, which means more reliability and lowered cost. Since the Apple comes fully assembled, tested & burned-in and has a complete power supply on-board, initial set-up is essentially "hassle free" and you can be running within minutes. At \$666.66 (including 4K bytes RAM!) it opens many new possibilities for users and systems manufacturers.

You Don't Need an Expensive Teletype.

Using the built-in video terminal and keyboard interface, you

avoid all the expense, noise and maintenance associated with a teletype. And the Apple video terminal is six times faster than a teletype, which means more throughput and less waiting. The Apple connects directly to a video monitor (or home TV with an inexpensive RF modulator) and displays 960 easy to read characters in 24 rows of 40 characters per line with automatic scrolling. The video display section contains its own 1K bytes of memory, so all the RAM memory is available for user programs. And the Keyboard Interface lets you use almost any ASCII-encoded keyboard.

The Apple Computer makes it possible for many people with limited budgets to step up to a video terminal as an I/O device for their computer.

No More Switches, No More Lights.

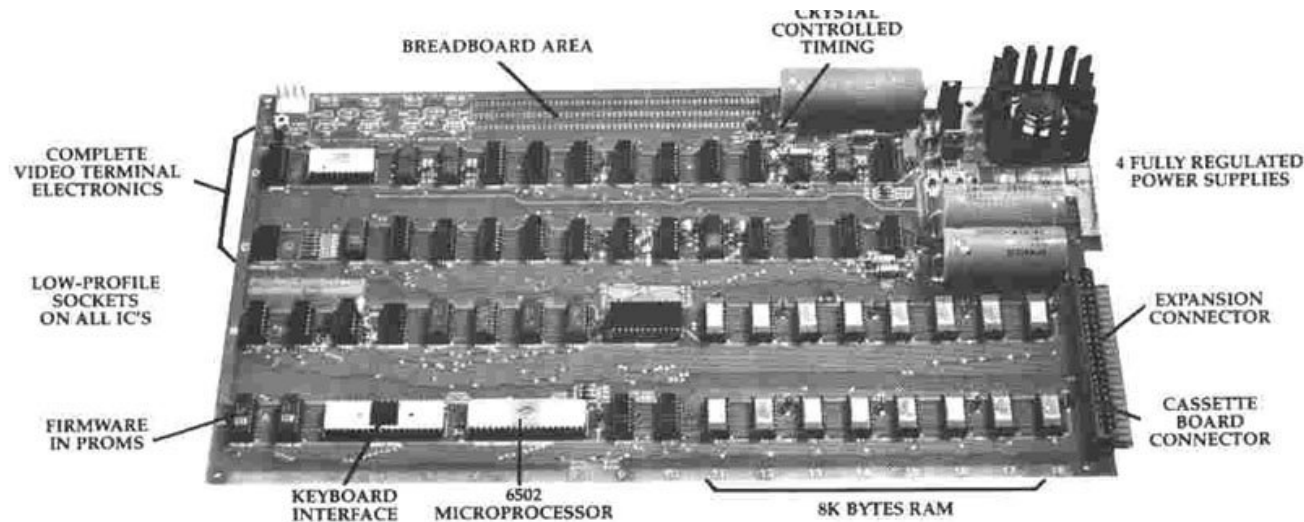
Compared to switches and LED's, a video terminal can display vast amounts of information simultaneously. The Apple video terminal can display the contents of 192 memory locations at once on the screen. And the firmware in PROMS enables you to enter,

display and debug programs (all in hex) from the keyboard, rendering a front panel unnecessary. The firmware also allows your programs to print characters on the display, and since you'll be looking at letters and numbers instead of just LED's, the door is open to all kinds of alphanumeric software (i.e., Games and BASIC).

8K Bytes RAM in 16 Chips!

The Apple Computer uses the new 16-pin 4K dynamic memory chips. They are faster and take $\frac{1}{4}$ the space and power of even the low power 2102's (the memory chip that everyone else uses). That means 8K bytes in sixteen chips. It also means no more 28 amp power supplies.

The system is fully expandable to 65K via an edge connector which carries both the address and data busses, power supplies and all timing signals. All dynamic memory refreshing for both on and off-board memory is done automatically. Also, the Apple Computer can be upgraded to use the 16K chips when they become available. That's 32K bytes on-board RAM in 16 IC's —the equivalent of 256 2102's!



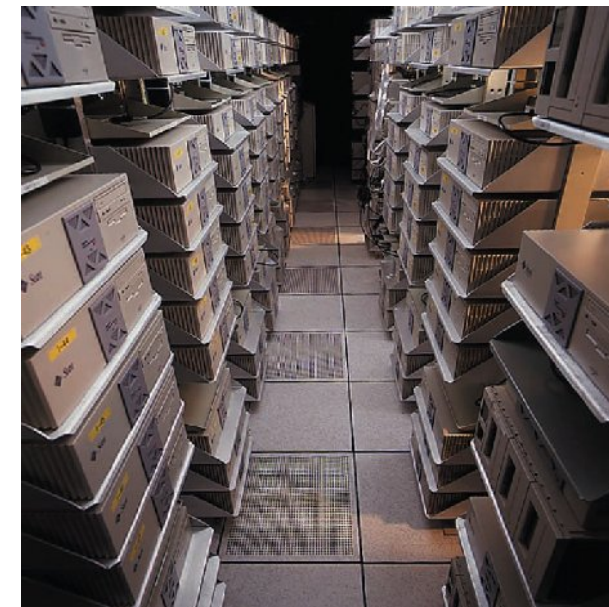
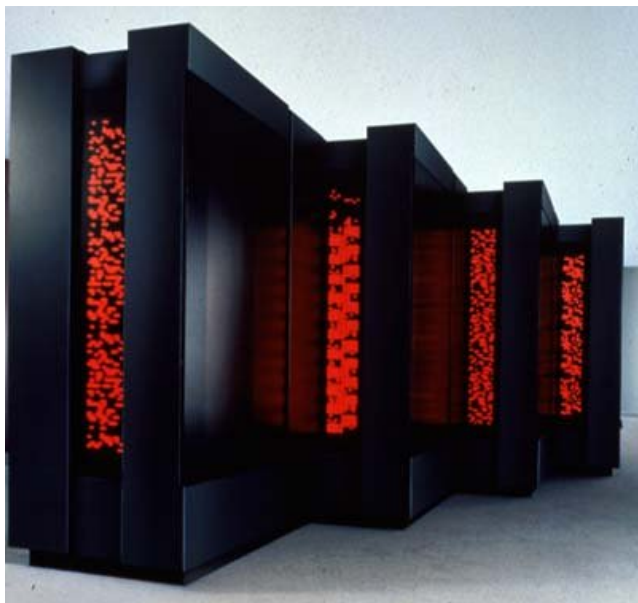
Parallel Computing and Clusters

High-end scientific apps want to use many CPUs at once

- Parallel processing to crunch on enormous data sets
- Need OS and language primitives for dividing program into parallel activities
- Need OS primitives for fast communication between processors
 - *degree of speedup dictated by communication/computation ratio*

Many kinds of parallel machines:

- SMPs: symmetric multiprocessors – several CPUs accessing the same memory
- MPPs: massively parallel processors – each CPU may have its own memory
- Clusters: connect a lot of commodity machines with a fast network



Distributed OS

Goal – Make use of geographically distributed resources

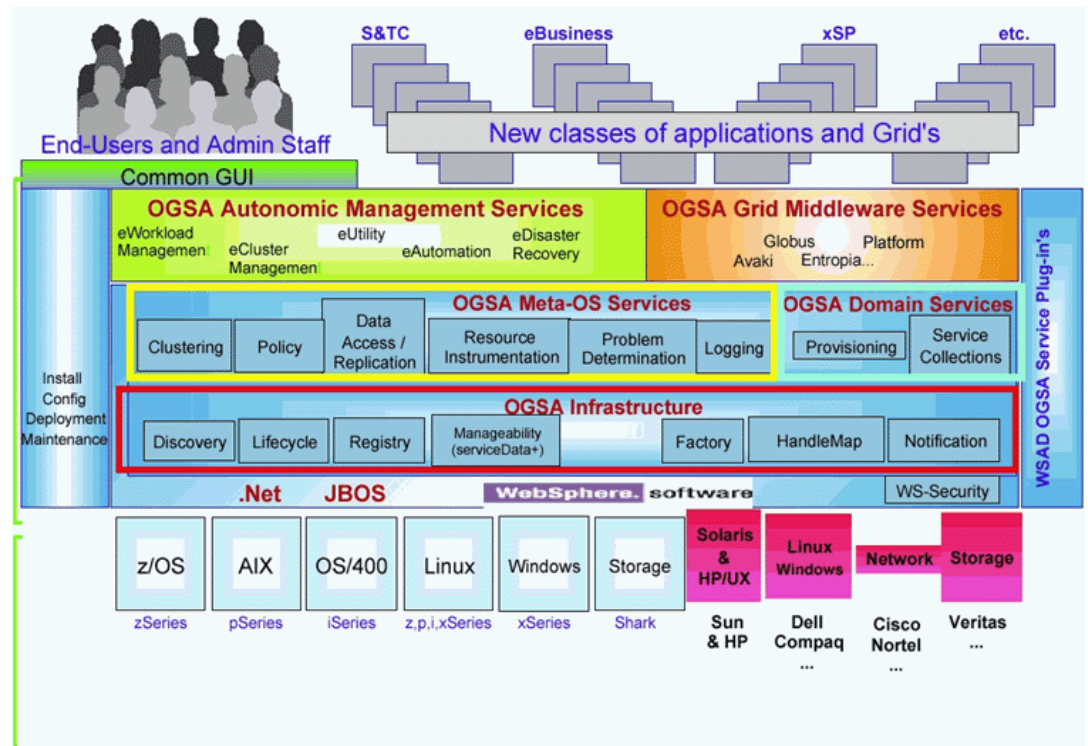
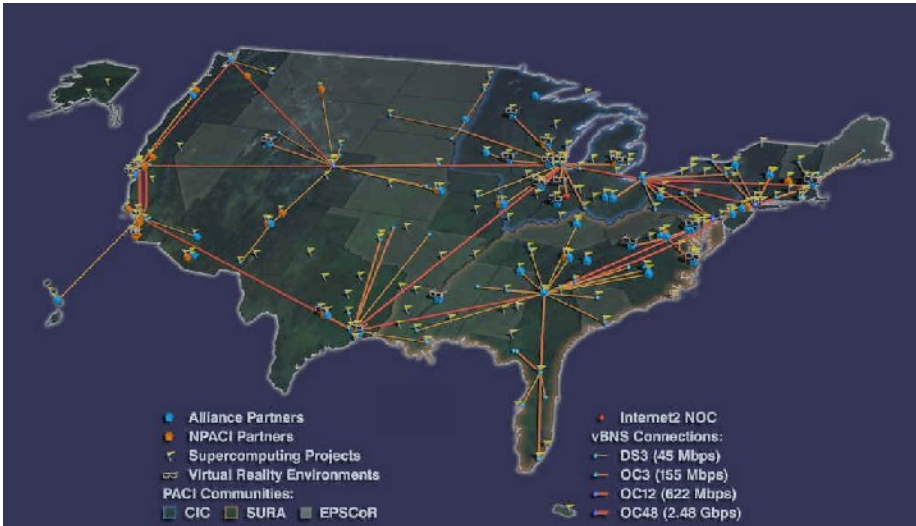
- workstations on a LAN
- servers across the Internet

Supports communication between applications

- interprocess communication (on a single machine):
 - *message passing and shared memory*
- networking protocols (across multiple machines):
 - *TCP/IP, Java RMI, .NET SOAP*

“The Grid”, .NET, and OGSA

- Idea: Seamlessly connect vast computational resources across the Internet



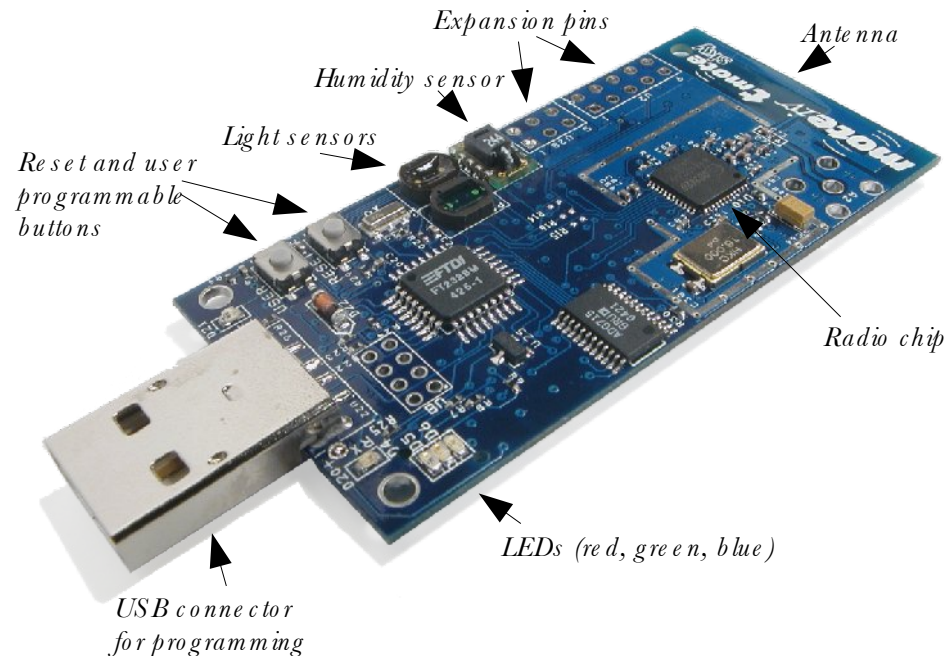
Mobile and Embedded Devices

The rise of tiny computers everywhere – ubiquitous computing

- Processor cost low enough to embed in many devices
 - *PDA's, cell phones, pagers, ...*
- How many CPUs are in your car? On your body right now?

Very different hardware capabilities...

- slow processors
- small amount of memory (e.g. 8 MB)
- no disk – but maybe quasi-permanent storage such as EEPROM
- Various wireless networks (cellular, Bluetooth, 802.11)



What you will do in this course

Learn about the wonderful world of OS design

- Lectures cover concepts and ideas
- Assignments give you hands-on experience

Build major pieces of your own operating system — OS/161

- You will implement: system calls, scheduler, virtual memory system, filesystem

Earn the right to call yourself a kernel hacker

- I know Linus ... so I will back you up on this one



The Work Factor

CS161 is not the easiest course at Harvard...

- But it is not the hardest, either

We are actively working to **reduce** the workload in the course

- We have reduced major aspects of each assignment
- Focus more on having fun and doing some hacking
- This is an experiment!

The Fine Print

- Building an OS is still a major undertaking
- You will be programming in C on UNIX systems, on top of a hardware emulator
- You will encounter bugs that you have never seen before

Syllabus

- Processes and threads
- Synchronization, semaphores, monitors
 - *Assignment 1: Synchronization*
- CPU scheduling policies
 - *Assignment 2: System calls and scheduling*
- Virtual memory, paging, and TLBs
- Case study: The Linux virtual memory system
- **Midterm and spring break**
 - *Assignment 3: Virtual memory*
- Disks, filesystems, FFS, and LFS, and Linux ext3fs filesystem
 - *Assignment 4: File systems*
- RAID and NFS filesystems
- Security and protection
- OS design for multiprocessor and multicore systems
- Network protocol stack design
- Case study: Internet service design
- Microkernels
- Alternative OS designs: SPIN and Exokernel
- Distributed operating systems
- **Take-home final exam**

Grading

50% — Programming Assignments

- 5 assignments in total
- Graded on design document, code structure, and whether your code works

15% — In-class midterm (March 22, before spring break)

- T/F, multiple choice, and short answer questions
- Focuses on OS concepts and problem solving (not OS/161 code!)

25% — Take-home (24-hour) final exam

- Much like the midterm but you get more time.

10% — Class participation

- Coming to lectures and participating in the discussion is expected!

Most people who take this course do very well (lots of A's).

- We tend to be lenient on the grading...

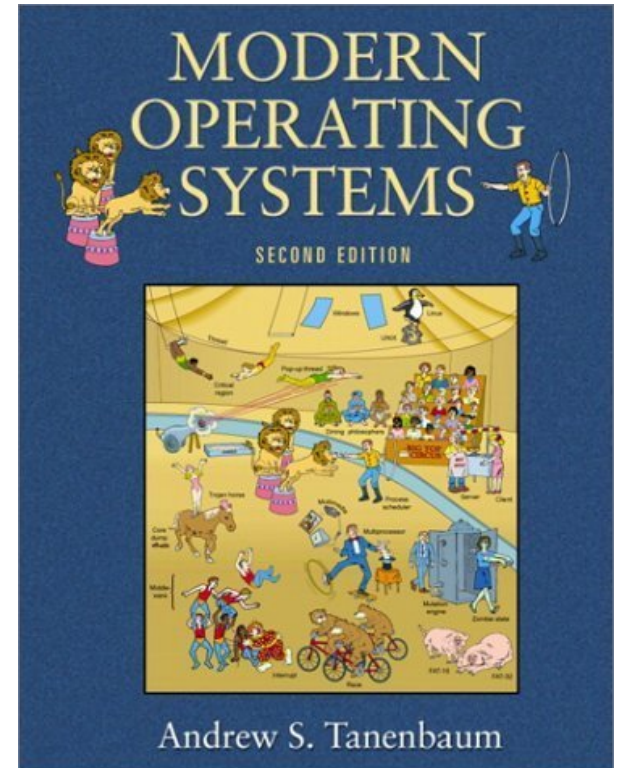
Textbook and Website

Required Text: Modern Operating Systems,
2nd Ed., by Tanenbaum

- Available at Amazon and other fine bookstores
- Book is *essential* for understanding course concepts
- Especially useful for midterm and final exams!

Course website:

- <http://www.eecs.harvard.edu/~mdw/course/cs161>
- Syllabus, lecture notes, assignments, etc.
- Check frequently for updates!



Course Staff

Instructor: Matt Welsh (mdw@eecs.harvard.edu)

- Office hours: Thursdays, 9:30-11:30 – **no appt. necessary**
- Office Maxwell Dworkin 233

Teaching staff



*Geoff Werner-Allen
(werner@eecs)*



*Andrew McCollum
(mccollum@fas)*



*Steve Dawson-Haggerty
(sdawson@fas)*

Contact info

- E-mail to cs161@fas for all course questions, comments, etc.

Sections and Office Hours

Sections

- 2 sections a week, times TBD
- Sections are critical – not everything will be covered in lecture!
- Sections will serve two purposes:
 - *Elaborate on lecture material and go into in-depth examples*
 - *Discuss details of OS/161 system and assignments*

First Section will be scheduled next week (TBA)

- Overview of OS/161, toolchain, debugging with GDB, other juicy tidbits

TF Office Hours

- TF's will be on hand to answer questions on code and assignments
- Location: Computer lab in the Science Center basement
 - *(starting in a couple of weeks)*