

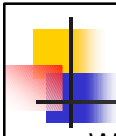
Operating Systems

Arvind Krishnamurthy
Spring 2004



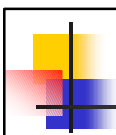
Today's lecture

- What is an operating system?
- History of operating systems
- Course overview
 - course information
 - schedule, assignments, grading and policy
 - other organization issues
 - see web pages for more information
- Why study operating systems?



Discussion

- What is an operating system?
 - A program that acts as an intermediary between users and the hardware
- What is an operating system supposed to do?
 - What tasks do we expect an OS to perform?
 - What needs to be part of an OS?
- Is "emacs" an operating system?
- Is "windows 3.x/DOS" an operating system?
- Should shell/x-windows be part of an OS?



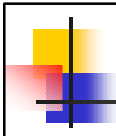
Imagine this...

- You have the raw hardware (processor with memory, disk, network, etc.)
- Take a program which looks like:

```
int array[SI ZE];

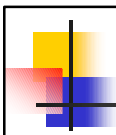
main() {
    int sum = 0, i = 0;

    while (i < count) {
        sum += array[f(i)];
        i += g(i);
    }
}
```
- What could go wrong?



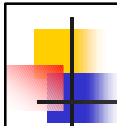
Different Roles of an OS

- OS as a police state:
 - Isolate programs
 - Make sure that no program gets too much of any resource
 - Make sure that programs do not corrupt resources
- Techniques:
 - Preemption: take away resources after a while
 - Use interrupts to stop programs after a certain period of time or when interesting things happen
 - Interposition: OS between application and stuff
 - For example, on every memory access check whether that access is legal
 - Dual mode operation: allow OS to do interesting things when CPU is running in "kernel mode"



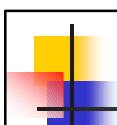
Different Roles of an OS

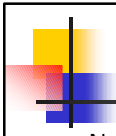
- OS as a welfare state:
 - Provide the abstraction of "unlimited virtual memory"
 - Create programs with almost unlimited data segments
 - OS will store some of it in memory, some in disk, and page in appropriately
 - Provide a file system abstraction
 - Hierarchical arrangement of files and directories
 - Provide protection
 - Support networking, windowing, etc.



OS History: Change

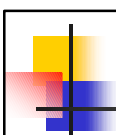
	1981	2003	factor
CPU MHz	1	3000	3000
DRAM capacity	128KB	1GB	8000
Disk capacity	10MB	400GB	40000
Net bandwidth	10Kb/s	155Mb/s	15000
# address bits	16	64	4
# users/machine	>10	<=1	0.1
Price	\$25K	\$2K	12

- 
- ## History Phase 1
- Hardware expensive, humans cheap
 - When computers cost millions of \$s, optimize for more efficient use of the hardware
 - One user at a time. OS as subroutine library (referred to as "batch monitor")
 - Batch monitor (which is preloaded in memory) repeats:
 - compile program input as punched card
 - load binary to some fixed location in memory
 - run for predefined duration, print results
 - What are the problems associated with this approach?



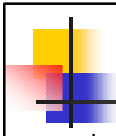
Memory Protection, Relocation

- Next step enabled by:
 - Memory protection: prevent users from interfering with each other/OS
 - Relocation: load program at any memory location, execute
- Multiprogramming: several programs run at the same time
 - Small jobs not delayed by large jobs
 - Overlap between I/O and CPU
 - OS requests I/O, goes back to a different job, waits for interrupt
- Bad news: OS must manage all these; each step logical, but gets complicated at some point
- Multics: announced in 1963, ran in 1969
- OS 360 released with 1000 bugs
- Unix based on Multics, but simplified to get it to work!



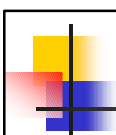
The Unix Story

- Bell Labs was one of the players in Multics
- Ken Thompson was working on a "Space Travel" game program; ran on a GE 635 m/c at a cost of \$75 in CPU time; started porting it to an unused PDP-7 in 1969
 - Developed a small file system, ported the assembler, developed useful utilities (file copy, etc.)
 - Ritchie developed C language, Ken ported Unix to C
 - Developed shell, bunch of composable tools
 - Went to Berkeley, taught a course, and the rest is history
- Implementation details:
 - PDP-11/45
 - 16-bit word, 144KB of core memory
 - 1MB disk, four 2.5MB removable disks
 - Unix occupies: 42KB ("very large number of device drivers and enjoys a generous allotment of space for I/O buffers")



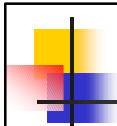
History Phase 2

- Interactive timesharing:
 - Use cheap terminals to let multiple users interact with the system at the same time
 - Sacrifice CPU time to get better response time for users
 - Multiple programs are active at a given point in time
 - What is the pitfall associated with this?
- At some point: hardware became cheap, humans expensive
- Personal computing:
 - Computers are cheap, so give everyone a computer
 - OS became subroutine again, but since then, have added back in memory protection and multiprogramming
- Networking:
 - Allow multiple computers to look like a single entity



This Course

- **13 week lectures on OS fundamentals**
- **Course requirements (subject to minor changes)**
 - 60% on assignments (as0 is easy), paper reviews
 - 15% open-book, in-class midterm
 - 25% open-book final
 - Alternative: semester-long project
- **Assignments (as1-as4) and course policies**
 - extend and improve Nachos --- an instructional OS
 - extensive hacking in "clean" C++
 - 3 persons / team; 7 free late days (4 per assignment max)
 - Assignment mechanics will be announced soon (design reviews)



Course Information

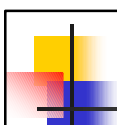
Textbooks: Applied Operating System Concepts by SGG (recommended but not required)

Readings consist of:

- Technical papers (about 10-15; supplements class discussion: no reviews required)
- Advanced papers (about 10-15): turn in reviews before lecture
- Background readings

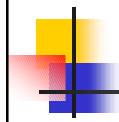
Official URL: <http://zoo.cs.yale.edu/classes/cs422>
[<http://lambda.cs.yale.edu/cs422>]

For help, send email to instructor/TA.



Why study operating systems?

- Abstraction:
 - Unlimited processing power, infinite storage, single entity
- System Design:
 - Tradeoffs between performance and functionality, division of labor between HW and SW
- Understand how computers work:
 - Combine language, hardware, data structures, algorithms
- Give you experience in hacking systems software
 - "can I add some new functionality to an existing system?"
 - "this system is so slow, can I do anything about it?"



Course Map

- Broad outline of the course:
 - Threads and concurrency
 - Next two weeks
 - Virtual Memory
 - File Systems
 - Networks
 - Security
- Wednesday: introduction to threads & processes