# Introduction to Operating Systems and Practicum in Operating Systems

COS 414/415

Spring 2002

Prof. Emin Gün Sirer

# Administrative

- Instructors:
  - Prof. Emin Gün Sirer, egs@cs, 4119A Upson, MF 1:25-2:15
- Communication
  - cs414@cs.cornell.edu
- TAs:
  - Alin Dobra, dobra@cs
  - Hongzhiu Liu, liuhz@cs
  - Joseph Aliperti, jra12@cs
  - Vincent Eng, ve14@cs
  - Devon Welles, dmw22@cs
  - Stephen Enochson, sre6@cs
- All official information is on the web site
  - http://www.cs.cornell.edu/courses/cs414/2002SP/

# Administrative

- 414: Intro to Operating Systems
  - Fundamentals of OS design
- Textbook
  - Silberschatz & Galvin, Operating System Concepts, 6th Edition,
- Reading, weekly homeworks
- 415: Practicum in Operating Systems
  - Major programming assignment
  - This year, we’ll use PDAs for the project
  - May work in pairs

# Grading

- Course prerequisite: Mastery of the material in CS 314, computer architecture
- 414: Intro to Operating Systems
  - Reading Assignments (~10%)
  - Midterm (~30%)
  - Final (~50%)
  - Subjective criteria (~10%)
- 415: Practicum in Operating Systems
  - Six projects (100%)
- This is a rough guide

# Academic Integrity

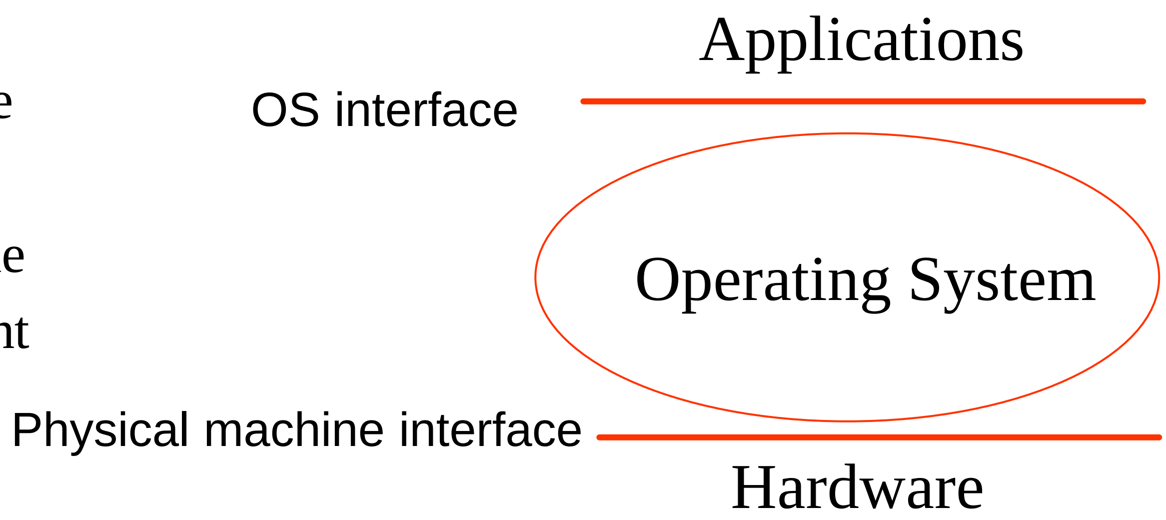
- Everything you turn in must be your own work
- Certain types of collaboration are a part of the learning experience
  - May consult with others on C syntax, problem clarification, debugging strategies, etc.
  - May NOT be in possession of someone else's homework or project, may NOT plagiarize answers to homework questions, may NOT copy code, etc.
  - The academic integrity guidelines provide the general ground rules
- Dishonesty has no place in any community
  - The penalty is an immediate F in 414 and 415

# Course Outline

- History, architectural support
- Concurrency, processes, threads
- Synchronization, monitors, semaphores, condition variables, mutual exclusion
- Networking, distributed systems
- Memory Management, virtual memory
- Storage Management, I/O, filesystems
- Security
- Case studies

# What is an Operating System?

- Definition: An Operating System (OS) provides a virtual machine on top of the hardware that is more convenient than the raw hardware interface
  - "All of the code you did not write"
  - Simpler
  - More reliable
  - More secure
  - More portable
  - More efficient
  - …



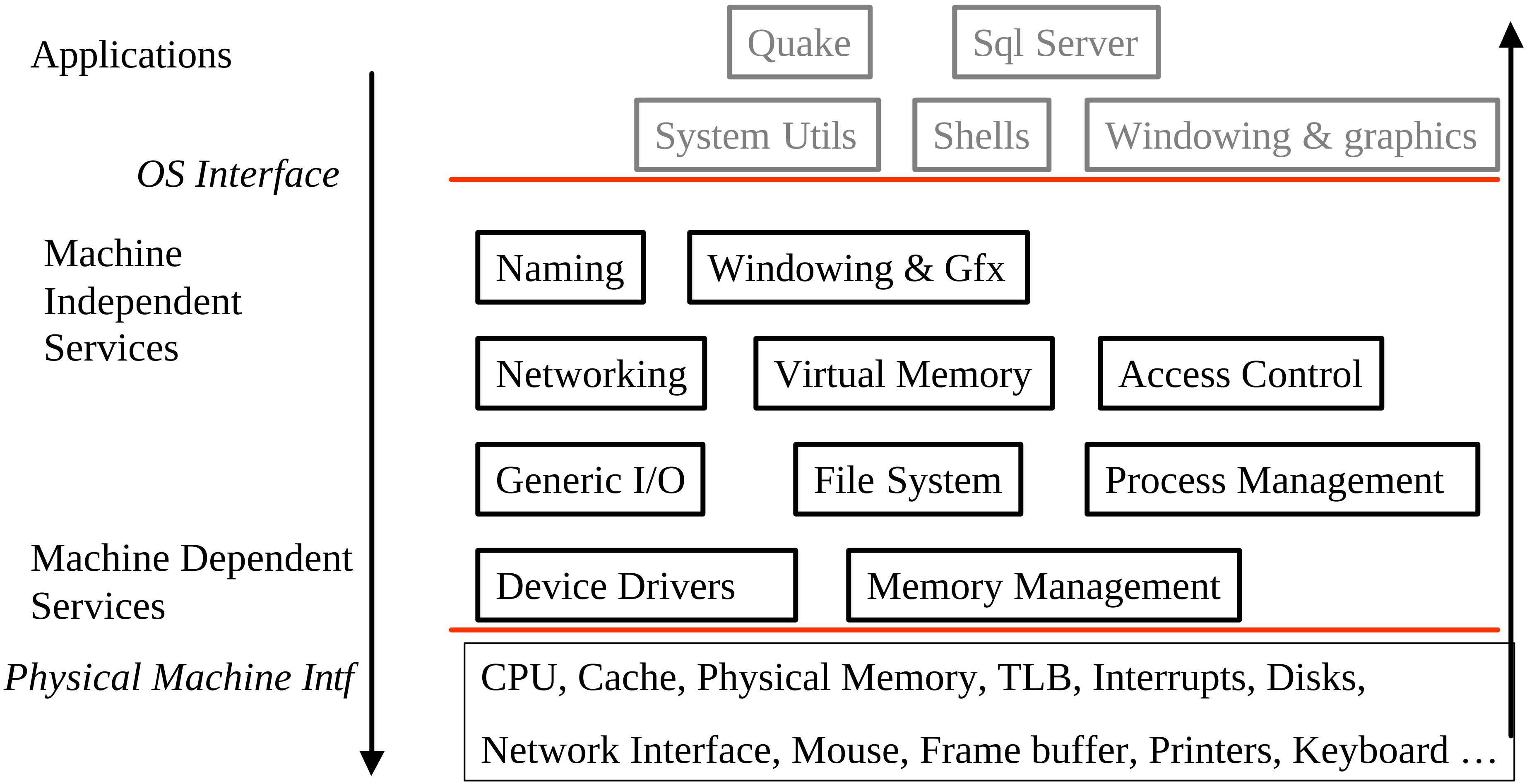
# What do Operating Systems Do ?

- Manage physical and virtual resources
- Provide users with a well-behaved environment
- Define a set of logical resources (objects) and a set of well-defined operations on those resources (i.e. an interface to those objects)
- Provide *mechanisms* and *policies* for the control of resources
- Control how different users and programs interact

# What Resources Need to Be Managed?

- The CPU(s)
- Memory
- Storage devices (disks, tapes, etc)
- Networks
- Input devices (keyboard, mouse, cameras, etc.)
- Output devices (printers, displays, speakers, etc.)

# What’s in an OS?




Logical OS Structure

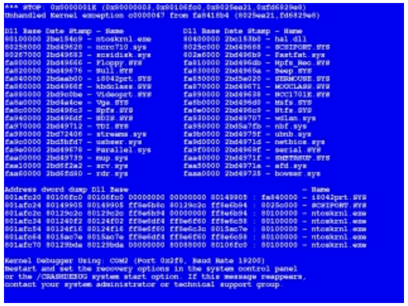
# Major Issues in Operating Systems

- **Structure --** how is an operating system organized ?
- **Concurrency --** how are parallel activities created and controlled ?
- **Sharing --** how are resources shared among users ?
- **Naming --** how are resources named by users or programs ?
- **Protection --** how is one user/program protected from another ?
- **Security --** how to authenticate, control access, and secure privacy ?
- **Performance --** why is it so slow ?
- **Reliability and fault tolerance –** how do we deal with failures ?
- **Extensibility --** how do we add new features ?
- **Communication --** how can we exchange information ?

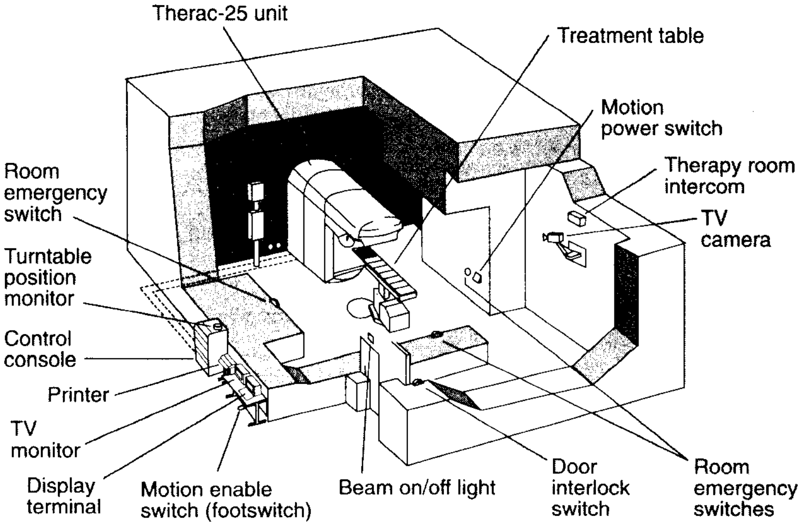
# Major Issues in OS (2)

- **Scale  and growth --** what happens as demands or resources increase ?
- **Persistence --** how to make data outlast the processes that created them
- **Compatibility --** can we ever do anything new ?
- **Distribution --** accessing the world of information
- **Accounting --** who pays the bills, and how do we control resource usage?

# Why is this material critical ?



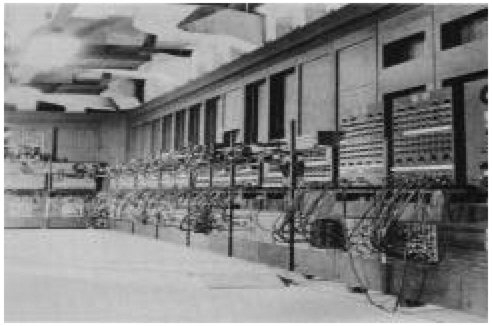
- Concurrency: Therac-25, Shuttle livelock
- Persistence: Denver airport
- Communication: Air traffic control system
- Virtual Memory: BSOD
- Security: IRS



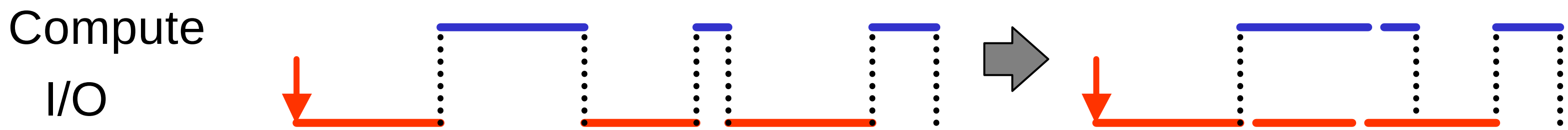
# Therac-25

- Software engineers might insist that it was the development process that failed
- In reality, people died because a programmer could (or did) not implement proper semaphores
    - They did not use well-defined synchronization primitives
- This class will ensure that you will become better engineers than the people involved in these incidents

# A Brief History of Operating Systems

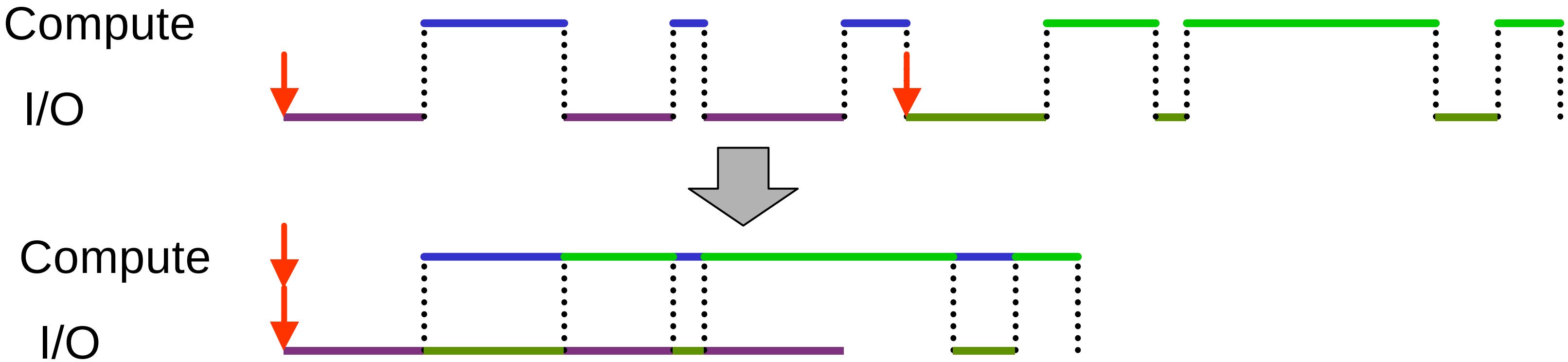


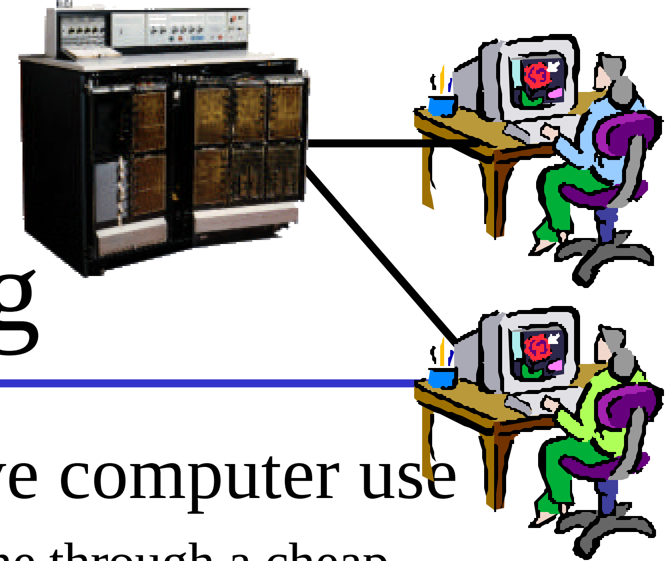
- Initially, the OS was just a run-time library
  - You linked your application with the OS, loaded the whole program into memory, and ran it
  - How do you get it into the computer ? Through the control panel!
- Simple batch systems
  - Permanently resident OS in primary memory
  - It loaded a single job from card reader, ran it, and loaded the next job...
  - *Control cards* in the input file told the OS what to do
  - *Spooling* allowed jobs to be read ahead of time onto tape/disk or into memory



# Multiprogrammed BatchSystems

- *Multiprogramming* systems provided increased utilization
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computation of another
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA
  - Optimizes for *throughput* at the cost of *response time*



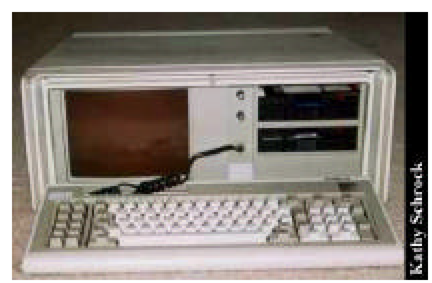


# Timesharing

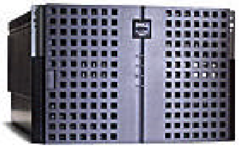
- *Timesharing* supported interactive computer use
  - Each user connects to a central machine through a cheap terminal, feels as if she has the entire machine
  - Based on time-slicing -- dividing CPU equally among the users
  - Permitted active viewing, editing, debugging, participation of users in the execution process
  - Security mechanisms required to isolate users from each other
  - Requires memory protection hardware for isolation
  - Optimizes for *response time* at the cost of *throughput*

# Personal Computing

- Computers are cheap, so give everyone a dedicated computer
- Initially, the OS became a library again due to hardware constraints
- Multiprogramming, memory protection, and other advances were added back
  - For entirely different reasons

# Parallel Operating Systems



- Support parallel applications wishing to get speedup of computationally complex tasks
- Needs basic primitives for dividing one task into multiple parallel activities
- Supports efficient communication between those activities
- Supports synchronization of activities to coordinate sharing of information
- It's common now to use networks of high-performance PCs/workstations as a parallel computer

# Distributed Operating Systems

- Distributed systems facilitate use of geographically distributed resources
  - Machines connected by wires, no shared memory or clock
- Supports communication between parts of a job or different jobs
  - Interprocess communication
- Sharing of distributed resources, hardware and software
  - Resource utilization and access
- Permits some parallelism, but speedup is not the issue

# Real-time Operating Systems

- Goal: To cope with rigid time constraints
- Hard real-time
  - OS guarantees that applications will meet their deadlines
  - Examples: TCAS, health monitors, factory control, etc.

- Soft real-time
  - OS provides prioritization, on a best-effort basis
  - No deadline guarantees, but bounded delays
  - Examples: most electronic appliances


- Real-time means “predictable”
  - NOT fast

# Ubiquitous Computing

- The decreased cost of processing makes it possible to embed computers everywhere.  Each "embedded" application needs its own control software:
  - PDAs, cell phones, intelligent appliances, etc.
- In the near future, you will have 100s of these devices
  - If not already
- Poses lots of problems for current systems
  - Structure, naming, scaling, security, etc.
- We will tackle some of them in this class

# Lessons from History

- The point is not that batch systems were ridiculous
  - They were exactly right for the tradeoffs at the time
- The tradeoffs change

| | 1981 | 2001 | Factor |
|---|---|---|---|
| MIPS | 1 | 1000 | 1000 |
| $/MIPS | $100000 | $5000 | 20000 |
| DRAM | 128KB | 256MB | 2000 |
| Disk | 10MB | 80GB | 8000 |
| Net Bandwidth | 9600 b/s | 100 Mb/s | 10000 |
| # Users | >> 10 | <= 1 | 0.1 |

- Need to understand the fundamentals
  - So you can design better systems for tomorrow’s tradeoffs

# COS 414/415

- In this class we will learn:
  - What the parts of an OS are
  - How the OS and each sub-part is structured
  - What the important mechanisms are
  - What the important policies are
  - What algorithms are typically used
- We will do this through reading, lectures, and a project
  - Project will involve some aspect of ubiquitous computing using HP Jornada 720's and Palmax @migo 600's equipped with Aeronet cards
  - Reading: Chapters 1 & 2
- You will need to keep up with all three of these