COS 318: Operating Systems

Introduction

Kai Li
Computer Science Department
Princeton University

(http://www.cs.princeton.edu/courses/cs318/)

Today

- Administrative Issues
- What is operating system?
- Why study operating systems?
- What is in COS318

Help

- Instructors
  - Kai Li, 321 CS Building, li@cs.princeton.edu
    Office hours: Tue 3-5pm
  - Several faculty members to give lectures
- Teaching Assistants
  - Nick Johnson (Projects 1-5)
  - Shi Li (Projects 1-5)
  - Lars A. Bongo (Final project)
- Information
  - Website: http://www.cs.princeton.edu/courses/cos318
  - **Subscribe to cos318@lists.cs.princeton.edu**

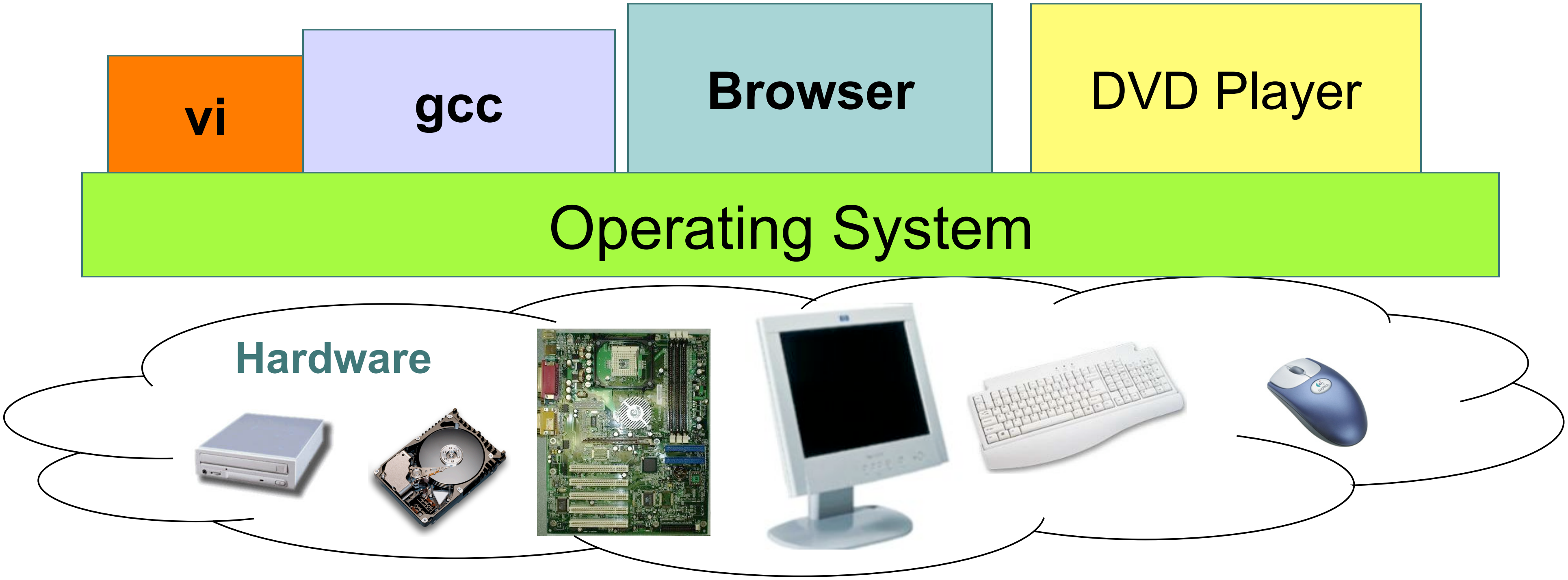
Resolve “TBD”

- Precept
  - Time: Tue and Wed 8:30pm – 9:30pm
  - Location: default is this room
- Special tutorial on assembly programming and kernel debugging
  - 9/21: 7:30-8:30pm
- Design review
  - Monday evening (signup sheet)

COS318 in Systems Course Sequence

- Prerequisites
  - COS 217: Introduction to Programming Systems
  - COS 226: Algorithms and Data Structures
- 300-400 courses in systems
  - **COS318: Operating Systems**
  - COS320: Compiler Techniques
  - COS333: Advanced Programming Techniques
  - COS432: Information Security
  - COS471: Computer Architecture
- Courses needing COS318
  - COS 461: Computer Networks
  - COS 518: Advanced Operating Systems
  - COS 561: Advanced Computer Networks

What Is Operating System?



vi

gcc

Browser

DVD Player

Operating System

Hardware

- Software between applications and hardware
- Make finite resources "infinite"
- Provide protection and security

What Do Operating Systems Do?

- System construction
  - Raw hardware devices are not usable
  - Make hardware usable
  - Make unreliable components reliable
- Protection
  - Simple OS is inefficient
  - Enable to run multiple applications safely
  - Mechanisms to prevent applications from crashing a system?
- Resource management
  - Resources are always limited
  - Make finite CPU, memory and I/O “infinite”
  - Make resource allocation fair

Some Examples

- System example
  - What if a user tries to access disk blocks?
  - What if a network link is noisy
- Protection example
  - What if a program starts randomly accessing memory?
  - What if a user tries to push the system limit?

```
int main() {
        while(1)
                fork();
}
```

- Resource management example
  - What if many programs are running infinite loops?

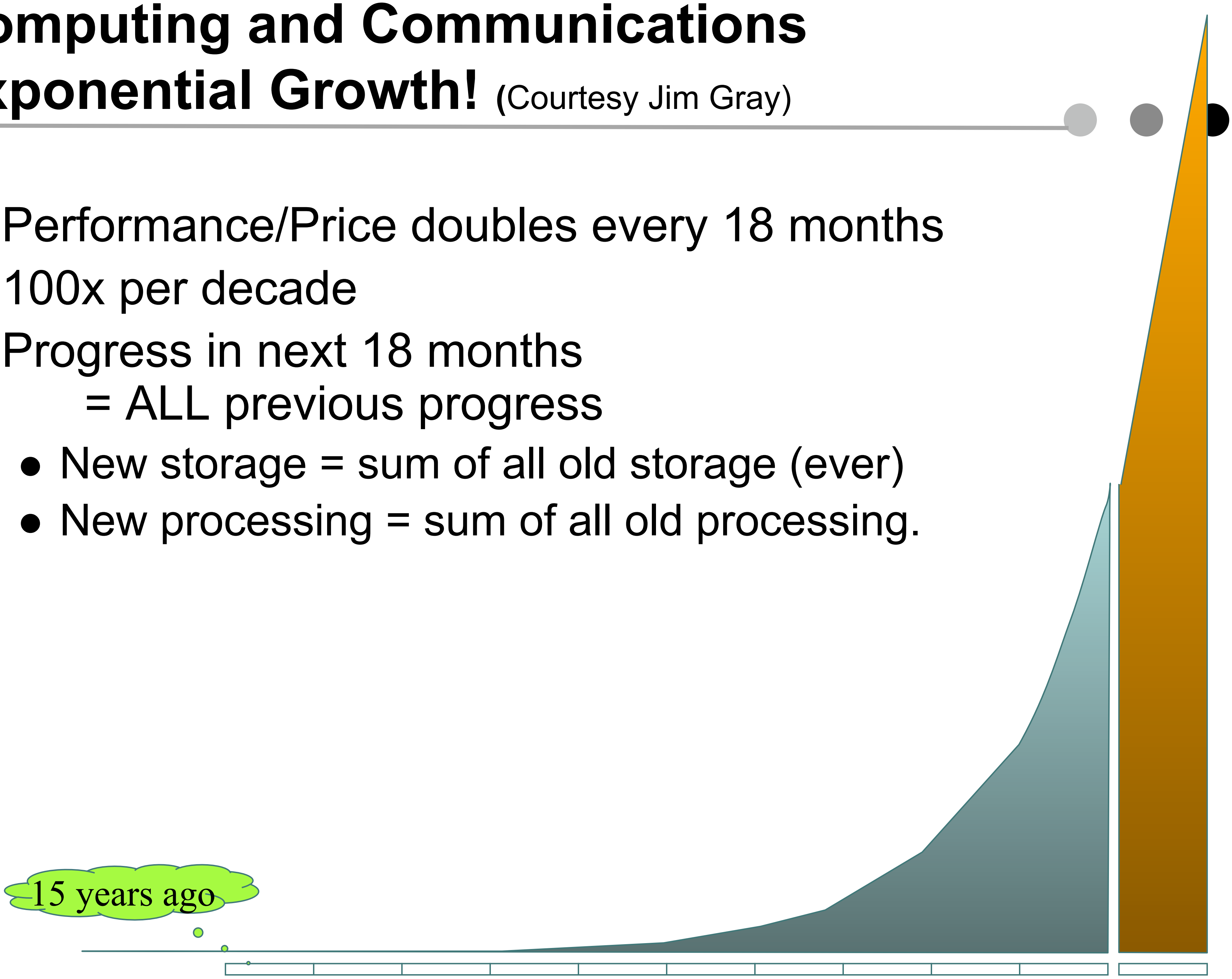
```
while (1);
```

A Typical Academic Computer (1988 vs. 2008)

	1988	2008	Ratio
Intel CPU transistors	0.5M	1.9B	~4000x
Intel CPU core x clock	10Mhz	4×2.66Ghz	~1000x
DRAM	2MB	16GB	8000x
Disk	40MB	1TB	25,000x
Network BW	10Mbits/sec	10GBits/sec	1000x
Address bits	32	64	2x
Users/machine	10s	< 1	>10x
$/machine	$30K	$3K	1/10x
$/Mhz	$30,000/10	$3,000/10,000	1/10,000x

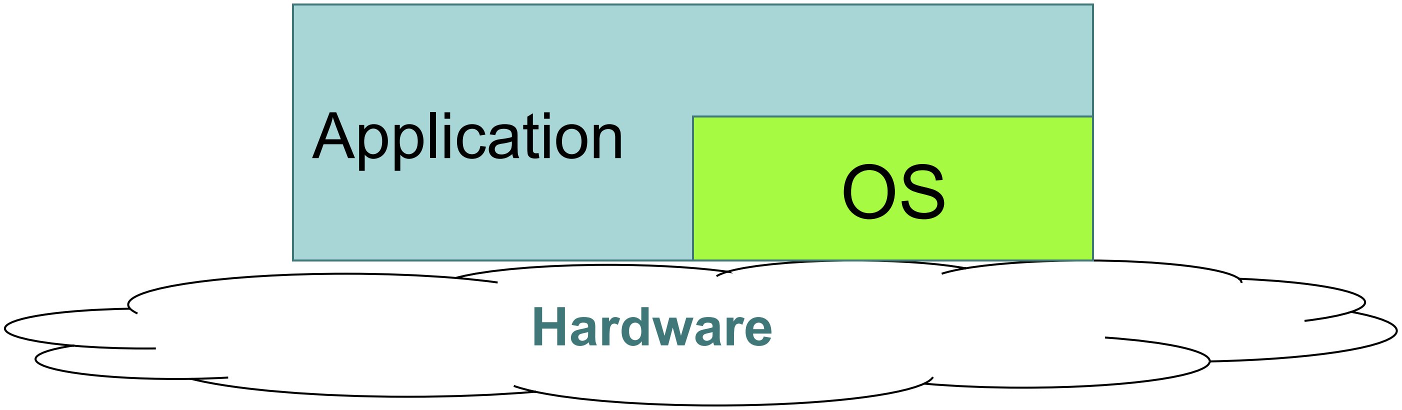
Computing and Communications Exponential Growth! (Courtesy Jim Gray)

- Performance/Price doubles every 18 months
- 100x per decade
- Progress in next 18 months
  = ALL previous progress
  - New storage = sum of all old storage (ever)
  - New processing = sum of all old processing.



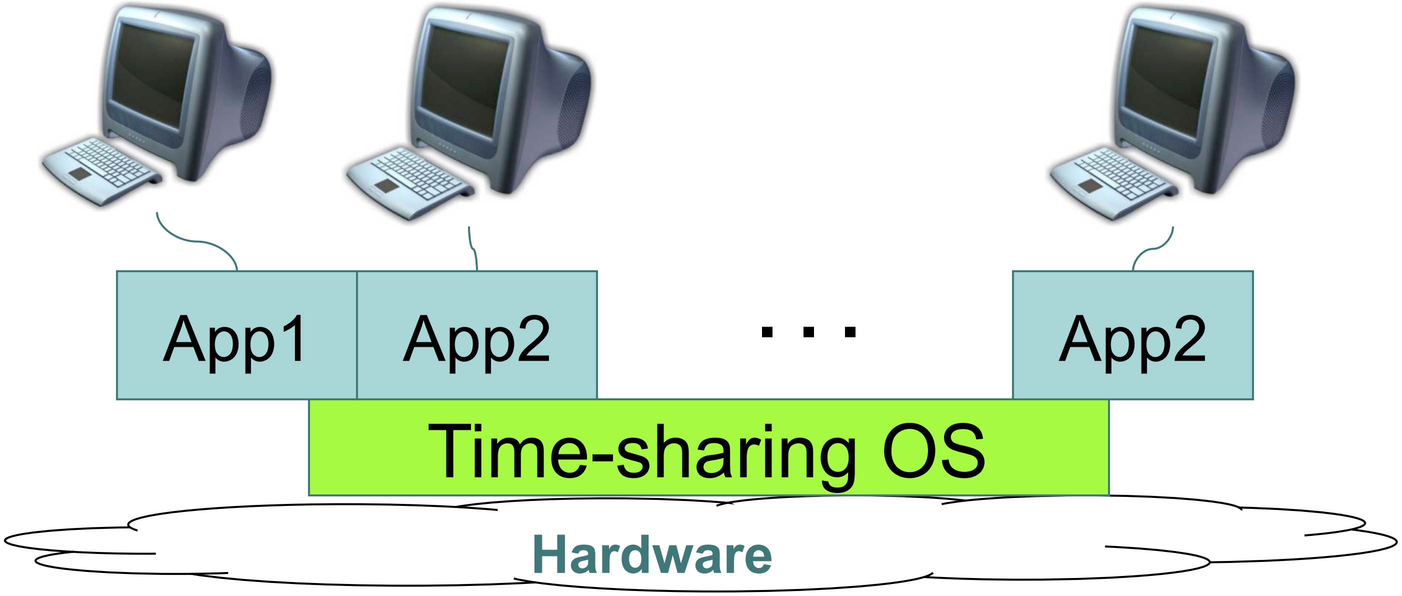
Phase 1: Hardware Expensive, Human Cheap

- User at console, OS as subroutine library
- Batch monitor (no protection): load, run, print
- Development
  - Data channels, interrupts; overlap I/O and CPU
  - Direct Memory Access (DMA)
  - Memory protection: keep bugs to individual programs
  - Multics: designed in 1963 and run in 1969
- Assumption: No bad people. No bad programs. Minimum interactions



Phase 2: Hardware Cheap, Human Expensive

- Use cheap terminals to share a computer
- Time-sharing OS
- Unix enters the mainstream
- Problems: thrashing as the number of users increases



Phase 3: HW Cheaper, Human More Expensive

- Personal computer
  - Altos OS, Ethernet, Bitmap display, laser printer
  - Pop-menu window interface, email, publishing SW, spreadsheet, FTP, Telnet
  - Eventually >100M unites per year
- PC operating system
  - Memory protection
  - Multiprogramming
  - Networking

Now: > 1 Machines per User

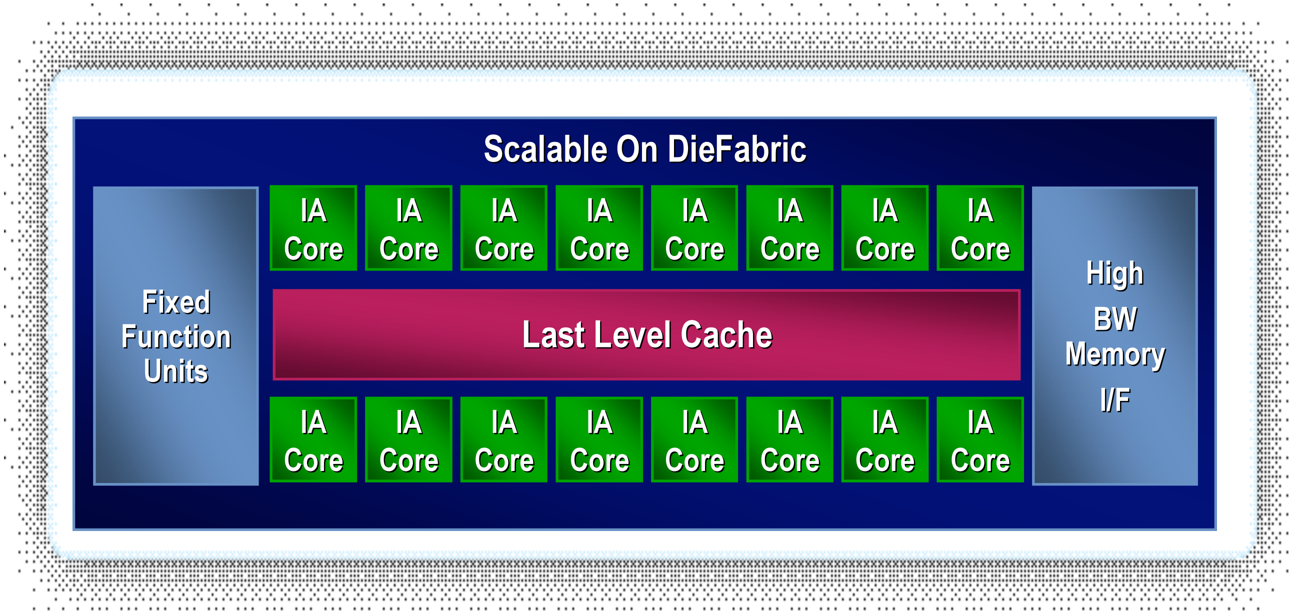
- Pervasive computers
  - Wearable computers
  - Communication devices
  - Entertainment equipment
  - Computerized vehicle
- OS are specialized
  - Embedded OS
  - Specially configured general-purpose OS

Now: Multiple Processors per Machine

- Multiprocessors
  - SMP: Symmetric MultiProcessor
  - ccNUMA: Cache-Coherent Non-Uniform Memory Access
  - General-purpose, single-image OS with multiproccesor support
- Multicomputers
  - Supercomputer with many CPUs and high-speed communication
  - Specialized OS with special message-passing support
- Clusters
  - A network of PCs
  - Commodity OS

Trend: Multiple "Cores" per Processor

- Multicore or Manycore transition
  - Intel and AMD have released 4-core CPUs
  - SUN's Niagara processor has 8-cores
  - Azul packed 24-cores onto the same chip
  - Intel has a TFlop-chip with 80 cores
- Accelerated need for software support
  - OS support for manycores
  - Parallel programming of applications



Trend: Datacenter as A Computer

- Cloud computing
  - Hosting data in the cloud
  - Software as services
  - Examples:
    - Google, Microsoft, Salesforce, Yahoo, …
- Utility computing
  - Pay as you go for computing resources
  - Outsourced warehouse-scale hardware and software
  - Examples:
    - Amazon, Nirvanix

Why Study OS?

- Learn about concurrency
  - Parallel programs run on OS
  - OS runs on parallel hardware
  - Best way to learn concurrent programming
- OS is a key part of a computer system
  - It makes our life better (or worse)
  - It is “magic” to realize what we want
  - It gives us “power”
- Understand how a system works
  - How many procedures does a key stroke invoke?
  - What happens when your application references 0 as a pointer?
  - Real OS is huge and impossible to read everything, but building a small OS will go a long way

What Is in COS 318?

- Methodology
  - Lectures with discussions
  - Readings with topics
  - Six projects to build a small and real OS
- Covered concepts
  - Operating system structure
    - Processes, threads, system calls and virtual machine monitor
  - Synchronization
    - Mutex, semaphores and monitors
  - I/O subsystems
    - Device drivers, IPC, and introduction to networking
  - Virtual memory
    - Address spaces and paging
  - Storage system
    - Disks and file system

Materials

- Textbook
  - *Modern Operating Systems*, 3rd Edition, Andrew S. Tanenbaum
- Lecture notes
  - Available on website
- Precept notes
  - Available on website
- Other resources – on website

Exam, Reading, Participation and Grading

- Grading (not curved)

First 5 projects:	50% with extra points
Midterm:	20%
Final project or final exam	20%
Reading & participation	10%

- Midterm Exam
  - Test lecture materials and projects
  - Tentatively scheduled on Thursday of the midterm week
- Reading assignments
  - Submit your reading notes BEFORE each lecture
  - Grading (3: excellent, 2: good, 1: poor, 0: none)
- Participation
  - Signup sheet at each lecture

The First 5 Projects

- Projects
  - Bootup (150-300 lines)
  - Non-preemptive kernel (200-250 lines)
  - Preemptive kernel (100-150 lines)
  - Interprocess communication and driver (300-350 lines)
  - Virtual memory (300-450 lines)
- How
  - Pair up with a partner, will change after 3 projects
  - Each project takes two weeks
  - Design review at the end of week one
  - All projects due Mondays 11:59pm
- The Lab
  - Linux cluster in 010 Friends Center, a good place to be
  - You can setup your own Linux PC to do projects

Project Grading

- Design Review
  - A signup sheet for making appointments
  - 10 minutes with the TA in charge
  - 0-5 points for each design review
  - 10% deduction if missing the appointment
- Project completion
  - 10 points for each project
  - Extra points available
- Late policy of grading projects
  - 1 hour: 98.6%, 6 hours: 92%, 1 day: 71.7%
  - 3 days: 36.8%, 7 days: 9.7%

Final Project

- A simple file system
- Grading (20 points)
- Do it alone
- Due on Dean’s date (~3 weeks)

Things To Do

- **Do not put your code on the web**
  - Other schools are using similar projects
- For today’s material:
  - Read MOS 1.1-1.3
- For next time
  - Read MOS 1.4-1.5

- Now: Pair up and fill out the form!