COS 318: Operating Systems

Introduction

Kai Li
Computer Science Department
Princeton University

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

Today

- Course staff and logistics
- What is operating system?
- Evolution of operating systems
- Why study operating systems?

Information and Staff

- Website
  - http://www.cs.princeton.edu/courses/cos318
  - Schedule, projects, lectures and precepts … (no paper)
- Textbooks
  - *Modern Operating Systems, 3rd Edition, A. S.*Tanenbaum
  - *Operating Systems: Principles and Practice, Beta Edition,* T. Anderson and M. Dahlin
- Instructor
  - Kai Li, 321 CS Building, li@cs.princeton.edu
    Office hours: Tue 3-5pm
- Teaching assistants
  - Aaron Blankstein (Project 4 and 5)
  - Scott Erickson (Project 2 and final project)
  - Yida Wang (Project 1 and 3)

Logistics

- Precepts
  - Time: Tue 7:30pm – 8:20pm in CS building 104
  - No second session
- Project 1
  - A tutorial on assembly programming and kernel debugging
    - ??? 9/20: 7:30-8:30pm in CS building 104
  - Design review
    - 9/24 (Monday) 3pm – 10pm (Friend 010)
    - Sign up online (1 slot per team)
  - Due: 9/30 (Sunday) 11:59pm

Grading, Exams, and Reading

- Grading (not curved)
  - First 5 projects: 45% with extra points
  - Final project 15%
  - Midterm: 15%
  - Final exam 15%
  - Reading & participation 10%
- Midterm and Final Exam
  - Test lecture materials and projects
  - Midterm: Tuesday of the midterm week, 10/23
- Reading and participating
  - Submit your reading notes in ASCII format BEFORE lecture
  - Grading (3: excellent, 2: good, 1: poor, 0: none)
  - Write your name and concise notes (one small paragraph for each question)

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)
  - File system
- How
  - Pair up with a partner for project 1, 2, 3
  - Different partner for 4, 5
  - Do yourself for 6
  - Each project takes 2-3 weeks
  - Design review at the end of week one
  - All projects due Sundays 11:59pm
- The Lab aka “The Fishbowl”
  - Linux cluster in 010 Friends Center, a good place to be
  - You can setup your own environment to do projects

Project Grading

- Design Review
  - Signup online for making appointments
  - 10 minutes with the TA in charge
  - 0-5 points for each design review
  - 10% deduction for missing an appointment
- Project completion
  - 10 points for each project plus possible extra points
- Late policy of grading projects
  - 1 hour: 98.6%, 6 hours: 92%, 1 day: 71.7%
  - 3 days: 36.8%, 7 days: 9.7%

Piazza for Discussions

- Piazza is a convenient forum
  - Based on last year's experience
- Easy ask and answer questions
  - Students are encouraged to answer questions
  - Staff will try to answer in timely manner
- Only use email if the question is personal/private

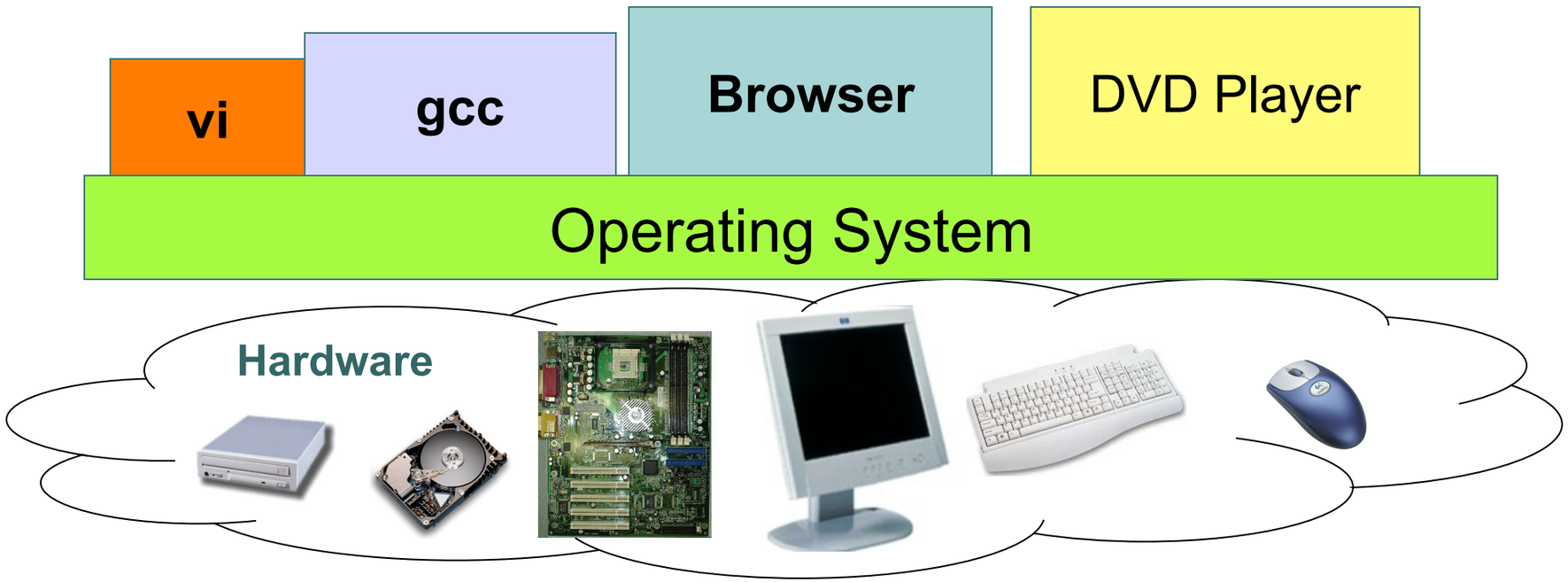
Ethics and Other Issues

- **Do not put your code or design on the web**
  - Other schools are using similar projects
- **Follow Honor System**
  - Ask me if you are not sure
  - Ask each other questions is okay
  - Work must be your own (or your team)

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
  - COS475: Computer Architecture
- Courses needing COS318
  - COS 461: Computer Networks
  - COS 518: Advanced Operating Systems
  - COS 561: Advanced Computer Networks

What Is Operating System?



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

What Do Operating Systems Do?

- Provide a layer of abstraction
  - User programs can deal with simpler, high-level concepts
  - Hide complex and unreliable hardware
  - Protect application software from crashing a system
- Implement the OS abstraction: manage resources
  - Manage application interaction with hardware resources
  - Make finite CPU, memory and I/O “infinite”
  - Allow multiple users to share resources without hurting each other

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?

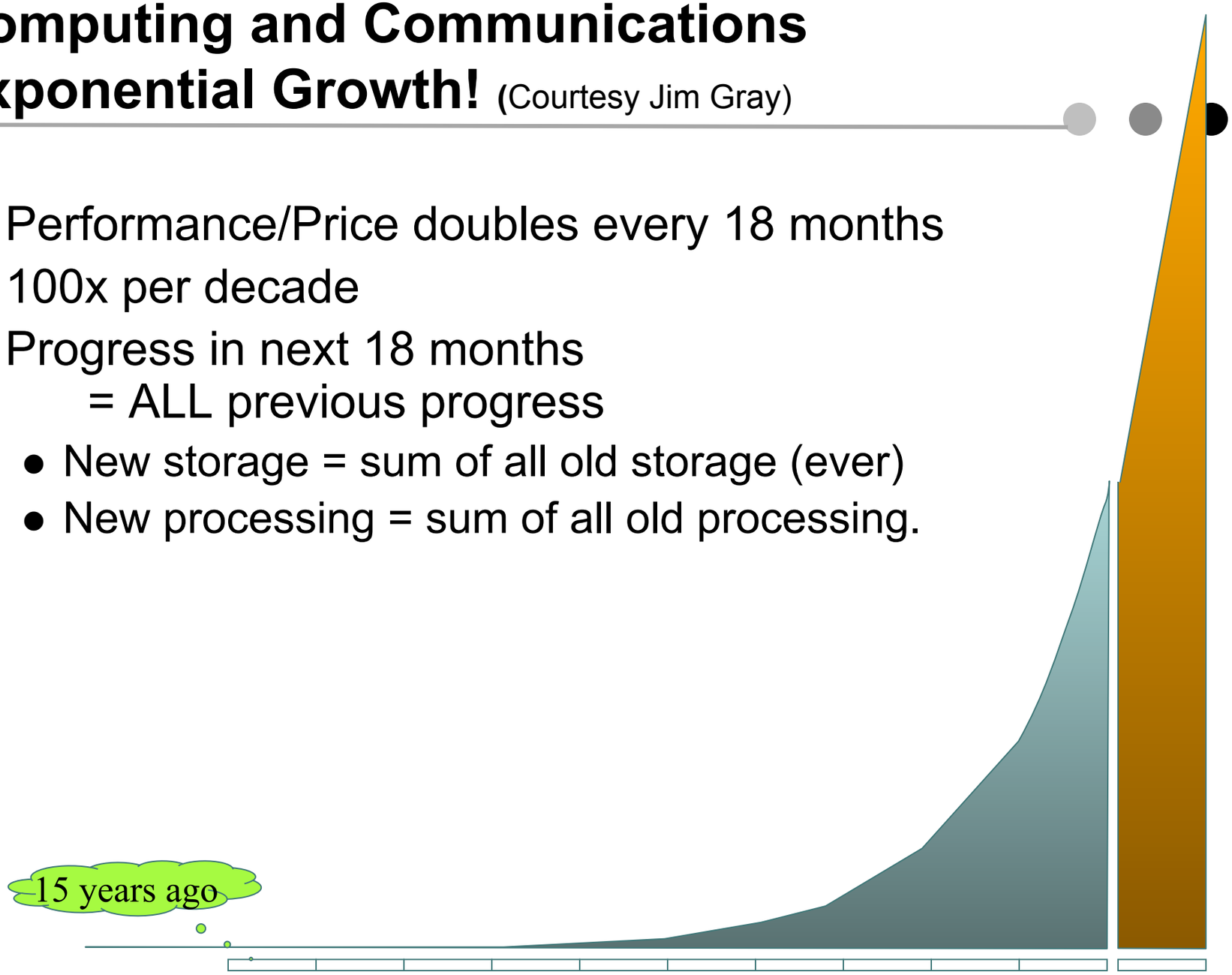
```
while (1);
```

A Typical Academic Computer (1981 vs. 2011)

	1981	2011	Ratio
Intel CPU transistors	0.1M	1.9B	~20000x
Intel CPU core x clock	10Mhz	10×2.4Ghz	~2,400x
DRAM	1MB	64GB	64,000x
Disk	5MB	1TB	200,000x
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	$3,000/24,000	1/2,400x

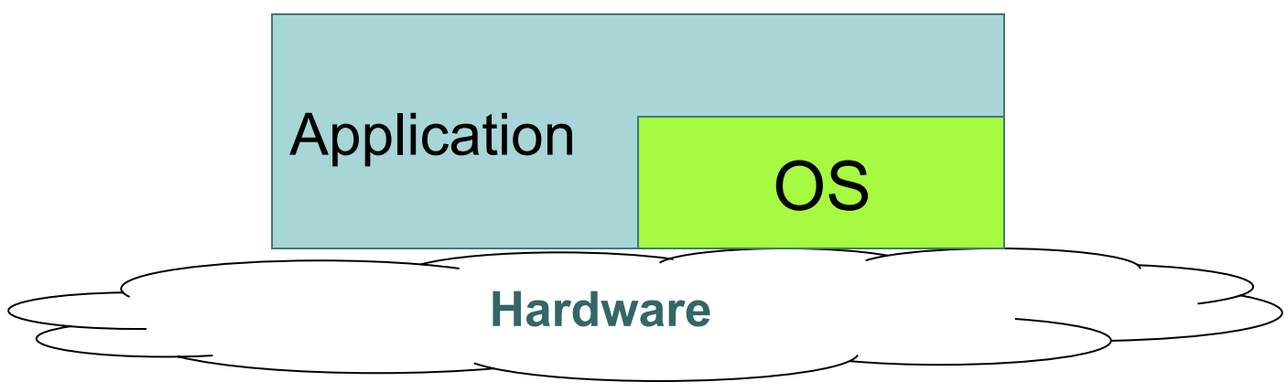
Computing and Communications Exponential Growth! (Courtesy Jim Gray)

- 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.



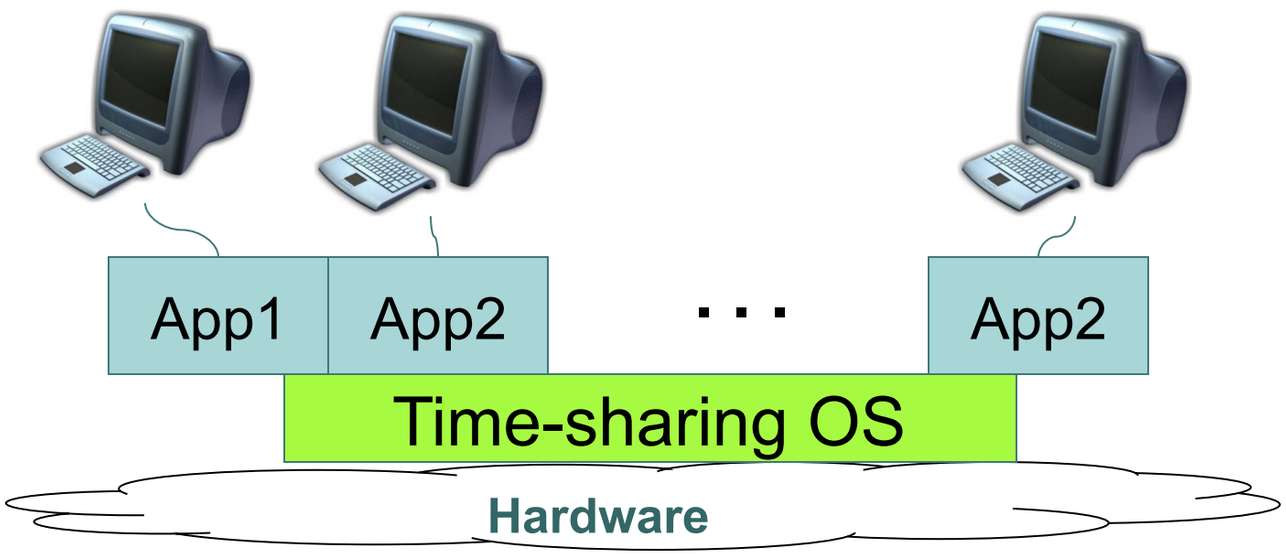
Phase 1: Hardware Expensive, Human Cheap

- 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

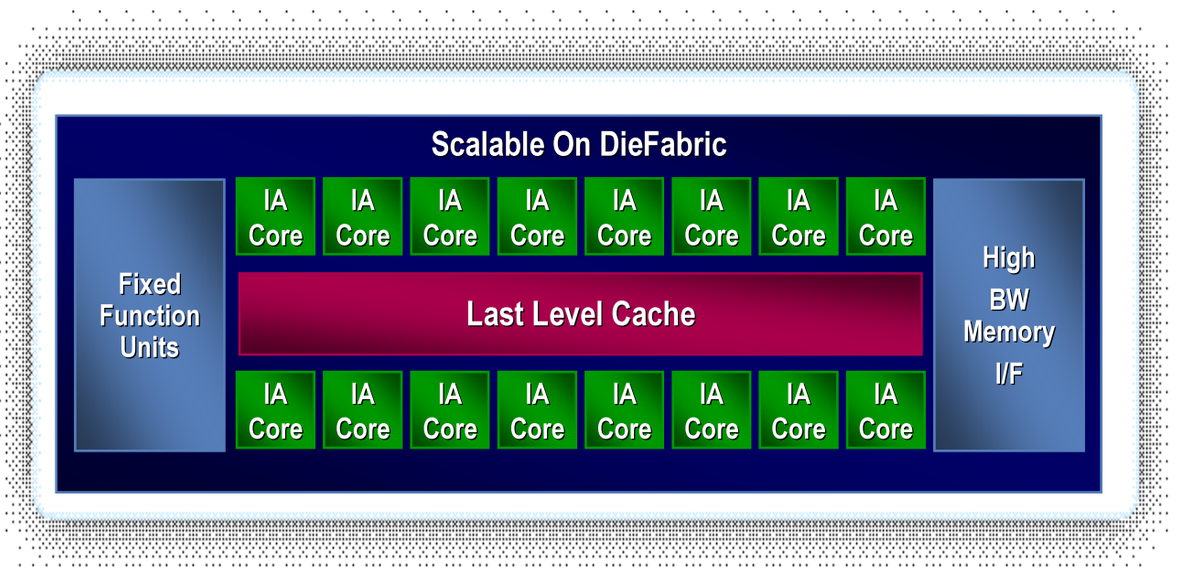
- Pervasive computers
  - Wearable computers
  - Communication devices
  - Entertainment equipment
  - Computerized vehicle
- OS are specialized
  - Embedded OS
  - Specially general-purpose OS (e.g. iOS, Android)

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
  - Server OS w/ cluster abstraction (e.g. MapReduce)

Trend: Multiple “Cores” per Processor

- Multicore or Manycore transition
  - Intel xeon processor has 10 cores / 20 threads
  - New Intel xeon phi has 50 cores
  - nVidia GPUs has 3000 FPUs
- 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?

- 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” (reduce fear factor)
- Learn about concurrency
  - Parallel programs run on OS
  - OS runs on parallel hardware
  - Best way to learn concurrent programming
- 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

Why Study OS?

- Important for studying other areas
  - Networking, distributed systems, security, …
- More employable
  - Become someone who understand “systems”
  - Become the top group of “athletes”
  - Ability to build things from ground up

- Question:
  - Why shouldn’t you study OS?

Things to Do

- Today's material
  - Read *MOS 1.1-1.3*
  - Lecture available online
- Next lecture
  - Read MOS 1.4-1.5
  - Summit notes
- Make "tent" and leave with me
  - Use next time
- Use piazza to find a partner
  - Work together on project 1, 2, 3