* Introduction
  + Give information
  + Office hours TBD
  + At this point, looks like I will be running section
  + Section will could be structured in a few different ways. Primarily, it will be a recitation section where we highlight / cover materiel in a similar vein as presented in lecture. Will also be used as time to overview topics for assignments, use extra time as free office hours.
  + I am your friend in this class!!! This class isn't that easy, come to me (and other TAs) for help
  + Questions about logistics?
* Today's (short) topic: Operating systems and how they handle the hard stuff
  + Yesterday's lecture was an introduction to operating systems and the abstractions they provide us. As modern day application developers, we don't have to worry about managing the fine grained details of our system we are working In. You just have to think about your program running and that's it. However, what's actually going on behind the scenes is much more complicated.
  + There are two main jobs of the OS: Abstraction of difficult tasks and management of resources (which is a really just a big difficult task).
  + What aspects of the system needed to be managed by the user in the past are now handled seamlessly in modern OS?
    - Processing time (CPU, GPU)
    - Physical Memory (RAM)
    - Persistent Memory (HDD, SSD)
    - Network Connection (Wifi, ethernet, bluetooth)
  + What aspects of the system are abstracted into the OS?
    - A big abstraction is how the OS gives the illusion that the user is working exclusively with resources (i.e. their own memory, own processing time, their own I/O devices).
    - The OS also abstracts the job of handling devices.
      * Audio - 3.5mm jack, USB speakers, bluetooth headset
      * Network connection - Could be over wifi, ethernet, bluetooth
      * Printer - Could be connected via usb or over the network, is it ink or laser printer?
    - Another big thing the OS also abstracts is the concept of security.
      * You will see how this is done in the future, but it helps protect malicious users from doing bad things.
  + Users get access to the resources managed by the OS via system calls. How many people are familiar with the concept of a system call?
    - a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating **system** it is executed on. A **system call** is a way for programs to interact with the operating **system**. There is a lot more to say about system calls, but we won't get into it today.
    - You probably use system calls all the time in your code without knowing it!
    - Can you guys think of any system calls?
      * <http://man7.org/linux/man-pages/man2/syscalls.2.html>
    - CPU/GPU Resource Access
      * In a linux environment, if you have ever started an application, that creates a new application. This uses the **fork** system call.
    - RAM
      * Did you ever write a program in 352 that used malloc? Malloc uses a system call named **mmap** to reserve physical memory to allocate to the user!
    - Persistent memory
      * Opening a file to read directly correlates to the system call **open**. The file will probably remain in RAM until you use **flush** to force the data onto the disk.
    - Network Connection
      * Those of you who have taken networking know there is many layers of complexity in network communications. You have to think about the different protocols you must use, managing your port, interface with a device to access the network. System calls help abstract that idea through the use of sockets.
  + So we see system calls are used to get resources that are being managed by the OS as well as us the abstractions provided to us! Hopefully this helps solidify what we mean when we say the OS manages our resources and provides us abstractions. Any questions?