6.033 - Intro to Computer Systems
Lecture 1
Katrina LaCurts, lacurts@mit.edu

1. Introduction to Systems
   - What is a system?
   - Complexity makes building systems difficult

2. Why is Complexity Bad?
   - Limits what we can build
   - Causes lots of other problems

3. Mitigating Complexity
   - We mitigate complexity with modularity and abstraction
     - Modular systems are easier to reason about, manage, change, improve
     - Modularity reduces fate-sharing.
     - Abstraction lets us specify interfaces without specifying implementation
     - Good abstraction decreases the number of connections between modules

4. Enforced Modularity
   - Soft modularity isn't enough
   - One way to enforce is with a client/server model
     - Reduces fate-sharing
     - Important: remote procedure calls (RPCs) != procedure calls (PCs)
       - Have to deal with different types of failure (network, server,..)
         - These failures are tricky, but starting with a modular design will let us reason about them and deal with them

5. Other Goals
   - Beyond complexity, we might also want: scalability, fault-tolerance, security, performance, etc.
   - Starting with a good, modular design helps achieve these properties
   - Difficult to get all at once; there are trade-offs