6.033 Spring 2015
Lecture #1

- Complexity
- Modularity and abstraction
- Enforced modularity via client/server models
http://mit.edu/6.033

Schedule

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 2</td>
<td>REC 1: Worse is Better</td>
<td>LEC 1: Enforced Modularity and Client/server Organization</td>
<td>REC 2: Therac-25</td>
<td>TUT 1: Introduction to system critiques (run by TAs)</td>
</tr>
<tr>
<td>Reg day</td>
<td>Preparation: Read Worse is Better</td>
<td>Supplemental Reading: Book sections 1.1-1.5, and 4.1-4.3</td>
<td>Preparation: Therac-25 paper</td>
<td>Assigned: Paper critique #1</td>
</tr>
<tr>
<td></td>
<td>Assigned: Hands-on DNS</td>
<td>First day of classes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 9</td>
<td>LEC 2: Naming</td>
<td>LEC 3: Operating systems</td>
<td>REC 4: UNIX</td>
<td>TUT 2: How to read a paper (run by communication instructors)</td>
</tr>
<tr>
<td></td>
<td>Supplemental Reading: Book sections 2.2, and 3.1</td>
<td></td>
<td></td>
<td>Due: Paper critique #1</td>
</tr>
<tr>
<td></td>
<td>Preparation: Book section 4.4: &quot;Case study: The Internet Domain Name System (DNS)</td>
<td>Supplemental Reading: Book sections 5.1, 5.3, and 5.4</td>
<td>Preparation: Unix paper</td>
<td>Assigned: Paper critique #2</td>
</tr>
<tr>
<td></td>
<td>DUE: Hands-on DNS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assigned: Hands-on UNIX</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fill out form for recitation assignments
link on home page
what is a system?

a set of interconnected components that has an expected behavior observed at the interface with its environment
6.033 Approach to Systems

**lectures:** big ideas + examples  
Katrina LaCurts, Hari Balakrishnan

**recitations:** read papers describing successful systems  
Arvind, Mark Day, Dina Katabi, Sam Madden, Martin Rinard, Karen Sollins, Peter Szolovits

**hands-ons:** play with successful systems

**design project:** practice designing and writing  
*TA*s: Ellen Finch, David Goehring, Ameesh Goyal, Webb Horn, Qian Long, Manali Naik, Andrew Nguyen, Amy Ousterhout, Cong Yan  
*Writing staff:* Jared Berezin, Amy Carleton, Amelia Herb, Nora Jackson, Janis Melvold, Juergen Schoenstein, Jessie Stickgold-Sarah, Linda Sutliff, Michael Trice

**exams:** reasoning about system design
what is a system?
a set of interconnected components that has an expected behavior observed at the interface with its environment

what makes building systems difficult?
complexity
Today’s Systems are Incredibly Complex

source: http://www.informationisbeautiful.net/visualizations/million-lines-of-code/
Emergent Properties
(ethernet example)

A

3Mbps link, 5μsec latency

=> minimum-packet size of 30 bits
for collision detection to work

B

experimental ethernet: 3Mbps link, 5μsec latency, 40-bit packet headers
Emergent Properties
(ethernet example)

10Mbps link, 12.5μsec latency

=> minimum-packet size of 250 bits for collision detection to work

*first ethernet standard*: 10Mbps link, 12.5μsec latency, **112-bit** packet headers

minimum packet size was an **emergent property** of ethernet
how can we mitigate complexity?
how do we enforce modularity?
Stub Clients and RPCs

Class `webBrowser`
(on machine 1)

```python
def main():
    html = browser_load_url(URL)
    ...

def browser_load_url(url):
    msg = url # could reformat
    send request
    wait for reply
    html = reply # could reformat
    return html
```

Class `webServer`
(on machine 2)

```python
def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```
Challenges with RPCs

Client  ←  Server
Challenges with RPCs

Client \hspace{1cm} internet \hspace{1cm} Server
Challenges with RPCs

Client \rightarrow internet \rightarrow Server

load("buy.html?item&ccNo=xxx")

load("buy.html?item&ccNo=xxx")

load("buy.html?item&ccNo=xxx")
Challenges with RPCs

problem: server can still fail
• **Complexity**
  Comes from many sources, limits what we can build, causes unforeseen issues; can be mitigated with **modularity** and **abstraction**

• **Enforced modularity**
  One way to enforce modularity is with a **client/server model**, where the two modules reside on different machines and communicate with RPCs; network/server failures are still an issue

**next lecture:** naming, which allows modules to communicate

**subsequent lectures:** operating systems, which provide modularity on a single machine