Lecture 12
Some project stuff
2018 Project Schedule

<table>
<thead>
<tr>
<th>Su</th>
<th>Mo</th>
<th>Tu</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **You are here**: March 8
- **You are now here!**: March 29

- First class
- Assignment 1 due
- Assignment 2 due
- Assignment 3 due
- Assignment 4 due; team meetings with bwk th
design document due by midnight Saturday
spring break
- Assignment 5 due; weekly TA meetings start
- Project prototype
- Alpha test
- Last class; beta test
demo days (exact days TBD)
projects due by midnight Sunday
Weekly TA meetings start March 26

• **one grad TA will be your advisor / mentor**
  – advise and monitor so you don’t get stuck
  – help find other groups that have already overcome your current hurdles if possible

• **everyone comes to each meeting prepared**
  – what progress have you made, what next steps are planned,
    what problems do you need help with

• **it's your project**
  – It’s your project to design and scope
  – It’s your code to write and debug
  – It’s your application to put into the hands of users to test
Elevator pitches start after break

• a 60-second summary of what your project is and why it’s going to be valuable.

• “Just make an affirmative statement about what you do and why it’s important. SpaceX has a great elevator pitch: ‘Launch costs haven’t come down in decades. We slash them by 90%. The market is $XXbn.’ ”

• “We cured pancreatic cancer in monkeys. We need cash for Phase II trials; if this works, it’s a $10 billion market annually.”

• (from Peter Thiel’s Stanford course, lecture 8)
CAS: Central Authentication Service

- If your project requires users to log in with a Princeton netid, don't ask users to send you their passwords at all, and especially not in the clear.

- OIT provides a central authentication service
  - The user visits your startup page
  - The user is asked to authenticate via OIT's service
  - The name and password are sent to an OIT site for validation (without passing through your code at all)
  - If OIT authenticates the user, your code is called.

- OIT web page about CAS:
  https://sp.princeton.edu/oit/eis/iam/authentication/CAS/CAS%20Developer%20KB.aspx

- Sample code:
  www.cs.princeton.edu/~bwk/public_html/CAS
Authentication for projects (etc.)

- **PHP version**
  ```php
  <?php
  require 'CASClient.php';
  $C = new CASClient();
  $netid = $C->Authenticate();
  echo "Hello $netid"; // or other code
  ?>
  ```

- **Python version**
  ```python
  import CASClient, os
  C = CASClient.CASClient()
  netid = C.Authenticate()
  print "Content-Type: text/html\n"
  print "Hello %s" % netid # or other code
  ```

- **Java version**
  ```java
  CASClient casClient = new CASClient();
  String netid = casClient.authenticate();
  System.out.println("Content-type: Text/html\n");
  System.out.println("Hello " + netid);
  ```
Behind the scenes in the client libraries

- your web page sends user to
  https://fed.princeton.edu/cas/login?
  service=url-where-user-will-log-in

- CAS sends user back to the service url to log in
  with a parameter ticket=hash-of-something

- your login code sends this back to
  https://fed.princeton.edu/cas/validate?
  ticket=hash&service=url...log-in

- result from this is either 1 line with "no"
  or two lines with "yes" and netid
Source code management systems

- Git, SVN, Mercurial, Bazaar, Perforce, ...
- for managing large projects with multiple people
  - work locally or across a network
- store and retrieve all versions of all directories and files in a project
  - source code, documentation, tests, binaries, ...
- support multiple concurrent users
  - independent editing of files
  - merged into single version
- required for COS 333 projects!
  - save all previous versions of all files so you can back out of a bad change
  - log changes to files so you can see who changed what and why
  - maintain consistency by resolving conflicting changes made by different users
  - protects against disasters
Basic sequence for all systems

- create a repository that holds copies of your files
  - including all changes and bookkeeping info
- each person checks out a copy of the files
  - "copy - modify - merge" model
  - get files from repository to work on
    - does not lock the repository
  - make changes in a local copy
  - when satisfied, check in (== commit) changes
- if my changes don't conflict with your changes
  - system updates its copies with the revised versions
  - automatically merges edits on different lines
  - keeps previous copies
- if my changes conflict with your changes
  - e.g., we both changed lines in the same part of a file,
    checkin is not permitted
  - we have to resolve the conflict manually
Git

- originally written by Linus Torvalds, 2005
- distributed
  - no central server: every working directory is a complete repository
  - has complete history and revision tracking capabilities
- originally for maintaining Linux kernel
  - lots of patches
  - many contributors
  - very distributed
- originally created because of
  - dispute with BitKeeper (a commercial system)
  - dissatisfaction with CVS / SVN
Basic Git sequences (git-scm.com/documentation, gitref.org)

```bash
cd project
git init
    makes .git repository
git add .
git commit
    makes a snapshot of current state
[modify files]
git add ... [for new ones]
git rm ... [for dead ones]
git commit
git log --stat
git clone [url]
    makes a copy of a repository
```
THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.
<table>
<thead>
<tr>
<th>Comment</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATED MAIN LOOP &amp; TIMING CONTROL</td>
<td>14 HOURS AGO</td>
</tr>
<tr>
<td>ENABLED CONFIG FILE PARSING</td>
<td>9 HOURS AGO</td>
</tr>
<tr>
<td>MISC BUGFIXES</td>
<td>5 HOURS AGO</td>
</tr>
<tr>
<td>CODE ADDITIONS/EDITS</td>
<td>4 HOURS AGO</td>
</tr>
<tr>
<td>MORE CODE</td>
<td>4 HOURS AGO</td>
</tr>
<tr>
<td>HERE HAVE CODE</td>
<td>4 HOURS AGO</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAAAAAAAAAAAAAA</td>
<td>3 HOURS AGO</td>
</tr>
<tr>
<td>ADKFJSLKDFJSKLDLFFFKJK</td>
<td>3 HOURS AGO</td>
</tr>
<tr>
<td>MY HANDS ARE TYPEING WORDS</td>
<td>2 HOURS AGO</td>
</tr>
<tr>
<td>HAAAAAAAAANDS</td>
<td>2 HOURS AGO</td>
</tr>
</tbody>
</table>

As a project drags on, my Git commit messages get less and less informative.
Private Github repositories (free)

- https://www.princeton.edu/researchcomputing/services/github-form-new/

GitHub Service Request

Please register here to request or configure GitHub repositories

Princeton University has contracted with GitHub to sponsor an Organization that supports private or public repositories for the use of the University community.

***UPDATE 12-14-2015***

GitHub has recently made changes to the structure and permissions of the Princeton University GitHub Organization. Please read the changes below carefully. If you have any questions please email cses@princeton.edu.

All existing repositories are unaffected and unchanged. You and your collaborators’ access privileges to existing repositories completely unaffected.

You can now create teams and add other Princeton members to the teams yourself. You can also control individual access to private repositories yourself, through the “collaborator” option in the repositories setting. Please see the GitHub FAQ for more detail.