Some project stuff
2017 Project Schedule

You are here

You are now here!

Mar

1 2 3 4

start thinking about possible projects & people

5 6 7 8 9 10 11

your team should be formed; schedule meeting with bwk

12 13 14 15 16 17 18

initial project discussions with bwk by 3/14

design document due by 3/19

19 20 21 22 23 24 25

spring break -- don't waste it

weekly TA meetings start this week

Apr

1

2 3 4 5 6 7 8

project prototype

9 10 11 12 13 14 15

alpha test

16 17 18 19 20 21 22

23 24 25 26 27 28 29

last class; beta test

30

demo days this week (probably M-W)

May

1 2 3 4 5 6

Project due by midnight Sunday

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27
Weekly TA meetings start March 27

• 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)
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
- highly recommended 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
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 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)

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 --summary


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>AAAAAAAAAAAAAAA</td>
<td>4 hours ago</td>
</tr>
<tr>
<td>ADKFJSLKDFJSKDLFJ</td>
<td>3 hours ago</td>
</tr>
<tr>
<td>My hands are typing 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-

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.