CSE306 Software Quality in Practice

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
"20% POLL" feedback

1. More time for the labs if possible, maybe spending a week on one lab to make sure the concepts are understood such as playing around with git and its functions.

2. As of now nothing, I'm personally gaining a lot from having 4 hours labs every week.

3. I think the file transferring between local machines and timberlake is annoying.

4. A little more guidance on the longer projects.

1. We are interleaving exercises with different tools so you can better see how they support each other. This week we're doing more with git and gdb.

2. OK - good to know!

3. Fair enough. Because we are working in a distributed and differently managed environment we're having to do some shuffling of files. This is not necessarily a bad skill to have :-)

4. Remember: the goal of the first long project, PRE, was document your current process. Building the code was secondary. The next one, EXP1, will give you a chance to practice what you've learned in lab, and demonstrate the progress you've made.
Another git exercise (more in-depth)

Another bug hunt

No team project this week. Start EXP 1 next week. Use that to demonstrate the skills you’ve learned from LEX01 - LEX06.
Intermediate git

- Concepts
  - Branch
  - Fetch
  - Merge
  - Reset
Local Machine
(e.g. your laptop, or timberlake if you've ssh'ed in)

Remote
(e.g. bitbucket, github, CSE servers)

stash
workspace
index staging
local repository
remote repository
What you see when working

stash
workspace
index staging
local repository
remote repository
Cloning a remote
Makes a copy of remote repo in local repo and checks out branch into workspace

stash
workspace
index staging
local repository
remote repository

`git clone`
Add a file to the staging area (add it to the index)

- stash
- workspace
- index staging
- local repository
- remote repository

`git add`
Create a new commit object with the staged items from the index

- stash
- workspace
- index staging
- local repository
- remote repository

```
git commit
```
Pushing files from local repo to remote repo

- stash
- workspace
- index staging
- local repository
- remote repository

`git push`
Pulling files

"git pull is shorthand for git fetch followed by git merge FETCH_HEAD"

[https://git-scm.com/docs/git-pull]
Grab files from remote

- stash
- workspace
- index staging
- local repository
- remote repository

`git fetch`
Create a commit combining the contents of two branches

- stash
- workspace
- index staging
- local repository
- remote repository

`git merge`
Recall the files in .git

-rw-r--r-- 1 alphonce staff  23 Apr 17 13:26 HEAD
drwxr-xr-x 2 alphonce staff  68 Apr 17 13:26 branches
-rw-r--r-- 1 alphonce staff  328 Apr 17 13:26 config
-rw-r--r-- 1 alphonce staff  73 Apr 17 13:26 description
drwxr-xr-x 12 alphonce staff 408 Apr 17 13:26 hooks
-rw-r--r-- 1 alphonce staff 137 Apr 17 13:26 index
drwxr-xr-x 3 alphonce staff 102 Apr 17 13:26 info
drwxr-xr-x 4 alphonce staff 136 Apr 17 13:26 logs
drwxr-xr-x  7 alphonce staff 238 Apr 17 13:26 objects
-rw-r--r-- 1 alphonce staff 107 Apr 17 13:26 packed-refs
drwxr-xr-x 5 alphonce staff 170 Apr 17 13:26 refs
pointer to the current branch
$ more head
ref: refs/heads/master
$ git ls-files
README.md
$ ls -l objects
total 0
drwxr-xr-x  3 alphonce  staff  102 Apr 17 13:26 25
drwxr-xr-x  3 alphonce  staff  102 Apr 17 13:26 39
drwxr-xr-x  3 alphonce  staff  102 Apr 17 13:26 9c
drwxr-xr-x  2 alphonce  staff   68 Apr 17 13:26 info
drwxr-xr-x  2 alphonce  staff   68 Apr 17 13:26 pack

$ git cat-file -t 25b4
commit
$ git cat-file -t 9ce9
tree
$ git cat-file -t 39af
blob

drwxr-xr-x  4 alphonce  staff  136 Apr 17 13:26 log
drwxr-xr-x  7 alphonce  staff  238 Apr 17 13:26 objects
-rw-r--r--  1 alphonce  staff  107 Apr 17 13:26 packed-refs
drwxr-xr-x  5 alphonce  staff  170 Apr 17 13:26 refs
pointers to commits

-rw-r--r-- 1 alphonce staff 23 Apr 17 13:26 HEAD
drwxr-xr-x 2 alphonce staff 68 Apr 17 13:26 branches
-rw-r--r-- 1 alphonce staff 328 Apr 17 13:26 config
-rw-r--r-- 1 alphonce staff 73 Apr 17 13:26 description
drwxr-xr-x 12 alphonce staff 408 Apr 17 13:26 hooks
-rw-r--r-- 1 alphonce staff 137 Apr 17 13:26 index
drwxr-xr-x 3 alphonce staff 102 Apr 17 13:26 info
drwxr-xr-x 4 alphonce staff 136 Apr 17 13:26 logs
drwxr-xr-x 7 alphonce staff 238 Apr 17 13:26 objects
-rw-r--r-- 1 alphonce staff 107 Apr 17 13:26 packed-refs
drwxr-xr-x 5 alphonce staff 170 Apr 17 13:26 refs
$ ls -l refs
 total 0
  drwxr-xr-x  3 alphonce staff  102 Apr 17 13:26 heads
  drwxr-xr-x  3 alphonce staff  102 Apr 17 13:26 remotes
  drwxr-xr-x  2 alphonce staff   68 Apr 17 13:26 tags
$ ls refs/heads/
  master
$ ls refs/remotes/
  origin
$ lsrefs/remotes/origin/
  HEAD master

Local branches

Remote HEAD and branches
Possible states of a file

- Modified
- Unmodified
- Untracked
- Staged

States transitions:
- Edit → Unmodified
- Commit → Staged
- Add → Modified
- Add → Untracked
- Add → Staged
commit preserves contents
(accidental removals can be recovered from)
understanding 'reset'

- workspace
- index staging
- local repository
  - HEAD
  - master
Create file
git add file


```
$ git commit -m "..."
```
Edit file
git add file
git commit -m "..."
edit file

`git add file`

`git commit -m "..."`
Let's take this as our starting point
**git reset --soft HEAD~**

**Moves the branch that HEAD refers to.**
RESULT: master is moved; HEAD still refers to master.
STARTING POINT
Moves the branch that HEAD refers to, and unstages changes to the index.
RESULT: master is moved; HEAD still refers to master.
STARTING POINT
Moves the branch that HEAD refers to, unstages changes, and makes workspace reflect the staging area.
RESULT: master is moved; HEAD still refers to master.

`git reset --hard HEAD~`

-- hard can result in data loss if workspace had uncommitted files.
understanding branching


git branch <name> creates a new pointer
Let's take this as our starting point.
git branch B
git checkout B
create fileB

git add fileB
git commit -m "..."
git checkout master
EXERCISE
Let us first switch to branch ‘a’:

`git checkout a`
Now edit and commit F2.

`emacs F2 (call edited version F2')`
`git add F2'`
`git commit -m "..."`
Create FS, add, and commit.

```bash
emacs FS
git add FS
git commit -m "..."
```