Why so software engineeringy?

Athula Balachandran
Wolf Richter

Are we really that serious?

I'm now TAing an undergrad course. What I'm most surprised is that some undergrads are REALLY SMART!! OTOH, grad TA, I, am so dull and stupid. :) It's because we have little motivation. We do not get paid for TAing. My profs may not be happy if I spend too much time on it (eg 20hr/week).

Like · Comment · 11 hours ago near Pittsburgh ·

[redacted] I heard that officially a TA job==20 hrs/week.
Like

[redacted] I heard [redacted] and Wolfgang Richter had a debate with this issue few days ago, I mean, how smart CMU undergrads are in general :-) I'm also TAing an undergrad course and looking forward to seeing bright students!
2 hours ago · Like

Wolfgang Richter My vote of confidence is in the students. I set the bar as high as possible. No coddling. We're going deep in hard and fast systems design and development. I give them source documents and source materials ;)
2 hours ago · Like

Wolfgang Richter How else are you supposed to produce the best grads in the world? :p
2 hours ago · Like

Wolfgang Richter see what I've been up to here:
https://github.com/theonewolf/15-441-Recitation-Sessions

theonewolf/15-441-Recitation-Sessions - GitHub
github.com
15-441-Recitation-Sessions - 15-441 Computer Networks Recitation Sessions from Carnegie Mellon University

2 hours ago · Like · Remove Preview

[redacted] Wolfgang Richter This is cool! I haven't seen any other courses using github as a class page.
2 hours ago · Like

Wolfgang Richter So, remember when I said I won't coddle? I'm exposing them to all real tools that will help them in life :p They're going to get git, svn, and other things exposed to them! Also, I reference original design documents (RFC's in this case).
2 hours ago · Like

[redacted] When I took my undergrad networking in Korea, TAs were not kind at all. They basically gave me a link to Beej's guide to network programming and some other link to RFC documents. I didn't like that at that time, but now I think that sometimes we need to let them learn by themselves with some initial pointers (+ Google!), because that's what they will do in the field sooner or later..
about an hour ago · Like

Wolfgang Richter Yep, no coddling :P I present syscalls to them, and point out all the error conditions so they handle them, but then they need to read man pages and get dirty writing code. And yeah, the end of the slide set is links---one to Beej's as well!
about an hour ago · Like

Wolfgang Richter [redacted] this is also a tricky point in my mind. We want to teach them certain concepts, looking up things is necessary no matter what. But I think that being lazy as a course staff, or not pointing them to material to specifically make them look things up is teaching the wrong thing. I'd _much_ rather have students be able to immediately have the material they need to focus on then learning and applying the systems concepts.
about an hour ago · Like

[redacted] What a great TA you are! :) I've witnessed some geek- and otaku-like CS guys... I bet some of them can go beyond (or love) Wolf's high standard. :-)
about an hour ago · Like

Wolfgang Richter My main goal is to lead by example :). I show them the important things to think about, and how to find it, and then let them build complex things with the building blocks I start them with.

Actually, yes; yes we are.

But we can have fun too!

Case Study: from Industry

Imagine you are a police officer on the road at 2:14am and there is a van weaving around on the road in front of you and it is obviously full of people who are dressed alike and have many different pieces of luggage and other objects that you can't quite see clearly in the vehicle.

Mission critical applications?

You reach over to your laptop mounted in your police car and key in the make model and license number and state. The message on the screen says: Sorry, this application is not available for the next 30 minutes.

Just a short outage?

You can imagine any possibility:

- a church group on an outing, someone driving getting tired
- a drunk driving his family
- a smuggler bringing illegal aliens
- a smuggler bringing in illegal drugs
- or for the fun of it, a group of terrorists preparing a major attack

What do you do now?

You have no information from a mission critical system with access to all sorts of databases including who the owners of the vehicle are, who might have leased it, rented it, or is the main driver for it, and intelligence data bases that can tell you if it might be illegals or terrorists, or some other sort of criminal. As a single police officer who has to deal with this now, when backup is 20 minutes away and you do not know if these people will be armed and desperate enough to shoot first.

Why?

The reason the data is not available is that there was no backup that took over immediately on failure. We deal with this in many ways, short term with clustering servers, having mirrors or RAID for data, should data storage fail, network redundancy should a network connection fail. However, there may also be a deeper reason for the failure which is dealt with in another way; disaster recovery and above that COOP or Continuance Of OPerations should a major disaster occur.

Netflix and the Chaos Monkey

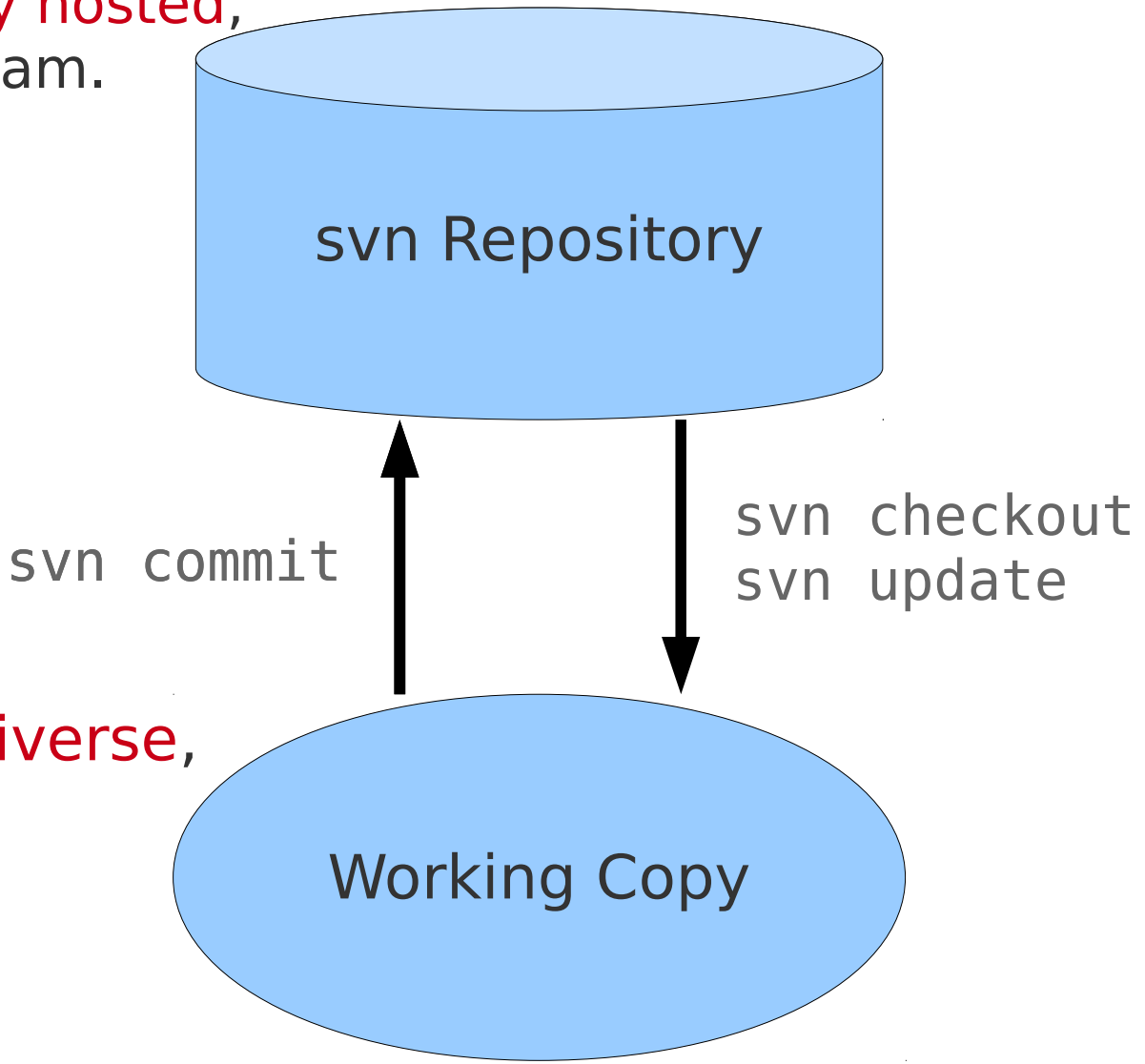
- One of a few surviving AWS/EC2 outages
- Randomly fail running instances
- Randomly fail processes
- Randomly introduce network failures
- Execute rarely encountered code paths
- On the production system

We'll be focusing on:

Backups
Code Documentation
Source Control Management
Code Structuring and Modularity

Subversion: svn

Usually remotely hosted,
shared with a team.



Your private universe,
before commit.

Getting started with svn

Roll your own:

- `svnadmin create repo`
- `svn checkout file:///.../svn/repo .`

Not your own:

- `svn checkout https://...`

Types of Repositories

Type	Meaning
file:///	Direct local disk access
http://	WebDAV protocol via Apache
https://	HTTP with SSL
svn://	Access svnserve via custom protocol
svn+ssh://	Same as svn://, via ssh tunnel

Daily workflow with svn

- Check for any remote updates
- Do your work
- Test your work
- Check differences, try to isolate changes
- Check for any remote updates
- Commit your work

Translated to svn commands

- svn update
  - Checks for new commits in remote repository
- vim, emacs, make, create, magic, etc.
- make test (run your changes!)
- svn status
  - See all changed files
- svn diff
  - Understand differences line by line (like diff util)
- svn update
- svn commit -m 'Isolated changes x and y'

Example repository

```
> svn update
At revision 0.
> echo 'this is a test' > test
> svn add test
A         test
> svn commit -m 'added a test'
Adding         test
Transmitting file data .
Committed revision 1.
> svn update
At revision 1.
> echo 'new text!' > test
> svn status
M       test
> svn diff
Index: test
===================================================================
--- test   (revision 1)
+++ test   (working copy)
@@ -1 +1 @@
-this is a test
+new text!
> svn update
At revision 1.
> svn commit -m 'updated text'
Sending        test
Transmitting file data .
Committed revision 2.
```

Dealing with collaboration

- Lock-modify-unlock
  - `svn lock <path>`
  - Modifications...
  - `svn unlock <path>`
- Copy-modify-merge
  - Optimistically edit things
  - Merge any files that others also commit

Locking in Practice

```
1> svn lock test
'test' locked by user 'wolf'.

2> svn status --username 'blockme' -u
     0           2   test
Status against revision:      2
2> echo 'i can have lock!?' > test
2> svn commit --username 'blockme' -m 'block on lock?'
Sending        test
Transmitting file data .svn: Commit failed (details follow):
svn: User blockme does not own lock on path '/test' (currently
locked by wolf)

1> echo "maybe if you're nice..." > test
1> svn commit --username 'wolf' -m 'finished with test'
Sending        test
Transmitting file data .
Committed revision 3.
```

Merging in Practice

```
2> svn update
Conflict discovered in 'test'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

Merging options

Type	Meaning
postpone	Mark conflict for later resolve
diff-full	Show all changes
edit	Use an editor to resolve
mine-conflict	Accept your version only
theirs-conflict	Accept their version only
show	Show more/all options

svn Resources

1) svn help

2) svn book
http://svnbook.red-bean.com/en/1.6/svn-book.html

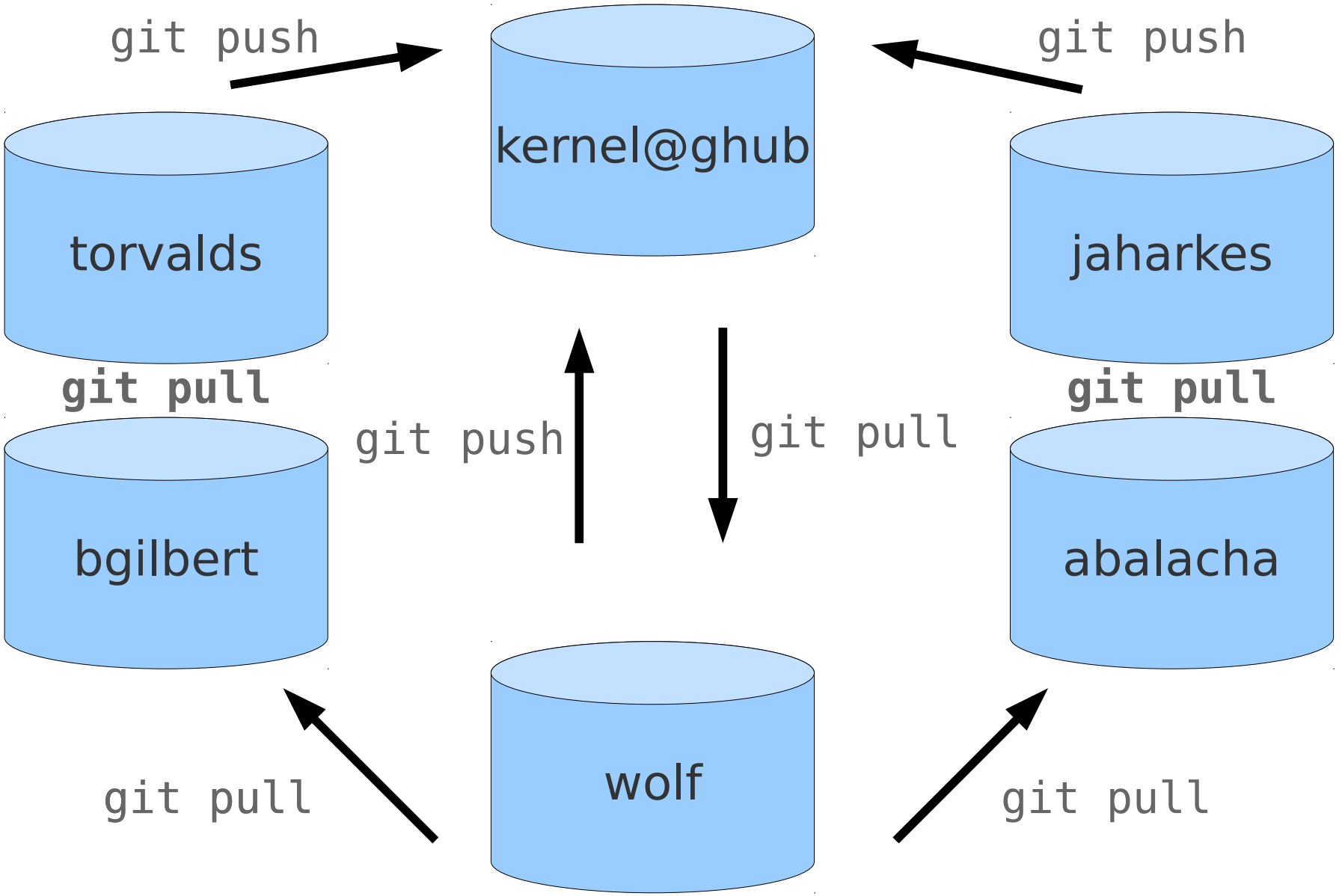
3) svn Cheatsheet
http://www.addedbytes.com/cheat-sheets/subversion-cheat-sheet/

git

- Fun, fun, fun!
- You already use it right? GitHub?
- If not, let's go down the rabbit hole...
- Resume padding
- Resume creator!

git

No notion of “working copy”–each is a full repository.



Getting started with git

Roll your own:

- `git config --global user.name "Wolfgang Richter"`
- `git config --global user.email "wolf@cs.cmu.edu"`
- `git init .`

Not your own:

- `git config --global user.name "Wolfgang Richter"`
- `git config --global user.email "wolf@cs.cmu.edu"`
- (1) `git clone git://git.kernel.org/pub/scm/git/git.git`
- (2) `git remote add origin git@github.com:username/Hello-World.git`

Types of Repositories

Type	Meaning
rsync://	rsync client to git repo
http://	HTTP hosted repo
https://	HTTP with SSL
git://	Special git server/protocol
ssh://	git via ssh

Daily workflow with git

- Check for any remote updates
- Do your work
- Test your work
- Check differences, try to isolate changes
- Commit your work; repeat as needed
- Check for any remote updates
- Push changes, or submit pull request

Translated to git commands

- git pull
  - Checks for new commits in remote repository
- vim, emacs, make, create, magic, etc.
- make test (run your changes!)
- git status
  - See all changed files
- git diff
  - Understand differences line by line (like diff util)
- git add
  - Stage changes, potentially line by line
- git commit -m 'Isolated changes x and y'
- git push

Example repository

```
> git pull
Already up-to-date.
> git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   src/recitation2.odp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   src/liso.c~
no changes added to commit (use "git add" and/or "git commit -a")
> git diff
diff --git a/recitation2/src/recitation2.odp b/recitation2/src/recitation2.odp
index d3289ed..9a1fec3 100644
Binary files a/recitation2/src/recitation2.odp and
b/recitation2/src/recitation2.odp differ
```

Example repository

```
> git add src/recitation2.odp
> git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   src/recitation2.odp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   src/liso.c~
> git commit -m 'working on git section, also hello class!'
[master 73d717b] working on git section, also hello class!
 1 files changed, 0 insertions(+), 0 deletions(-)
> git push
Counting objects: 12, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 653.27 KiB, done.
Total 11 (delta 3), reused 0 (delta 0)
To git@github.com:theonewolf/15-441-Recitation-Sessions.git
   2e7a763..73d717b  master -> master
```

Dealing with collaboration

- Branching – Cheap and effective
  - Make development or feature branches
  - Rebase and merge when features complete
  - `git branch https`
  - `git checkout https`
  - …
- Pull/Push with Merge
  - Standard model: pull requests
  - Or push into “central” repository

Merging in Practice

```
> git pull
Auto-merging test
CONFLICT (content): Merge conflict in test
Automatic merge failed; fix conflicts and then commit the result.
> cat test
<<<<<<< HEAD
helloX world
=======
helloY world
>>>>>>> 29a240d5017c73ca4f78466afcf1fd5b8f46808f
```

Choose how to merge—yours or other author's.

Finalize, commit, then push, or request a pull.

svn vs git

svn	git
Costly branching	Cheap local branching
No local repository	Everything is local
Large repos are slow	Designed for very large repos
Centralized	Distributed
Restricted workflows	Multiple workflows

http://whygitisbetterthanx.com/

git's Superiority

It just is. Trust me.

svn: every commit makes trusted code untrusted

git: commit, commit, commit...; rebase; merge

Extra git tips

- `gitk` – GUI to interact with git repository
- `git svn` – Use git on top of svn...
- `git branch` – create dev branches
- `git tag` – create release tag
- `git bisect` – binary search for bad commit
- `git rebase` – forward-port local commits

git Resources

1) man gittutorial

2) Git User's Manual
http://www.kernel.org/pub/software/scm/git/docs/user-manual.html

3) Git Immersion
http://gitimmersion.com/

4) Git Cheatsheet
https://github.com/AlexZeitler/gitcheatsheet

Code Structure: GNU make

- Recipes for your code
  - Compilation
  - Installation
  - Cleanup
  - Testing
- Composed of a series of
  - targets [the recipes]
  - which have dependencies
  - and commands
  - also, variables...

Makefile from recitation1

```
##############################################################################
# Makefile                                                                   #
#                                                                            #
# Description: This file contains the make rules for Recitation 1.           #
#                                                                            #
# Authors: Athula Balachandran <abalacha@cs.cmu.edu>,                        #
#          Wolf Richter <wolf@cs.cmu.edu>                                    #
#                                                                            #
##############################################################################

SERVER_SRC = echo_server.c
CLIENT_SRC = echo_client.c
OPTIONS = -Wall

default: echo_server echo_client

echo_server:
    @gcc $(SERVER_SRC) -o echo_server $(OPTIONS)

echo_client:
    @gcc $(CLIENT_SRC) -o echo_client $(OPTIONS)

clean:
    @rm echo_server echo_client
```

GNU make Resources

1) GNU make Manual
http://www.gnu.org/software/make/manual/html_node/index.html

Wanna be “official”?

- Learn GNU autotools
- Standardize workflow with
  - Generation of ./configure script
  - Generation of Makefiles and make install target
  - Generation of helper scripts
- Preparing to release open source?
  - (1) Pick a license
  - (2) autotoolize
  - (3) GitHub !

Project 1 Sneak Peek

```c
#include <stdio.h>
#include <stdlib.h>
#include "select_engine.h"
#include "http_parser.h"
#include "logging.h"

#define USAGE "\nUsage: %s <PORT> <LOG_FILE> <LOCK_FILE>\n\n"

int main(int argc, char* argv[])
{
        int port;
        char* flog, * flock;

        if (argc < 4)
        {
                fprintf(stdout, USAGE, argv[0]);
                return EXIT_FAILURE;
        }

        port = atoi(argv[1]);
        flog = argv[2];
        flock = argv[3];

        struct select_engine engine;
        liso_engine_create(&engine, port, flog, flock);
        liso_engine_register_http_handler(&engine, parser_http_handler);
        liso_engine_register_http_disconnect_handler(&engine, parser_disconnect_handler);

        liso_logging_log("liso", "main", "Starting Liso server on port %d", port);

        return liso_engine_event_loop(&engine);
}
```

GitHub:

Git it, got it, good.

git clone git://github.com/theonewolf/15-441-Recitation-Sessions.git