Computer System Organization Recitation [Fall 2017]

CSCI-UA 201-006

R1: Introduction

Know your staff

- Instructor: Prof. Zhaoguo Wang
- Recitation Instructor: Chien-Chin Huang (Jien-Jin)(me)
  - huang@cs.nyu.edu
  - Office Hour: Thu 3-4pm (60 5th Ave Office Rm 406)
- Instructional Assistants:
  - Chien-Chin Huang(me)
  - Hung-Wei Chen

Important URLs

- https://github.com/nyu-cso-17fall/cso17-recitation
- https://github.com/nyu-cso-17fall/cso17-labs
- Be sure to sign up for Piazza.

What is this recitation for?

- Exercises that will help you understand CSO more.
- Tutorial of labs.
- Review of midterm I and II.

How are we going to proceed?

- Problems driven.
    - Except for today.

- If you are confident that you can solve all the exercises and labs by YOURSELF and don’t need extra guides through the course, you can skip the recitation.

- Exercises will be posted every Tuesday night and the deadline will be Thursday afternoon or night.
    - Except for today…
    - The deadline for the first recitation is 9/11 7:00pm.

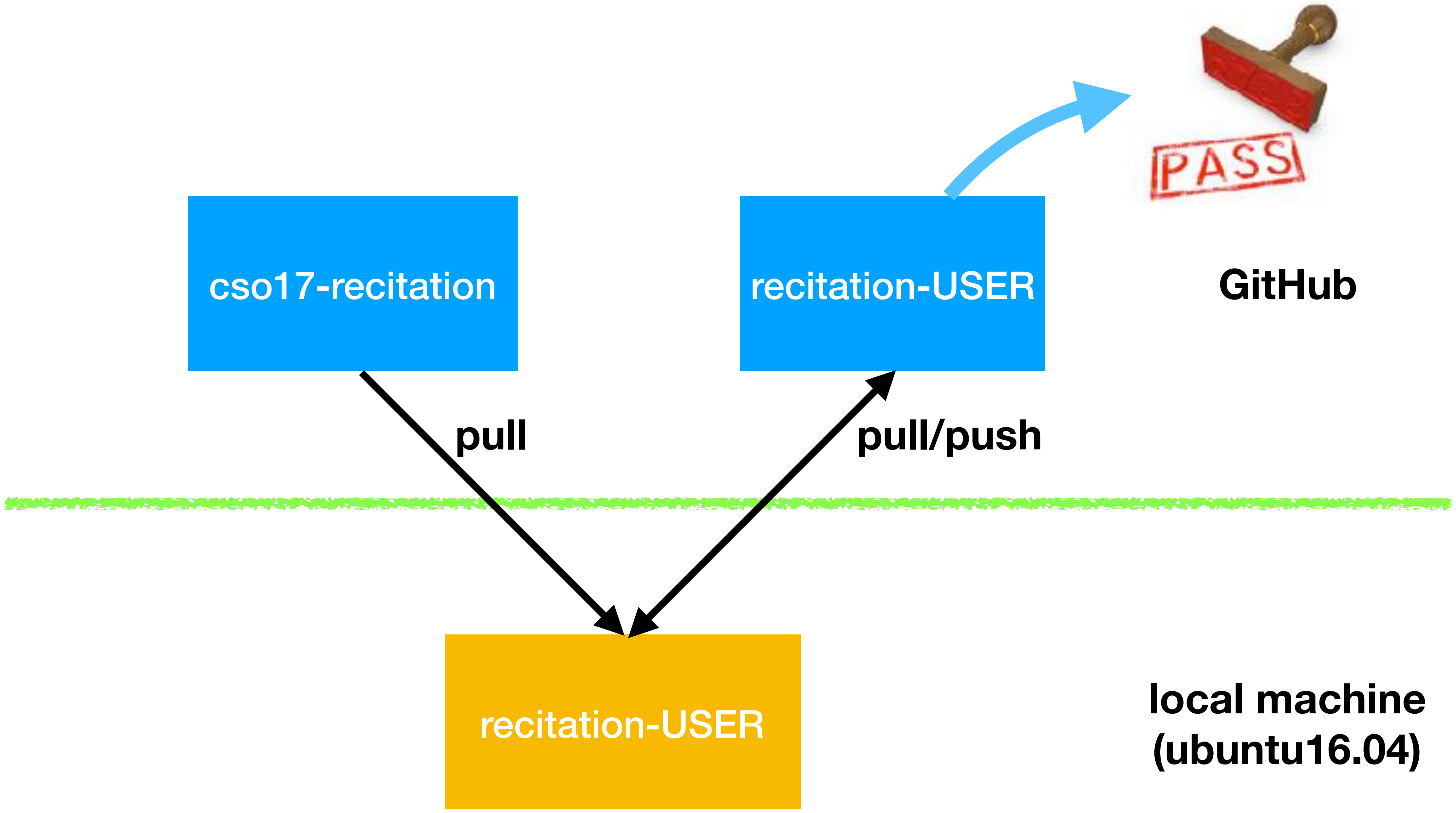
Let’s do some statistics

- How many of you have never used Unix-like systems?
  - How many of you have never used command line?
- How many of you have never programmed in C or C++?
  - How many of you have never programmed in Python?
- How many of you have never used version control softwares?
  - How many of you have never used Git?

Let’s begin

- How to setup your repo and submit your code?
- Unix/command line
- Program development
  - Editor (vim)
  - Compile
  - Multi-task environment (tmux)
  - Debug
  - Version control (Git)
- **Goal:**
  - **Setup your recitation-USERNAME repo.**
  - **Submit modified Makefile, fixed foo.c and hello.c.**

Git status for our recitation and labs



cso17-recitation

recitation-USER

GitHub

pull

pull/push

recitation-USER

local machine
(ubuntu16.04)

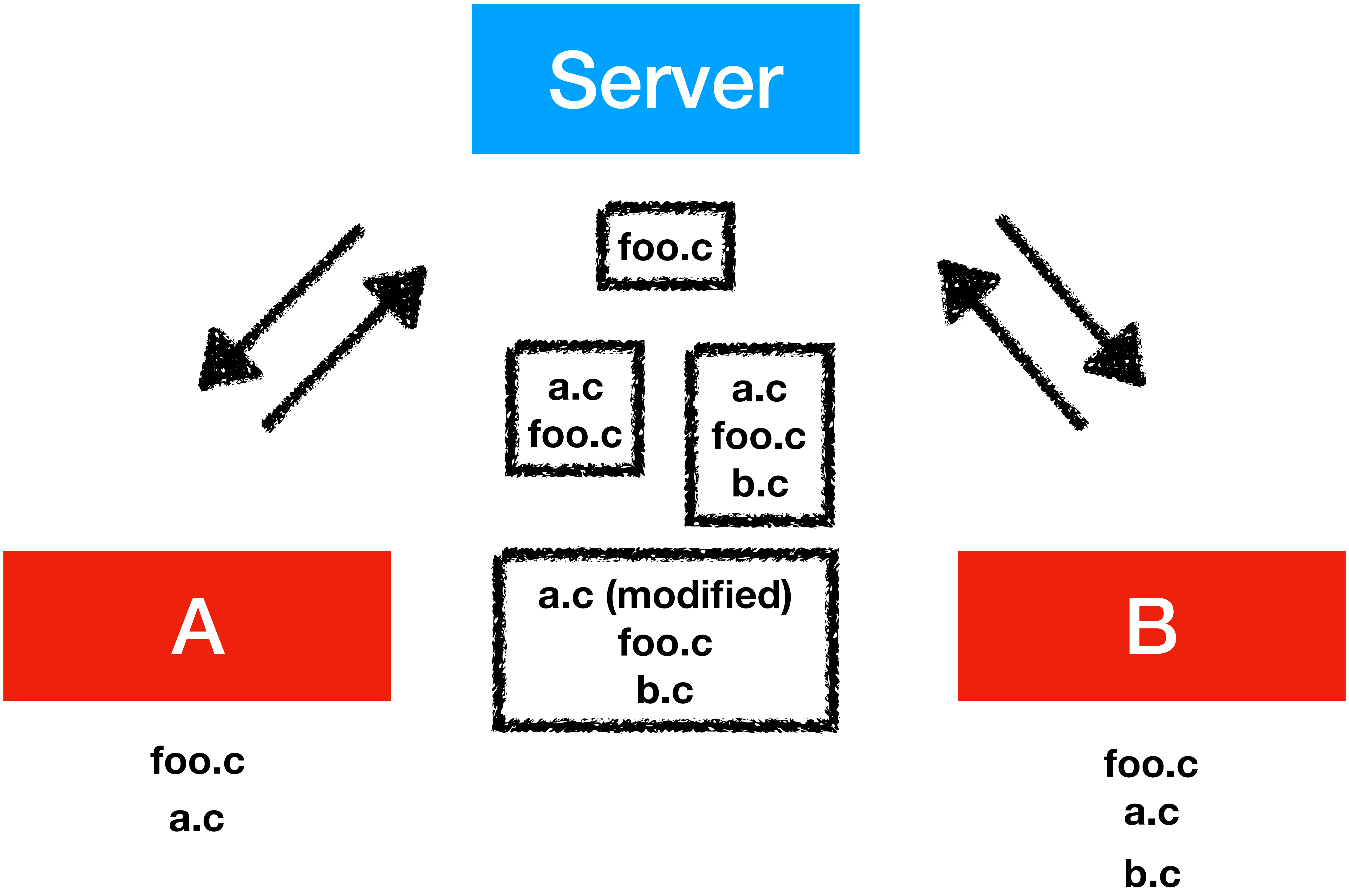
Version control system

- What?
  - Manages changes to documents, source files and other collections of information.
- Why?
  - Do you remember which source file you added/modified last week? Probably not.
  - Have you ever developed a project with other people? Coordinating programmers is hard.
- How?
  - CVS, SVN and GIT

Server/client version control system

- What?
  - A kind of version control system that puts all tracking metadata on a server. Clients can fetch/upload source files and information from the server.
- Why?
  - Strait-forward and easy to maintain.
  - Save space.

Version control system



Server

foo.c

a.c
foo.c

a.c
foo.c
b.c

a.c (modified)
foo.c
b.c

A

foo.c
a.c

B

foo.c
a.c
b.c

Branching

- What?
  - The duplication of an object under version control so that modifications can happen in parallel along both branches.
- Why?
  - Developing different features in a same project.

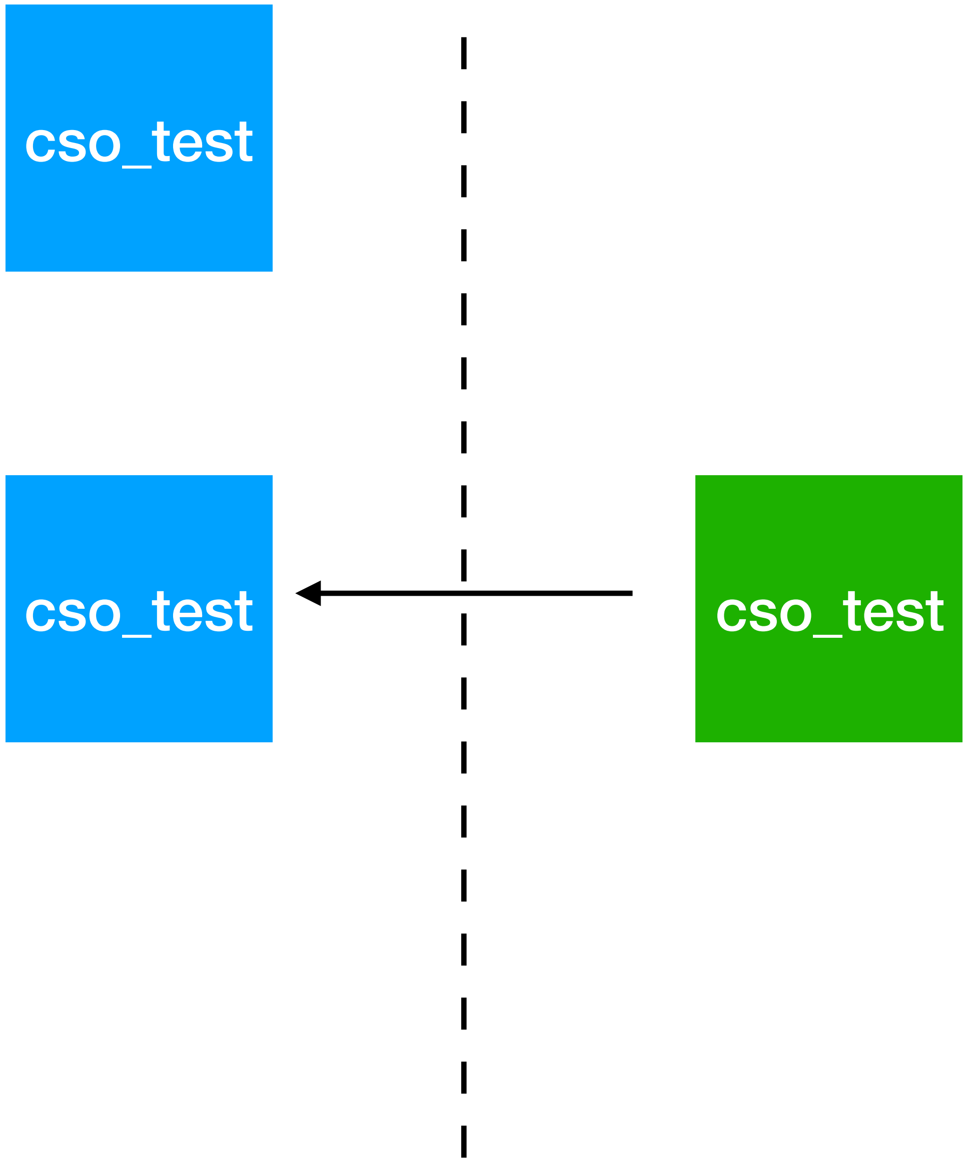
Distributed version control system

- What?
  - There is no “server”. Every client owns a complete repository locally (local repository) and can sync(push/pull) with any other remote repositories.
- Why?
  - There are hundreds or more projects and thousands or more developers in Linux community.
    - Coordinating the development using one single server is difficult.
  - Can work without network.

Git — initialization

- git init

- git clone
    - git remote -v



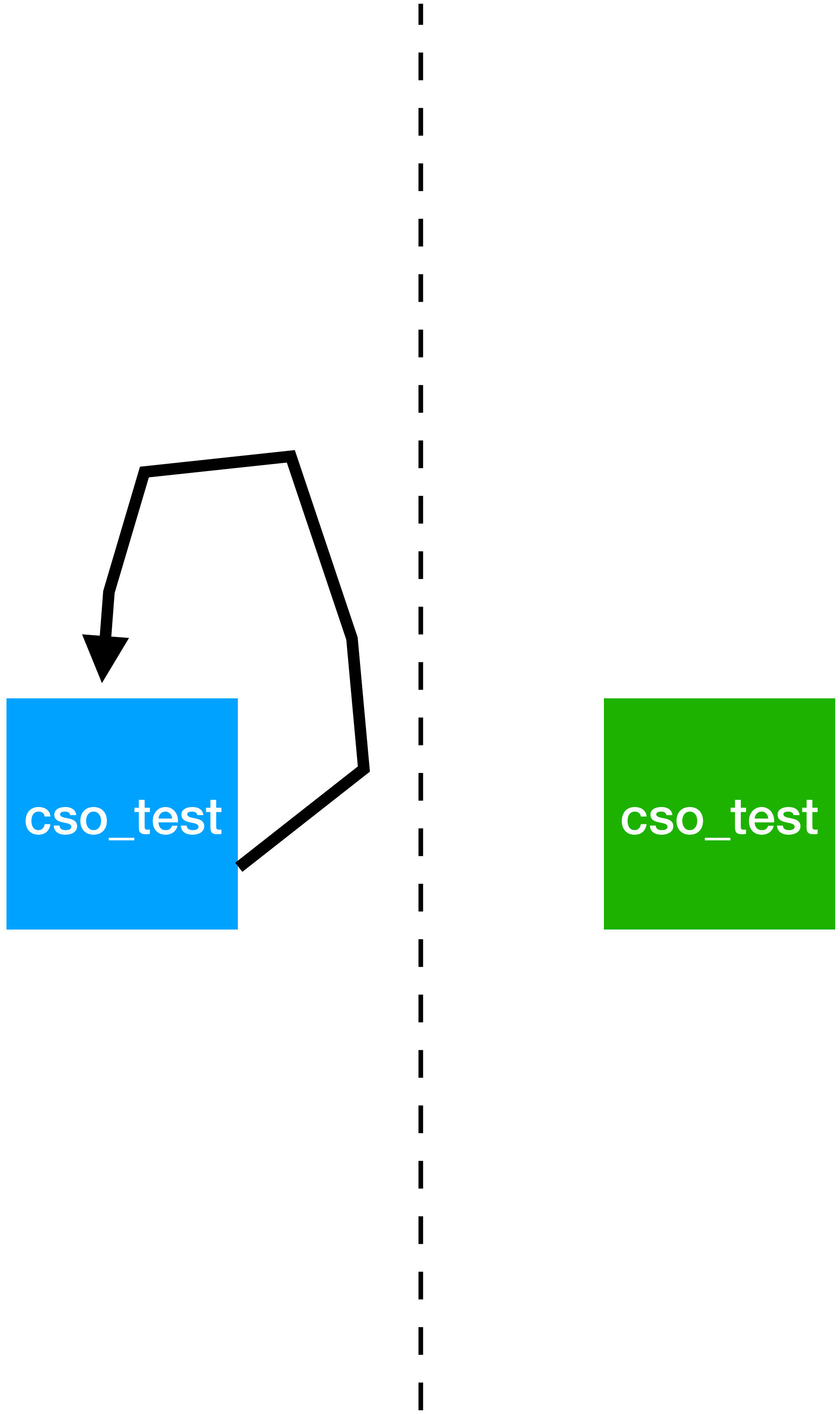
cso_test

cso_test

cso_test

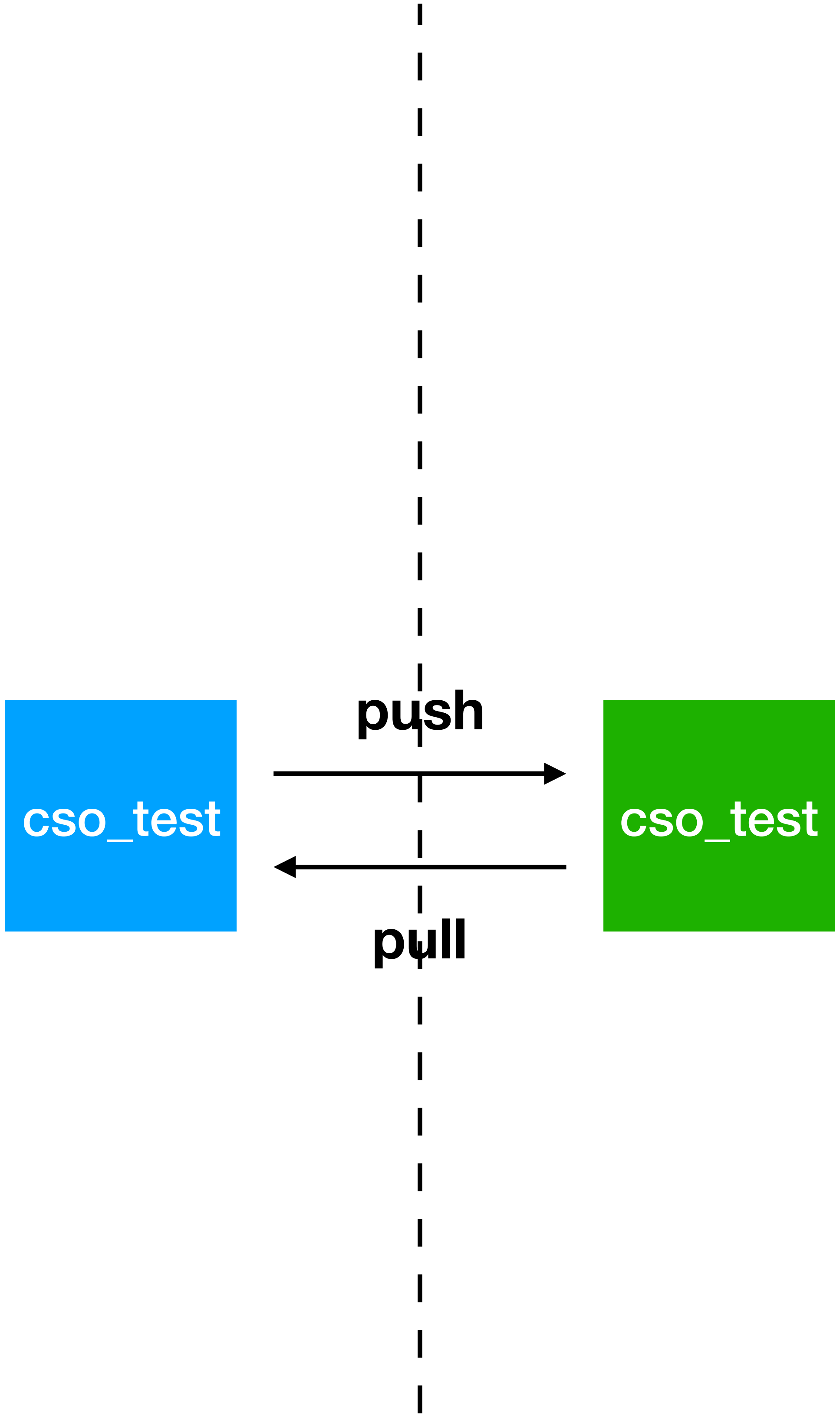
Git — commit

- git commit -m “comment”
- git add FILES
- git rm FILES
- git log



Git — remote

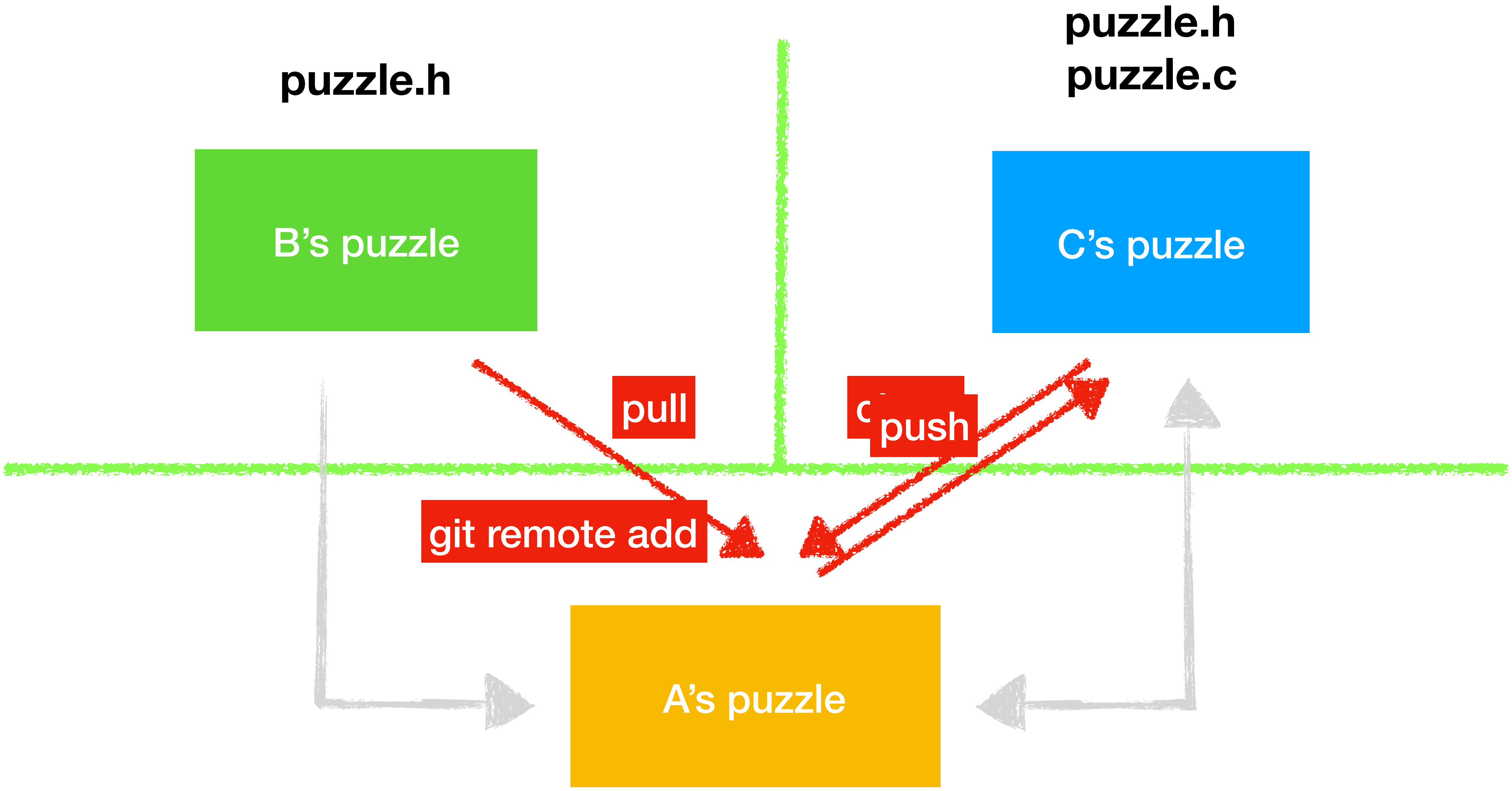
- git remote -v
- git push
  - git push origin master
- git pull
  - git pull origin master



Git — checkout/branch

- git checkout -b BRANCH_NAME
- git checkout HASH
  - git log
- git branch -a

Git status for our recitation and labs



puzzle.h

B's puzzle

puzzle.h
puzzle.c

C's puzzle

pull

c

push

git remote add

A's puzzle

puzzle.h

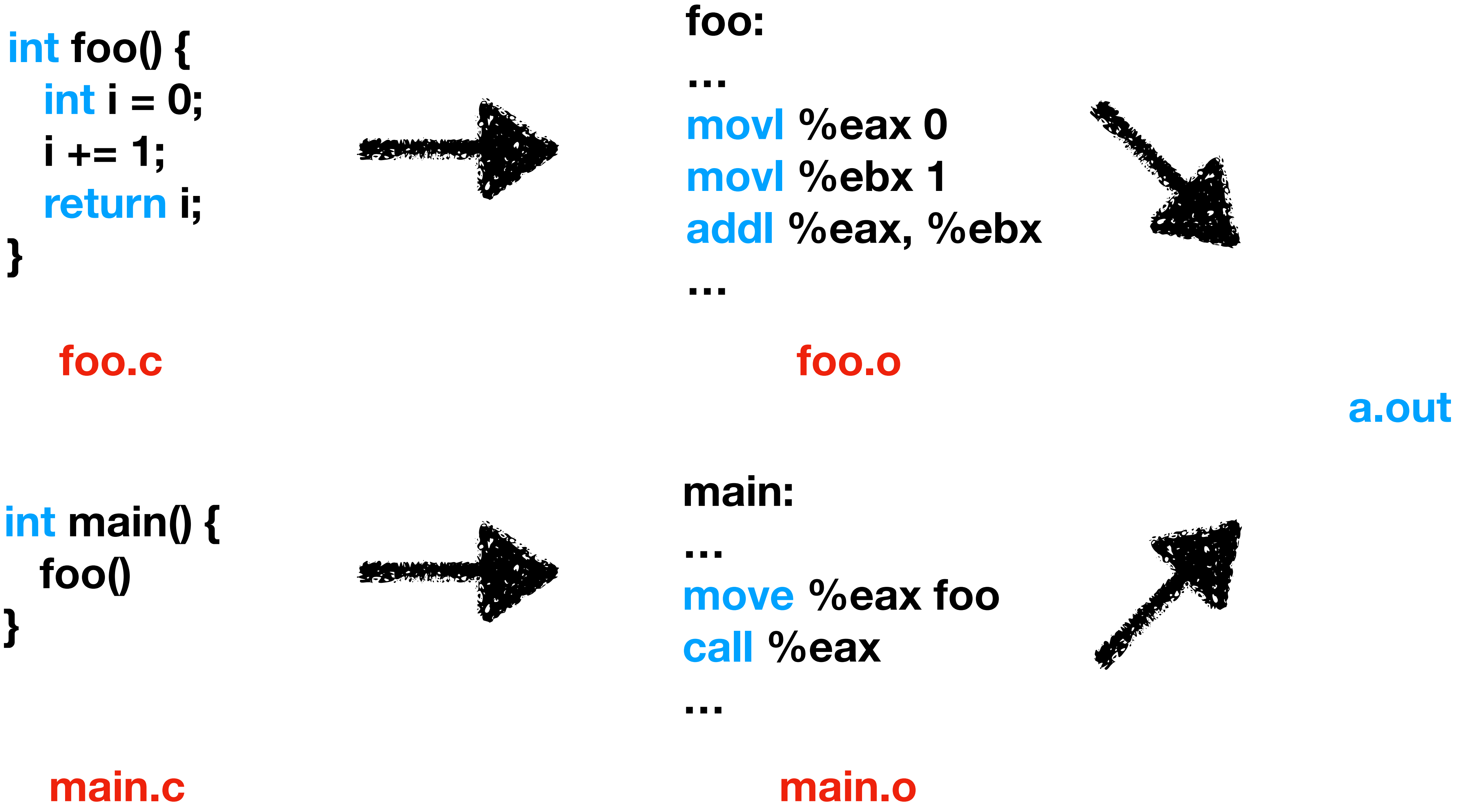
puzzle.c

- https://github.com/nyu-cso-17fall/cso17-recitation

Compilation

- Compiler
  - A software that transforms computer code written in one programming language (the source language) into another computer language (the target language).
  - gcc / cc
- Interpreter
  - A software that directly executes instructions written in a programming/ scripting language without previously compiling them into a machine language program.
- Linker
  - A linker is a software that takes one or more object files and combines them into a single executable file, library file, or another object file.
  - gcc / ld

Compilation



```
int foo() {
   int i = 0;
   i += 1;
   return i;
}
```

foo.c

```
foo:
...
movl %eax 0
movl %ebx 1
addl %eax, %ebx
...
```

foo.o

a.out

```
int main() {
   foo()
}
```

main.c

```
main:
...
move %eax foo
call %eax
...
```

main.o

Make

- What is Make?
  - A build automation tool that automatically builds executable programs and libraries from source code.
- Why?
  - A project can contains a lot of source files.
  - Each source file may needs different compiler option.
  - Dependencies exists among source files.
- How?
  - Describe everything in a file, Makefile, and Make will do everything for you.

Make with automatic variables

- Why automatic variables?
  - Specifying how to compiler everything does not remove our burdens.
  - Automatic variables can help us unify the same type of files.
- How?
  - $@ (target name)
  - $^ (name of all pre-requisites)
  - pattern-matching using % and *.

Debug

- How to debug?
  - Print logs, observe and then debug.
  - Use a debugger to help you.
    - gdb
- How to use gdb?
  - First you need to ask gcc to add debug information when compiling the source files.
  - Debug!