

SPAN 301 – SECCIÓN 1

Introducción

Nuestros Objetivos

- Introducción
- Aprender a decifrar palabras desconocidas
- Entender las diferencias entre diferentes dialectos del lenguaje española
- Confundir los estudiantes profundamente y hacerles pensar que estamos en una clase de español

CSE 333 – SECTION 1

Git Setup & Pointers

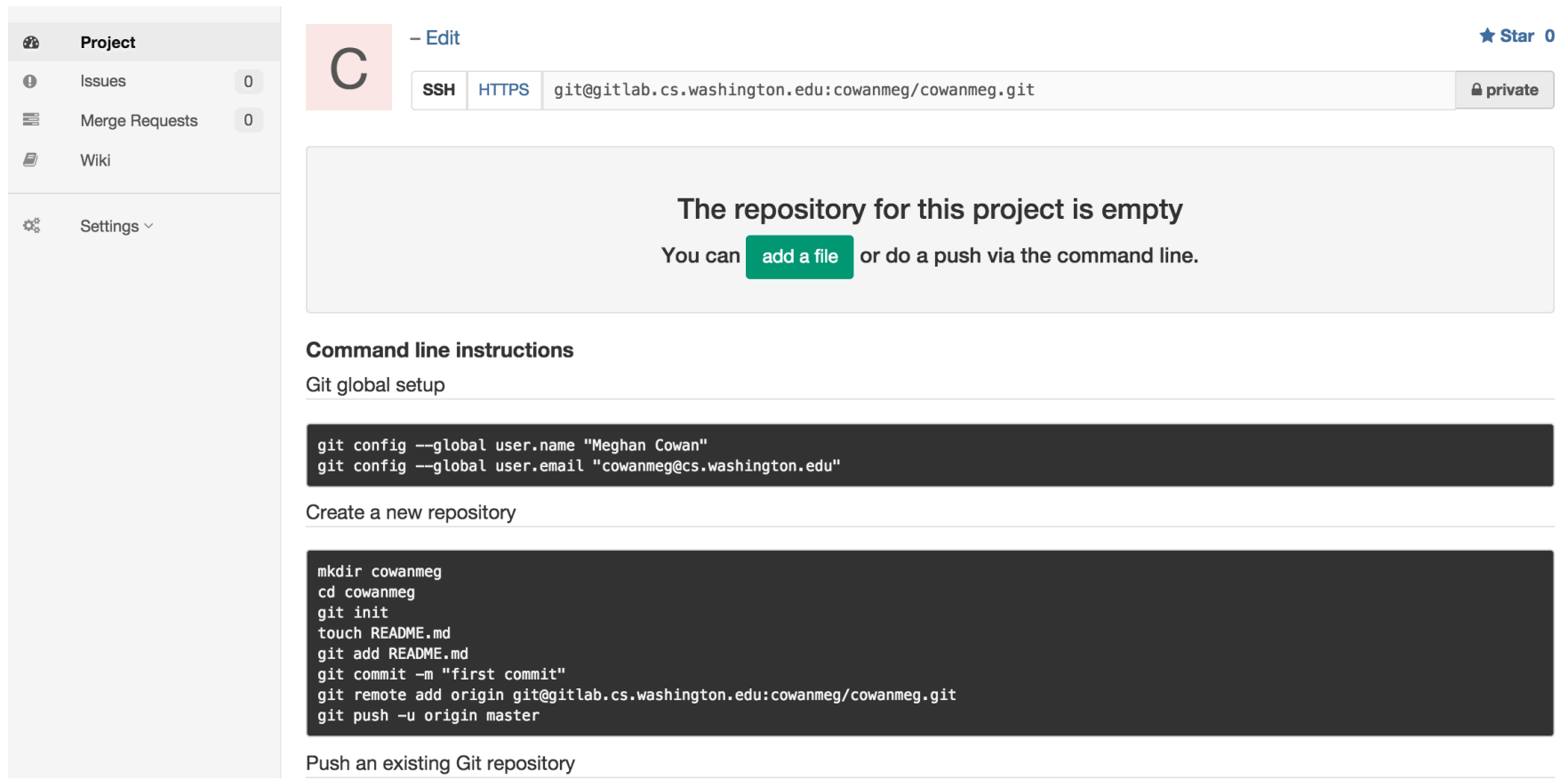
Your TAs

- Phillip Dang, Josh Nazarian, Sixto (Joshua) Rios, Soumya Vasisht, Qingda (Bruce) Wen, Zhitao (Reid) Zhang.
- Office hours are posted
 - everyday 4-5 pm starting today!
 - CSE006
- Staff Email
 - cse333-staff@cs.washington.edu
- Please use the discussion board!

Gitlab Intro - Sign In

- Sign In using your **CSE netID**
- <https://gitlab.cs.washington.edu/>

- Most of you should have repos created for you



Project

- Issues 0
- Merge Requests 0
- Wiki
- Settings

C - Edit ★ Star 0

SSH HTTPS git@gitlab.cs.washington.edu:cowanmeg/cowanmeg.git 🔒 private

The repository for this project is empty

You can [add a file](#) or do a push via the command line.

Command line instructions

Git global setup

```
git config --global user.name "Meghan Cowan"
git config --global user.email "cowanmeg@cs.washington.edu"
```

Create a new repository

```
mkdir cowanmeg
cd cowanmeg
git init
touch README.md
git add README.md
git commit -m "first commit"
git remote add origin git@gitlab.cs.washington.edu:cowanmeg/cowanmeg.git
git push -u origin master
```

Push an existing Git repository

SSH Key Generation

- Step 0: Check if you have a key
 - Run `cat ~/.ssh/id_rsa.pub`
 - If you see a long string starting with ssh-rsa or ssh-dsa go to Step 2.
- Step 1: Generate a new SSH key
 - Run `ssh-keygen -t rsa -C "$your_e-mail"` to generate a new key.
 - Click enter to skip creating or a password
 - git docs suggest creating a password, but it's overkill for 333 and complicates operations
- Step 2: Copy SSH key
 - run `cat ~/.ssh/id_rsa.pub`
 - Copy the complete key starting with ssh- and ending with your username and host
- Step 3: Add SSH key to gitlab
 - Navigate to your ssh-keys page (In the top menu bar click on profile then SSH Keys in the side menu)
 - Click the green 'Add SSH Key' button in the right corner.
 - Paste into the Key text box and give a Title to identify what machine the key is for.

First Commit

- **git clone <repo url from project page>**
Clones your repo
- **touch README.md**
Creates a file called README.md
- **git status**
Prints out the status of the repo.
Should see 1 new file README.md
- **git add README.md**
Stages a new file/updated file for commit.
git status: README.md staged for commit
- **git commit -m "First Commit"**
Commits all staged files with the comment in quotes.
git status: Your branch is ahead by 1 commit.
- **git push**
Publishes the changes to the central repo.
You should now see these changes in the web interface.
 - Might need **git push -u origin master** on first commit (only)

References

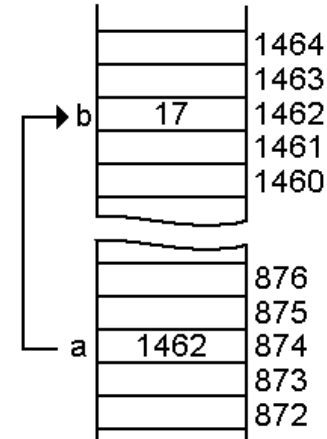
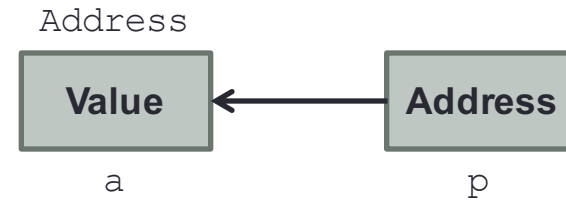
- SSH Key generation:
<https://gitlab.cs.washington.edu/help/ssh/README.md>
- Basic Git Tutorial:
<http://courses.cs.washington.edu/courses/cse333/16sp/hw/git.html>

Quick Refresher on C

- General purpose programming language
- Procedural
- Often used in low-level system programming
- Supports use of pointer arithmetic
- Provides facilities for managing memory
- C passes all of its arguments by value
 - Pass-by-reference is simulated by passing the address of a variable

Pointers

- A data type that stores an address
- Used to indirectly refer to values
- Can add to or subtract from the address
 - It's just another number



Example

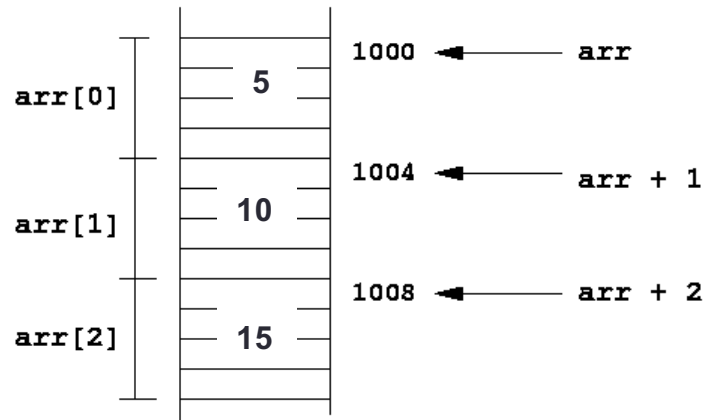
[basic_pointer.c]

```
#include <stdio.h>
void f(int *j) {
    (*j)++;
}
int main() {
    int i = 20;
    int *p = &i;
    f(p);
    printf("i = %d\n", i);
    return 0;
}
```

Arrays and pointers

- `arr[0] <==> *arr`
- `arr[2] <==> *(arr + 2)`

- How about `arr`, `arr+2`,
`*arr+2` or `*arr++`?



Output parameters

- C parameters are pass-by-value
- What if you want to modify a passed in parameter?
 - Why would this be useful in the first place?
 - Multiple return values

Output parameters

```
void make4_v1(int i) {  
    i = 4;  
}
```

```
void make4_v2(int *i) {  
    int j = 4;  
    i = &j;  
}
```

```
void make4_v3(int *i) {  
    *i = 4;  
}
```

See also: `[output_params.c]`

Pointers to pointers

45	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
	58			63		55			h	e	l	l	o	\0	

```
char *c = "hello";
```

```
char **cp = &c;
```

```
char ***cpp = &cp;
```

- Why could this be useful?

Function pointers

- We can have pointers to functions as well
- Syntax is a little awkward
 - Example: `int (*ptr_to_int_fn)(int, int)`
 - Makes sense if you think about it
- We will be using these in the homework assignments!
- Demo: `[function_pointer.c]`
`[buggy.c]`

Looking up documentation

- Don't go straight to Google / Stack Overflow / etc.
- Use the built-in man pages
 - `man <program/utility/function>`
 - `man -f <name>` or `whatis <name>`
 - `apropos <keyword>`
- Much more documentation is linked on the 333 home page
 - Under “Resources” on the left side of the page

Questions, Comments, Concerns

- Do you have any?
- Exercises going ok?
- Lectures make sense?