CSE 333 – SECTION 1

Git Setup & Function Pointers
Your TAs

- Meghan Cowan, Renshu Gu, Steven Lyubomirsky, Josh Rios, Nathan Wong, and Jack Xu

- Temporary office hours are posted
  - 3:00-4:00pm Thursday, 2:00-3:00pm Friday, 12:30-1:30pm Monday & TBD
  - CSE 007 Lab

- Staff Email
  - cse333-staff@cs.washington.edu

- Please use the discussion board!
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]

```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

- $\text{arr}[0] <\Rightarrow \ *\text{arr}$
- $\text{arr}[2] <\Rightarrow \ *(\text{arr} + 2)$

- How about $\text{arr}$, $\text{arr}+2$, $\*\text{arr}+2$ or $\*\text{arr}++$?
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

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: \texttt{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?
Gitlab Intro - Sign In

- Sign In using your CSE netID
- [https://gitlab.cs.washington.edu/](https://gitlab.cs.washington.edu/)

- Most of you should have repos created for you

---

The repository for this project is empty
You can [add a file](https://gitlab.cs.washington.edu/cowannmeg/cowannmeg.git) 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 "cowannmeg@cs.washington.edu"
```

Create a new repository

```
    mkdir cowannmeg
    cd cowannmeg
    git init
    touch README.md
    git add README.md
    git commit -am "first commit"
    git remote add origin git@gitlab.cs.washington.edu:cowannmeg/cowannmeg.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 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/cse333/17su/hw/git.html