

CSE 333 – SECTION 1

Git Setup & Function Pointers

Your TAs

- Renshu Gu, William Kim, Soumya Vasisht
- Initial office hours are posted
 - **When:** Mon – Fri, 12 – 1 pm
 - **Where:** CSE 006
- Staff Email
 - `cse333-staf[at]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

The screenshot shows the GitLab interface for a repository named 'C' (cowanmeg/cowanmeg.git). The left sidebar contains navigation links: Project, Issues (0), Merge Requests (0), Wiki, and Settings. The main content area shows the repository is empty, with a message: 'The repository for this project is empty. You can [add a file](#) or do a push via the command line.' Below this, there are sections for 'Command line instructions' and 'Create a new repository'. The 'Command line instructions' section includes 'Git global setup' with the following commands:

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

 The 'Create a new repository' section includes the following commands:

```
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
```

 At the bottom, there is a section for 'Push an existing Git repository'.

SSH Key Generation

Step 1a: 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 1b: Generate a new SSH key if necessary

- Run **ssh-keygen -t rsa -C "<netid>@cs.washington.edu"** to generate a new key
- Click enter to skip creating 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 (click on your avatar in the upper-right, then "Settings," then "SSH Keys" in the left-side menu)
- Paste into the "Key" text box and give a "Title" to identify what machine the key is for
- Click the green "Add key" button below "Title"

First Commit

- 1) **git clone <repo url from project page>**
 - Clones your repo
- 2) **touch README.md**
 - Creates an empty file called README.md
- 3) **git status**
 - Prints out the status of the repo: you should see 1 new file README.md
- 4) **git add README.md**
 - Stages a new file/updated file for commit. git status: README.md staged for commit
- 5) **git commit -m "First Commit"**
 - Commits all staged files with the provided comment/message.
git status: Your branch is ahead by 1 commit.
- 6) **git push**
 - Publishes the changes to the central repo. You should now see these changes in the web interface (may need to refresh).
 - 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:**

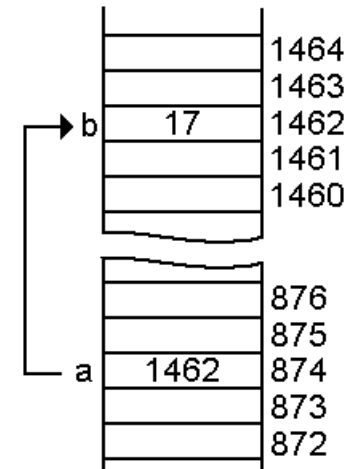
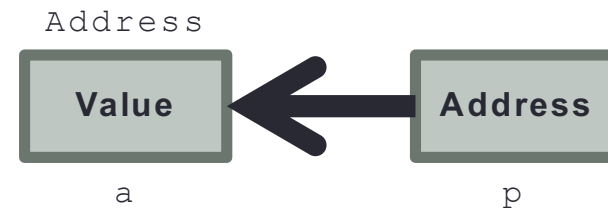
<https://courses.cs.washington.edu/courses/cse333/18su/hw/git.html>

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

- The name of an array can be used as a pointer to the first element in the array, and vice versa.

`arr[0] <==> *arr`

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

- How about `arr`, `arr+2`,

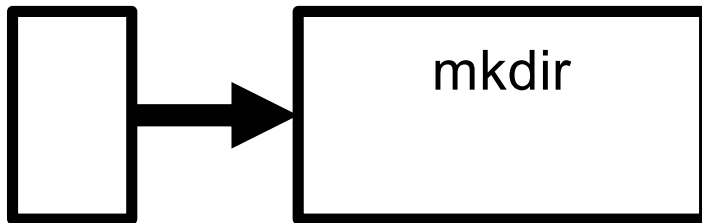
`*arr+2` or `*arr++`?

`*arr++` is treated as `*(arr++)` where the old value of `arr` is dereferenced but the pointer value of `arr` is incremented

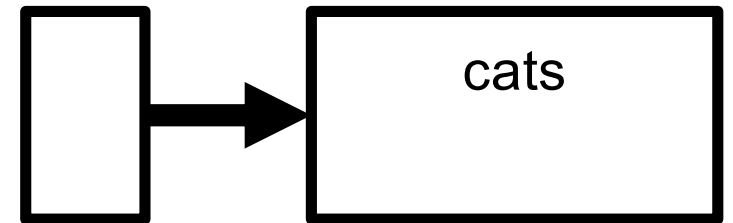
Arrays and pointers

- Given the following command: “mkdir -v cats dogs” and $\text{argv} = 0\text{x}1000$, draw a box-and-arrow memory diagram of argv and its contents for when mkdir executes.

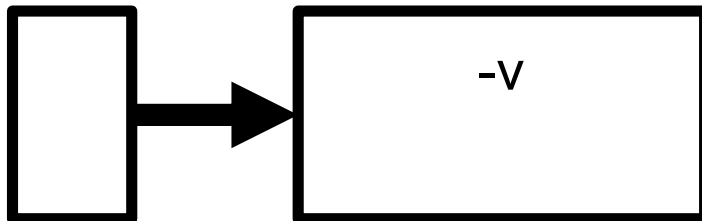
$\text{argv} = 0\text{x}1000$



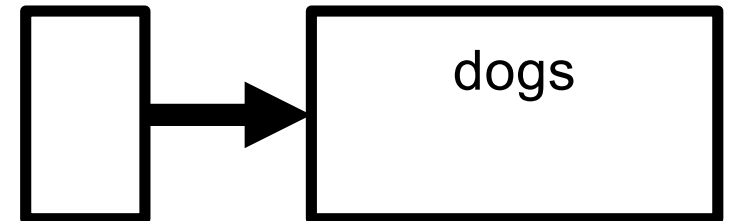
$\text{argv} + 2 = 0\text{x}1010$



$\text{argv} + 1 = 0\text{x}1008$



$\text{argv} + 3 = 0\text{x}1018$



- Using the same information from above, what values are returned by the following expressions?

`argv[0]` -> **address of the first character in "mkdir"**

`argv[0] + 1` -> **address of the second character in "mkdir"**

`argv[0][3]` -> **'i'**

`argv + 1` -> **0x1008**

`*(argv[1] + 1)` -> **'v'**

Output parameters

- 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
- We will be using these in the homework assignments!
- Syntax is a little awkward
 - Example: `int (*ptr_to_int_fn) (int, int)`
 - Makes sense if you think about it
- Demo: `[function_pointer.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?