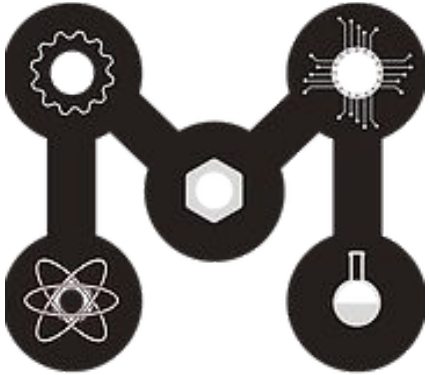
Git and GitHub Workshop

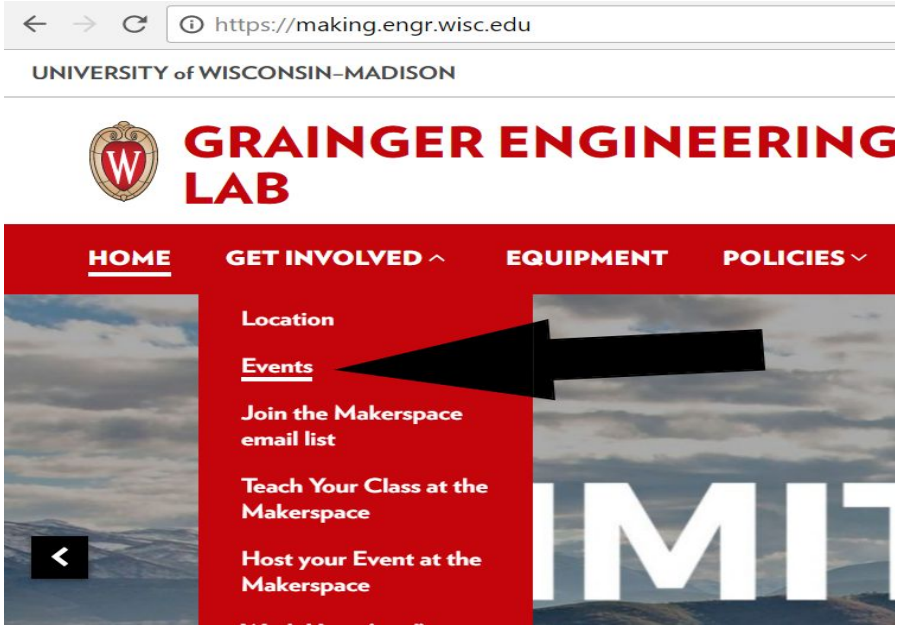
Be not afraid of Git

Hosted by:



More Workshops This Year!

https://making.engr.wisc.edu



FIRST THING'S FIRST!

- Git is **Form** of Version Control Software
  - AKA source control
  - Many other forms
- GitHub is a place to host your Git Repositories online
  - Other websites offer this too like BitBucket

Two Ways to use Git

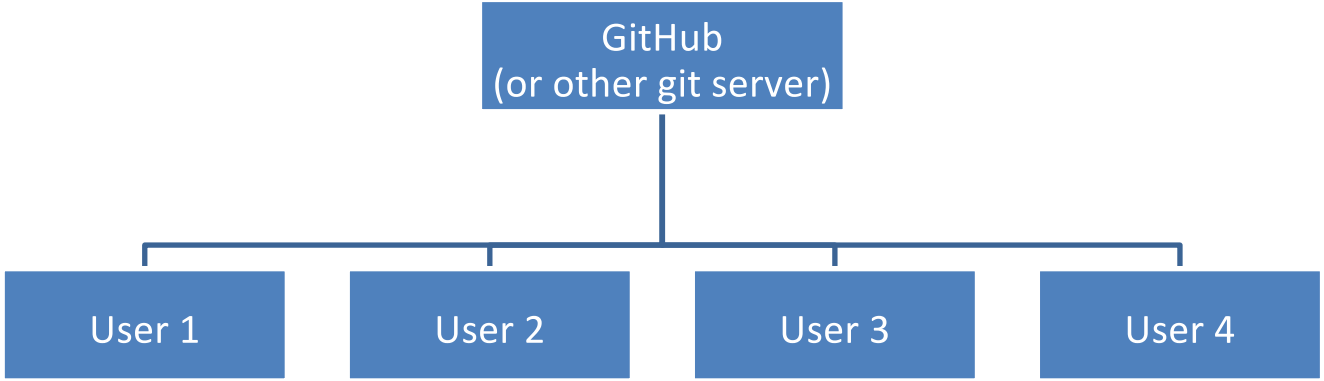
- Command Lines
  - **Pros**: You are really forced to understand Git
  - **Cons**: You are really forced to understand Git
- GUI Tools (GitHub Desktop, Git Extensions, etc)
  - **Pros**: Way easier to use and manage code
  - **Cons**: This is how you look to some people

Gotta Get Git

- Command Line
  - https://git-scm.com/downloads
    - Git Bash is just a Windows side application that opens a command prompt with Git
- GitHub Desktop GUI (what I recommend at first)
  - https://desktop.github.com/
- Make a GitHub account too!
  - Use Student email if possible

Git 101- What happens to code

- Repository (the code) is saved on local computer
- Repositories can then be **Pushed** to a remote server.
- Distributed Network



How Git Works

- Keeps tracks of difference in lines of files

9 demo/demo.cpp

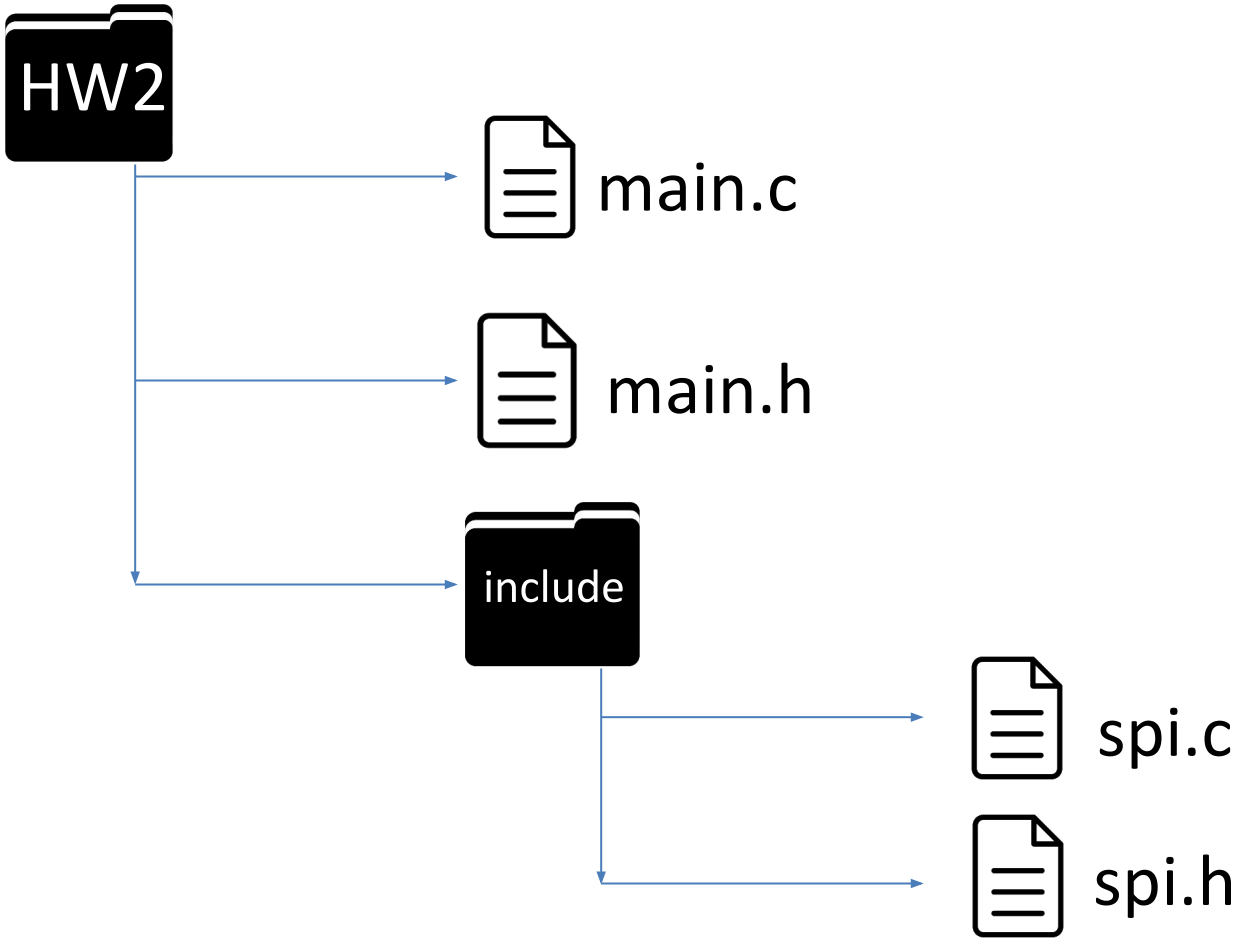
View

```diff
@@ -18,13 +18,18 @@ void on_new_test(char* body) {
 void onJoin(int uid){ printf("Join: %d\n", uid); }
 void onLeave(int uid){ printf("Leave: %d\n", uid); }

-int main() {

+int main(int argc, char* argv[]) {

+  if (argc < 3) {
+    printf("\n/demo <ip> <port>\n\n");
+    exit(1);
+  }
+

   WebSocket client_socket;

   cout << "START" << endl;

-  client_socket.connectSocket("192.168.1.105", 5000);

+  client_socket.connectSocket(argv[1], atoi(argv[2]));

   client_socket.setEvent(1, on_new_type);
   client_socket.setEvent(2, on_new_color);
```

32 server/server.c

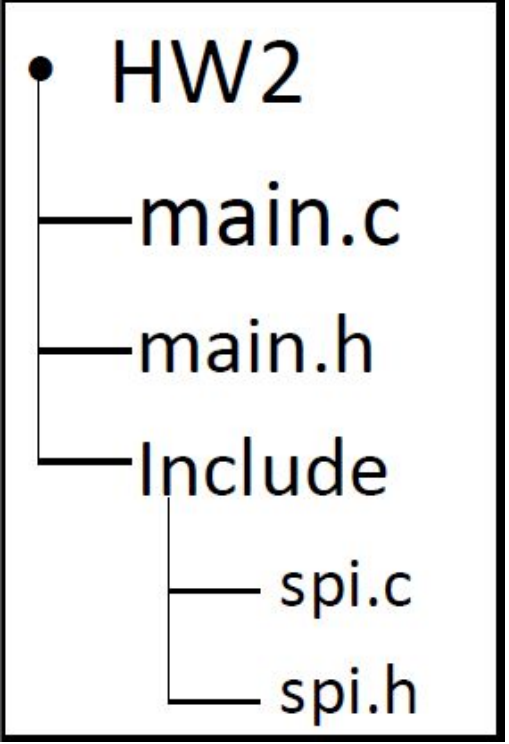
View

```diff
@@ -19,7 +19,7 @@
 #define MAX_CLIENTS 16
 #define DEFAULT_PORT 5000
 #define MAX_MESSAGE_BUFFER 1024
-#define MAX_MESSAGE_KEYS 16

+#define DEFAULT_MAX_KEYS 16
```


HW2
main.c
main.h
include
spi.c
spi.h

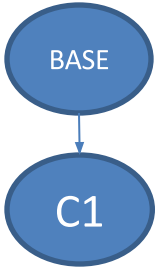
- HW2
  - main.c
  - main.h
  - Include
    - spi.c
    - spi.h

BASE

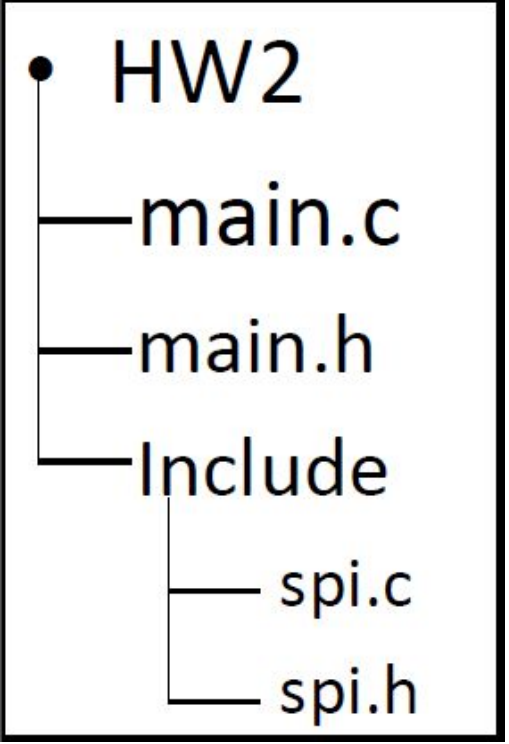


Change file

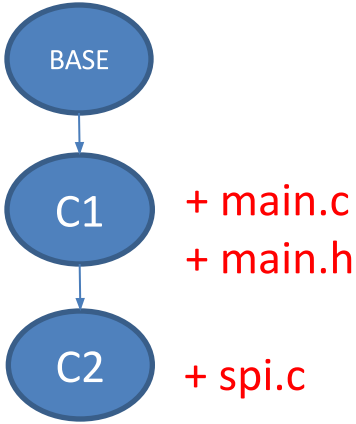
Change file

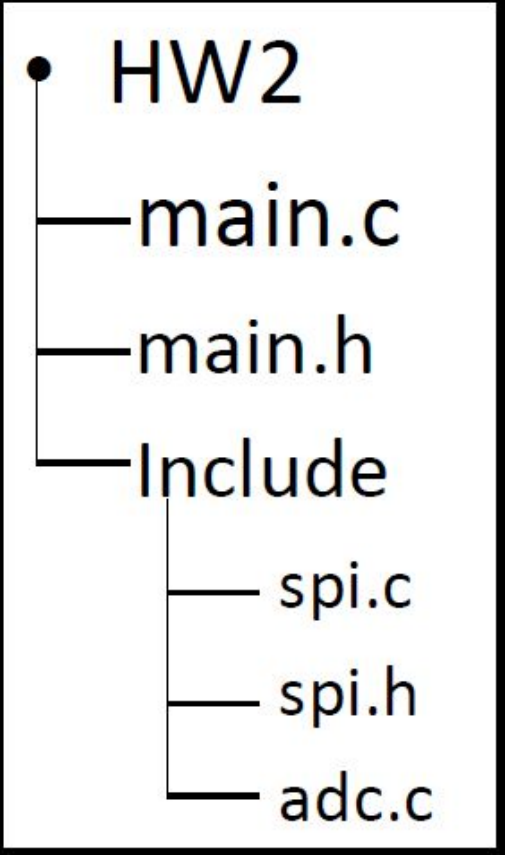


+ main.c
+ main.h

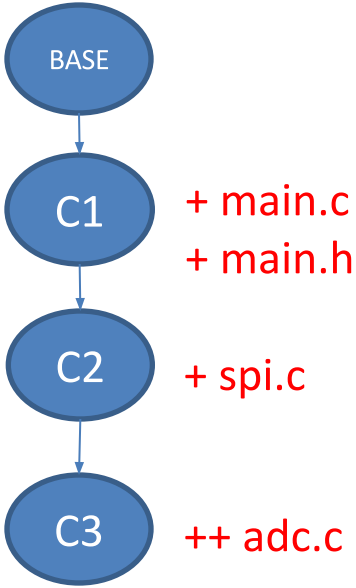

HW2
main.c
main.h
Include
spi.c
spi.h

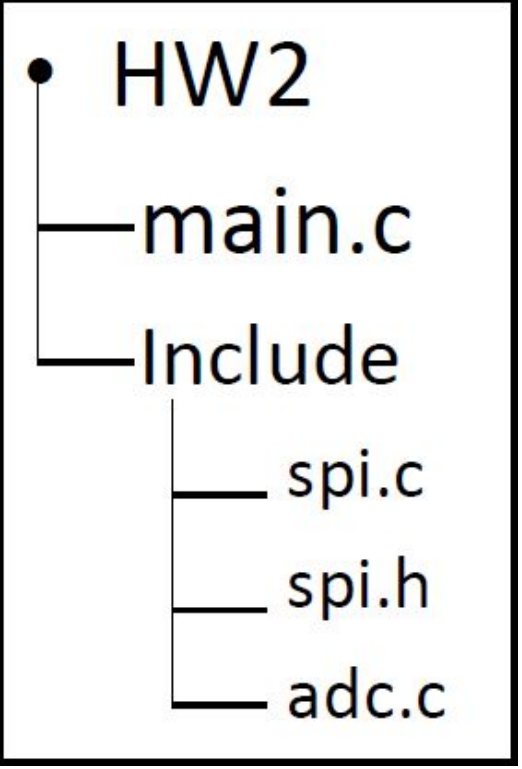
Change file


BASE
C1
+ main.c
+ main.h
C2
+ spi.c


HW2
main.c
main.h
Include
spi.c
spi.h
adc.c

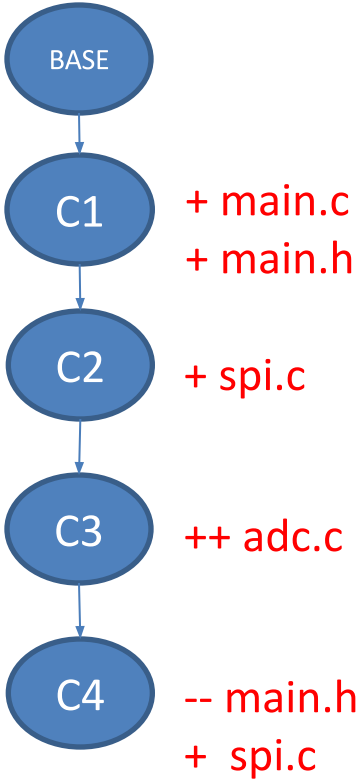
Added file


BASE
C1
+ main.c
+ main.h
C2
+ spi.c
C3
++ adc.c



Removed file

Change file



	BASE	C1	C2	C3	C4
main.c	v0	v1	v1	v1	v1
main.h	v0	v1	v1	v1	---
Include/spi.c	v0	v0	v1	v1	v2
Include/spi.h	v0	v0	v0	v0	v0
Include/adc.c	---	---	---	v0	v0

It Remembers
…so you don’t have too

- Git saves all past **Committed** saves in a .git file in the repository
- New people can go back to ANY old **Commit** made during life of repository
- It’s the history of the project - hence Version Control

You can even “blame” others

- Git’s blame feature allows you to see each line’s last change

fscrypt: factor out bio specific functions	3 months ago	46 struct workqueue_struct *fscrypt_read_workqueue;
fs crypto: move per-file encryption from..	2 years ago	47 static DEFINE_MUTEX(fscrypt_init_mutex);
		48
		49 static struct kmem_cache *fscrypt_ctx_cachep;
		50 struct kmem_cache *fscrypt_info_cachep;
		51
		52 /**
		53 * fscrypt_release_ctx() - Releases an encryption context
		54 * @ctx: The encryption context to release.
		55 *
		56 * If the encryption context was allocated from the pre-allocated pool, returns
		57 * it to that pool. Else, frees it.
		58 *
		59 * If there's a bounce page in the context, this frees that.
		60 */
		61 void fscrypt_release_ctx(struct fscrypt_ctx *ctx)
		62 {
		63 unsigned long flags;
		64
fscrypt: Rename FS_WRITE_PATH_FL to ...	4 months ago	65 if (ctx->flags & FS_CTX_HAS_BOUNCE_BUFFER_FL && ctx->w.bounce_page) {
fs crypto: move per-file encryption from..	2 years ago	66 mempool_free(ctx->w.bounce_page, fscrypt_bounce_page_pool);
		67 ctx->w.bounce_page = NULL;
		68 }

Git 101 - Branches

- Branches let you work on the code in your own crazy direction and **Merge** it back later
- Example: Make a “New-Feature” branch and when it is ready, **Merge** back to the Master Branch

Git 101 – Workflow

- **Fetch/Pull**
- Make your edits
- **Stage** your changes
- **Commit** your work
- **Push**

Git 101 – Workflow - Fetch

- If you have not cloned the repo

  git clone https://the.git.repo.git

- Get the latest updates before working

  git pull

- Don’t make folder then git clone
  - Cloning makes a new folder for you

Git 101 – Workflow - Edit

- Add, remove, edit all the files you want

Git 101 – Workflow - Stage

- Add the files you want to commit

git add –A

  – Will add all difference

git add main.c

  – Just adds main.c

git add myFolder/*

  – Adds entire folder

Git 101 – Workflow – Commit

- Take the “snapshot” of the repo
- Add a commit title

  git commit –m “Best commit EVA”

- Optionally add a comment

  git commit –m “Best commit EVA” –m “Here is more details”

Git 101 – Workflow - Push

- When ready, push changes to server

  git push

GitHub Permissions

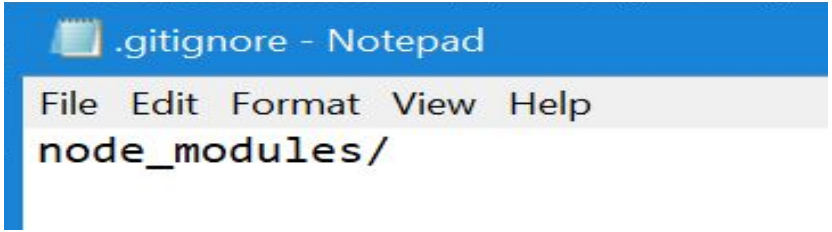
- For a project you can always just download the code and do what you want.
- If you want to make changes, either **Clone** or **Fork** the Repository
  - You can then send a **Pull Request** that will let someone in charge of Repo check your changes and **Merge** it
- If you set someone as a **Collaborator** they can **Push** code without having to submit a **Pull Request**

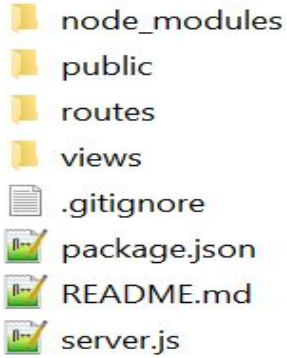
Almost Forgot about .gitignore

- Git is not the best with Binary files (.mp3, .pdf, .exe, .FileTypeICannotReadInNotepad)
- List all the folders and files that git will not recognize
- Almost all types of projects have a standard .gitignore template found on GitHub

.gitignore example

- We don’t want the node_modules folder commited
- Add it to .gitignore





Merge Conflicts

- Not that scary
- Happens when same lines are altered in two different commits
- Debunking the myth that merge conflicts are hard

Commit from User 1

```c
#include <stdio.h>
#include <stdlib.h>

#define VALUE 40

int main(int argc, char* arcv[]) {

    int a = 5;

    if ( a > 6 ) {
        printf("a is big");
    }

    return 0;

}
```

Commit from User 2

```c
#include <stdio.h>
#include <stdlib.h>

#define VALUE 40

int main(int argc, char* arcv[]) {

    int a = 5;

    if ( a > 4 ) {
        printf("a is small");
    }

    return 0;

}
```

Merge Conflict will need to be resolved before able to merge

Just open the file in text editor

The Merge Conflict File

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   #define VALUE 40
5
6   int main(int argc, char* arcv[]) {
7
8       int a = 5;
9
10  <<<<<<< commit/user1
11      if ( a > 4 ) {
12          printf("a is small");
13  |||||||
14      if ( a > 6 ) {
15          printf("a is big");
16  >>>>>>> Commit/user2
17      }
18
19      return 0;
20
21  }
```

Change file to look the way you want and save

…simple, I know

GitHub GUI – Make new repo

- Can either create online or bring it to local computer.
- Start new repository and push to GitHub when ready.
- Can take a current set of code and make it into a Git repo.
  - Will have no history prior to initialization of git

Recap

- Get Repository (**Clone**) or Sync it **(Fetch)**
- Know which **Branch** you are in
- Make your changes
- Set which changes you are **Staging**
- **Commit** the changes
- When ready, Sync or send a Pull Request

Time to get some practice!

tinyurl.com/GitWorkshopIEEE

[CASE SENSITIVE]