




CS 211 RECITATIONS WEEK 4

Wenjie Qiu
Teaching Assistant
Office Hours: Thu. noon – 1pm
wenjie.qiu@rutgers.edu



Content

- GDB
- C project and Makefile
- Version control with git
- typedef
- C preprocessor

GDB

Don't forget to compile with `-g` for debug information!

- `gdb myProgram` (in shell)
- `run arg1 arg2 ...` (in gdb)

<code>break</code>	set a breakpoint
<code>run</code>	run program
<code>list</code>	show original source code
<code>step</code>	step to next line (into a function)
<code>next</code>	step to next line (over function calls)
<code>continue</code>	continue running after stopping
<code>kill</code>	kill program being debugged
<code>quit</code>	exit gdb and kill program
<code>print</code>	evaluate source expression
<code>x</code>	display memory contents
<code>bt</code>	show call stack
<code>frame</code>	select stack frame

Helpful resources: <http://csapp.cs.cmu.edu/3e/docs/gdbnotes-x86-64.pdf>

C project and Makefile

Object files (.o)

- A .c file can also be **compiled** into an *object (.o) file* with **-c** :

```
$ gcc -c part1.c
$ ls
part1.c  part1.o  part2.c
```

→ produces

- a .o file is a binary “blob” of compiled C code that cannot be directly executed, but can be directly **linked** into a larger *executable* later
- You can **compile** and **link** a mixture of .c and .o files:

```
$ gcc -o myProgram part1.o part2.c
```

→ produces

Avoids recompilation of unchanged partial program files (e.g. **part1.o**)

Header files (.h)

- **header** : A C file whose only purpose is to be #included (#include is like java import statement)
 - generally a filename with the .h extension
 - holds shared variables, types, and function declarations
 - similar to a java interface: **contains function declarations but not implementations**
- key ideas:
 - every **name.c** intended to be a module (not a stand alone program) has a **name.h**
 - **name.h** declares all global functions/data of the module
 - other .c files that want to use the module will #include **name.h**

C project and Makefile

- Compiling *multi-file* programs repeatedly is cumbersome:

```
$ gcc -o myprogram file1.c file2.c file3.c
```

- What is make?

- *A utility for automatically compiling ("building") executables and libraries from source code.*
- *A very basic compilation manager*

- What is a makefile?

- *A script file that defines rules for what must be compiled and how to compile it*

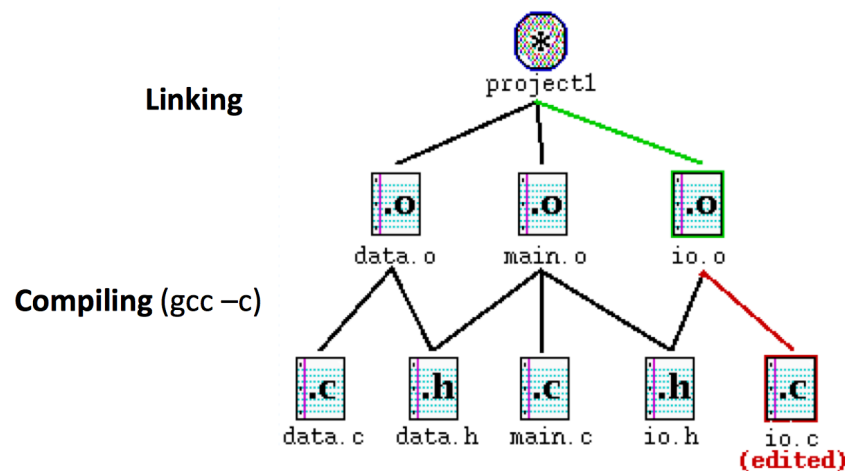
- Note that we can specify variables in makefile externally when running make

- *CFLAGS=-g -Wall -fsanitize=address -std=c89*
- *make CFLAGS="-g -Wall -fsanitize=address -std=c99" (no spaces before and after =)*

C project and Makefile

Dependencies

- **dependency** : When a file relies on the contents of another.
 - can be displayed as a *dependency graph*
 - to build `main.o`, we need `data.h`, `main.c`, and `io.h`
 - if any of those files is updated, we must rebuild `main.o`
 - if `main.o` is updated, we must update `project1`



Version control with git

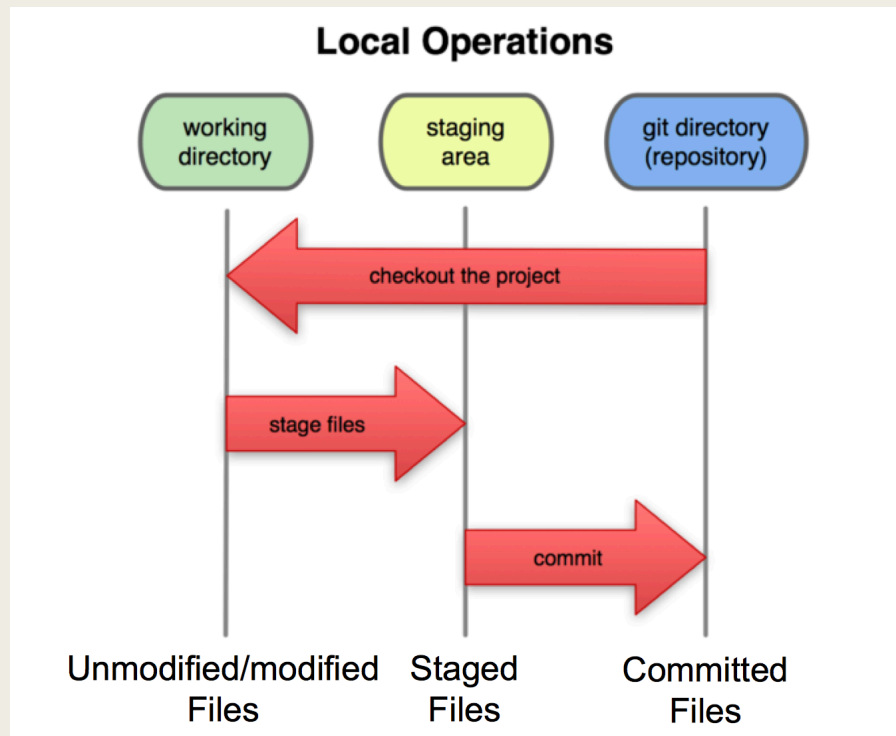
- git is not GitHub

- *git is a version control tool, can be used locally or upload your repositories on the cloud*
- *GitHub is the cloud 😊*

- Basic commands:

- *git init*
- *git add .* *(or git add *)*
- *git commit -m "your comments here"*
- *git log* *(show the commit history)*
- *git branch* *(create new branch)*
- *git checkout <branch/tags/commits>*
(HEAD pointer move to different place)

VERSION CONTROL WITH GIT



Version control with git

■ Configure

- `git config (- - global) user.name [your_name]`
- `git config (- - global) user.email [your_email]`

■ Using git on the cloud

- `git clone` *(clone the entire repo to local)*
- `git pull` *(keep your local repo updated)*
- `git push` *(push the local changes to your remote repo)*

■ Differences

- `git diff` *(find the differences)*

■ Tagging

- `git tag -a v1.0 -m "version 1.0 is done."`
- `git tag -d v1.0`

Version control with git

■ .gitignore

- *We only want to include code and documents, not binary, other intermediate trash files.*
- *Once we declare what we don't need, it won't be added to this repo*
- *What file should we dismiss? (<https://github.com/github/gitignore>)*
- *Other unnecessary huge files, e.g., pdf, pptx, docx, etc.*
- *Your repo grows! Every changes you made will be recorded, every (even deleted) files can be accessed.*

■ git will not commit empty directories

- *mkdir empty_dir*
- *git add . (nothing happens, and git commit will not work)*
- *what if we really want to keep the directory?*
- *cd empty_dir*
- *touch keep.txt (or any other names, e.g., gitkeep.txt, etc)*
- *So that we made the difference!*

typedef

- An example of typedef struct

```
typedef struct my_struct {  
    int a;  
    char b;  
} my_struct;
```

- typedef is associated with the semantics
 - *size, never negative, must huge enough*

```
typedef unsigned long long size_t // for 64-bits machines
```

```
typedef unsigned long size_t // for 32-bits machines
```

- Underscore t (“_t”) is usually a size type!
 - *time_t*
 - *clock_t*
 - *many important data structures in Linux kernels has their own types*

C preprocessor

- `#include`
- `#define SIZE 10`
- `#undef SIZE`

- `#define DEBUG`
- `#ifdef DEBUG`
 - `/* your code here */`*
 - `/* e.g., print something */`*
- `#endif` *// always don't forget to end your if*