Creative Software Programming

Lab 1: Supplementary

ITE1015

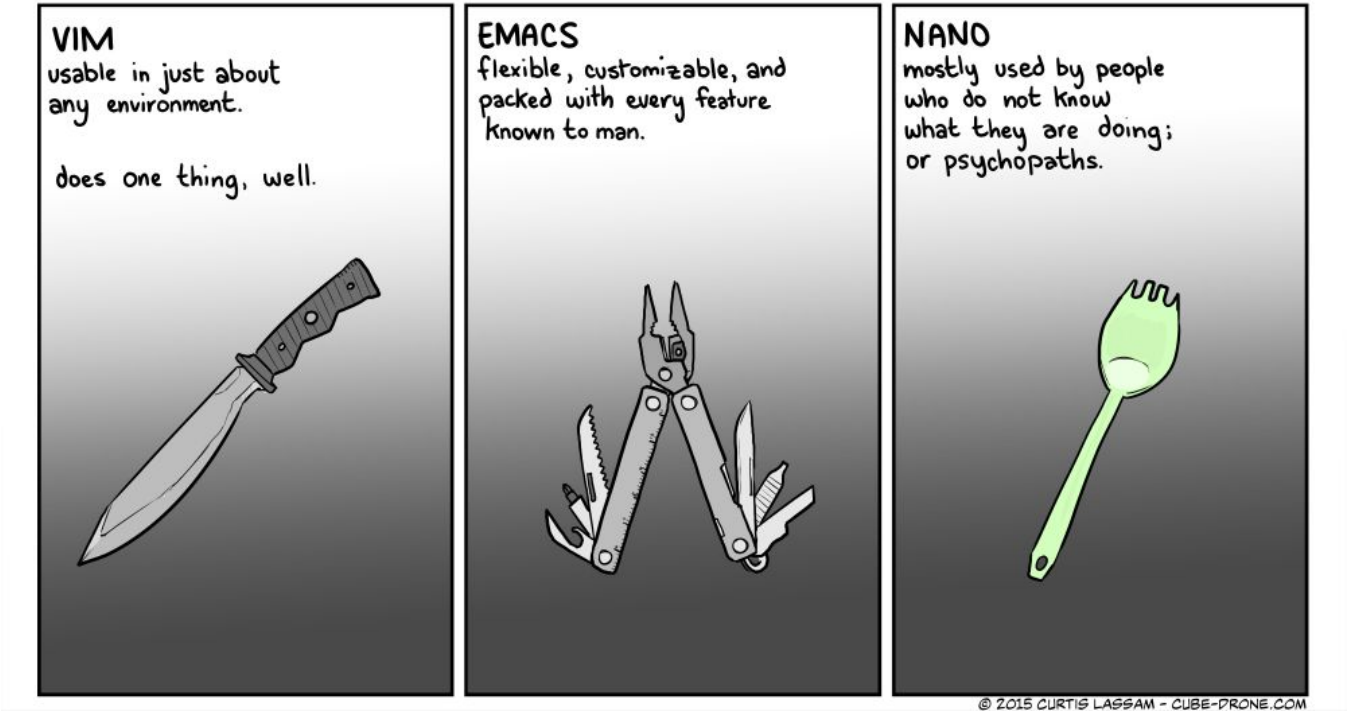
2020 second semester

Jiun Bae (maytryark@gmail.com)

Topic

- Advanced usage of VIM
- Concept of Git and basic usage
- C/C++ build and debugging

Vim


VIM
usable in just about any environment.
does one thing, well.
EMACS
flexible, customizable, and packed with every feature known to man.
NANO
mostly used by people who do not know what they are doing; or psychopaths.
© 2015 CURTIS LASSAM - CUBE-DRONE.COM

Vim usage

```
$ sudo apt install vim-gtk

# create test.txt or open if exists
$ vim test.txt

# .vimrc file is settings of Vim
$ vim ~/.vimrc
```

Vim config (.vimrc)

~/.vimrc

```
"type below to ~/.vimrc

syntax on                    "use syntax highlighting
filetype plugin indent on    "use auto-indentation

set expandtab                "use spaces instead of a tab
set tabstop=4                "number of spaces for a tab
set shiftwidth=4             "number of spaces for each step of indent

set nowrap                   "stop line breaking
set clipboard=unnamedplus    "use system clipboard (e.g. when using 'yy')
set ignorecase               "case-insensitive search
set incsearch                "use incremental searc
```

after editing, press ‘ESC - :w - Enter’ to save, ‘ESC - :q - Enter’ to quit.
Or ‘ESC - :x - Enter’ to save and quit.

Vim Visual Mode

Cheatsheet
- https://vim.rtorr.com/
- https://vim.rtorr.com/lang/ko

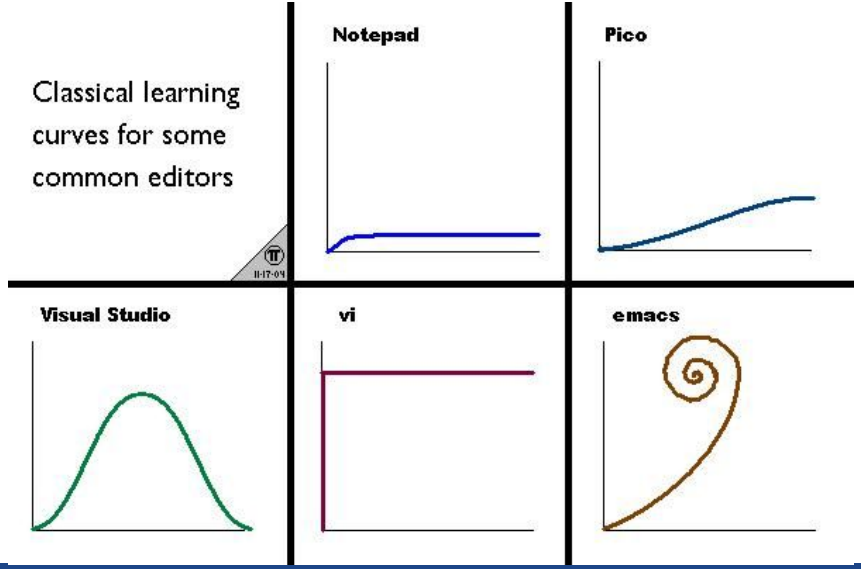
Marking text (visual mode)

- v - start visual mode, mark lines, then do a command (like y-yank)
- V - start linewise visual mode
- Ctrl + v - start visual block mode

In visual mode, mark a block and copy(y), paste(p) and so on.

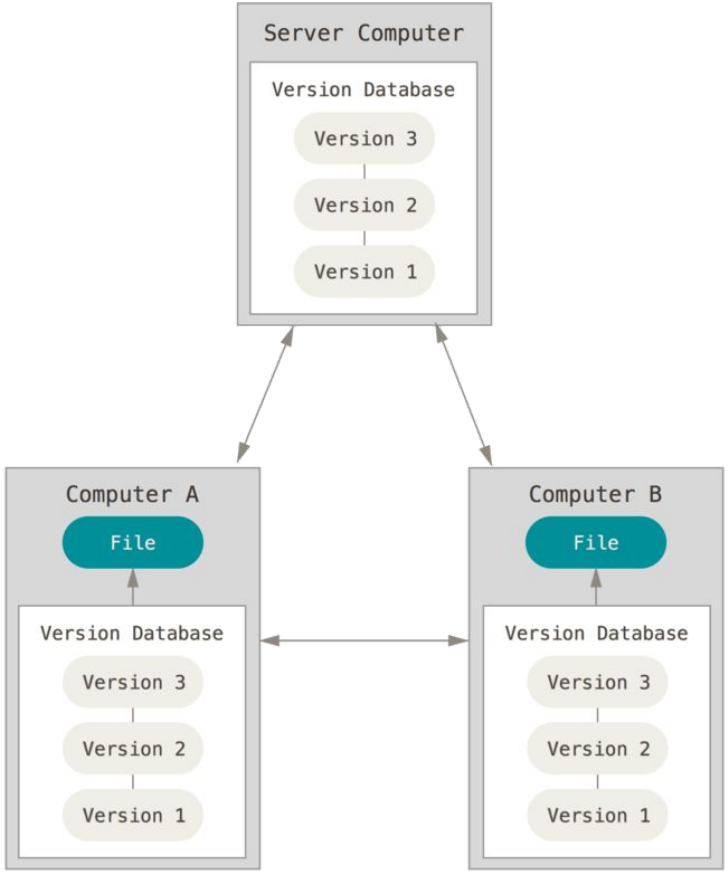
Vim appendix

- Game to learn vim (what????)
  - VIM Adventures: Learn VIM while playing a game
- Or want to use emacs?
-

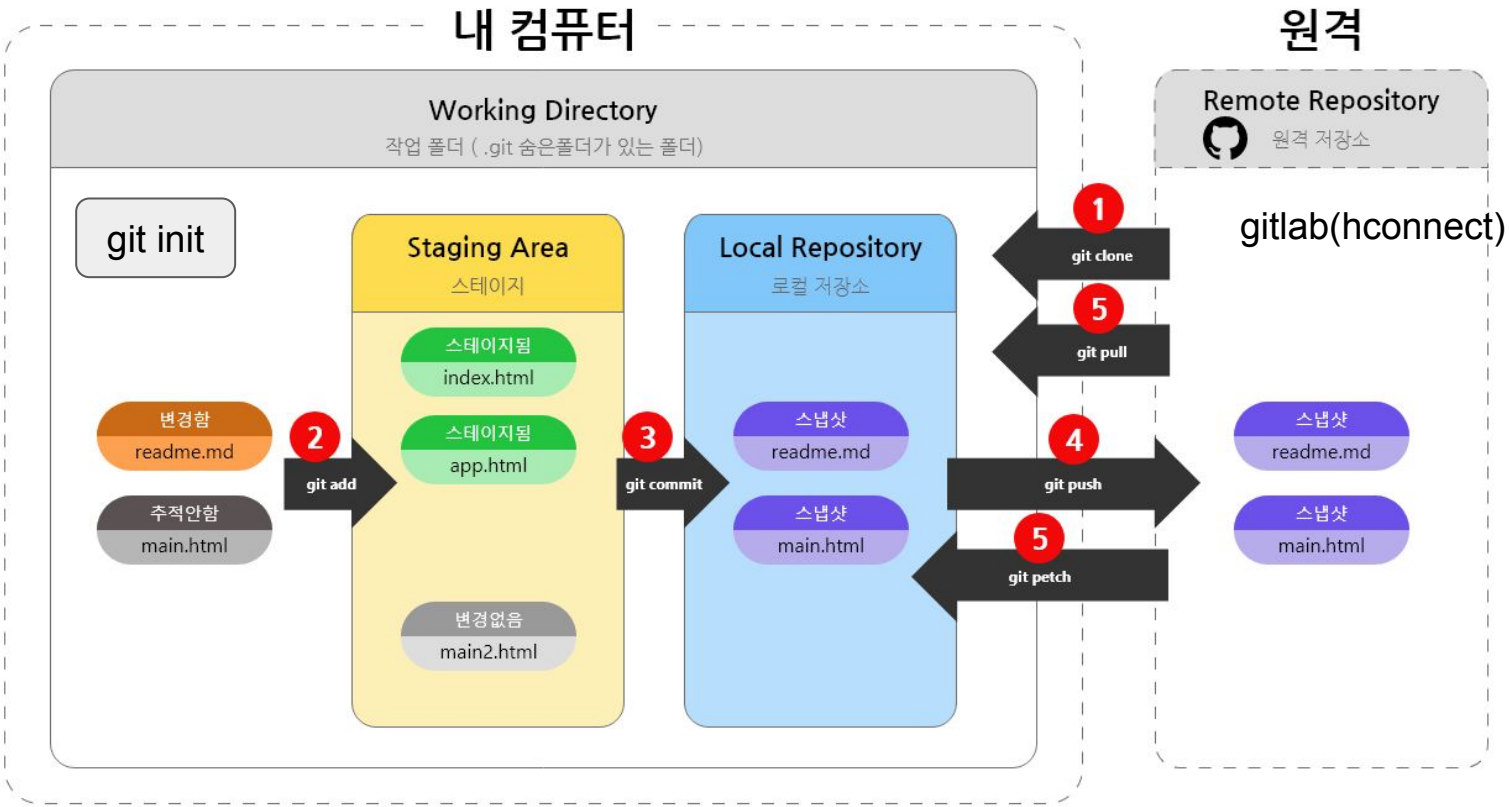


Git

Git is a distributed version-control system for tracking changes in source code during software development.



Git

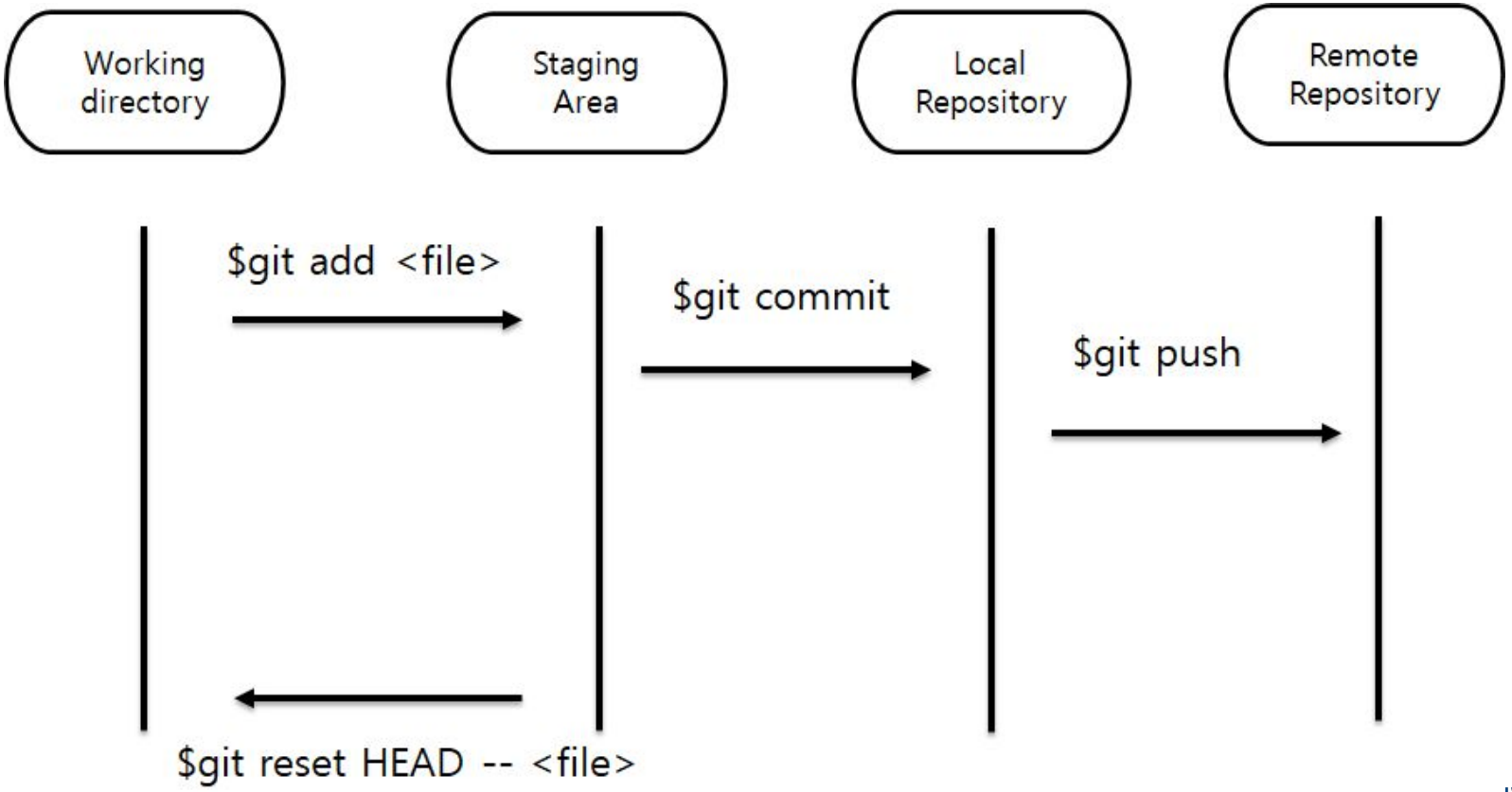


파일관리
(추적)의상태

Untracked	Tracked		
추적안함 Untracked	변경없음 Unmodified	변경함 Modified	스테이지됨 Staged

https://uxgjs.tistory.com/182

Git workflow



(Optional) Git documentation

- https://git-scm.com/book/en/v2
- https://git-scm.com/book/ko/v2

Basic Git usage

Git config

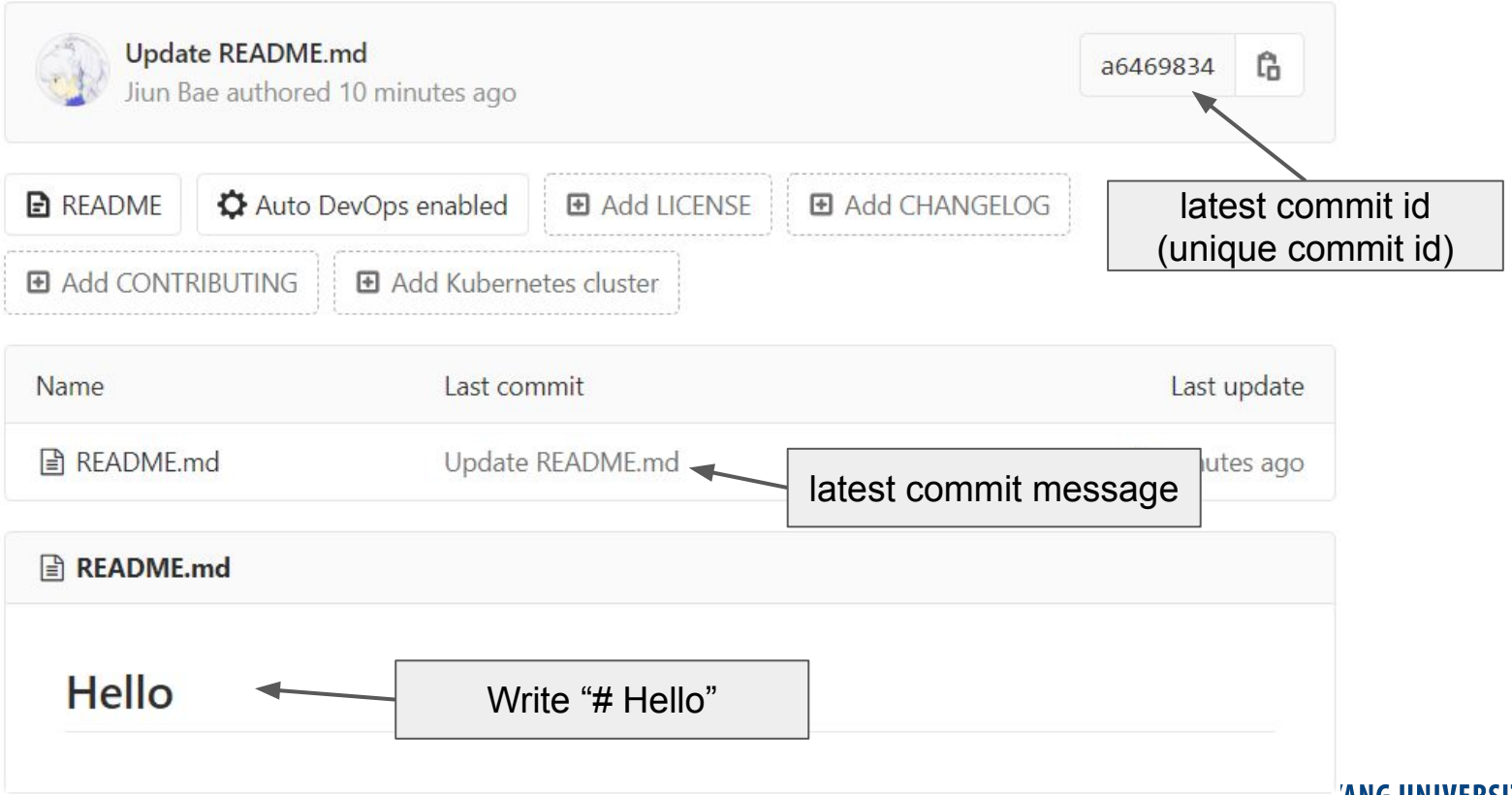
```
$ git config --global user.name "{YourName}"
$ git config --global user.email "{StudentID}@hanyang.ac.kr"
```

Git basic workflow

```
# clone repository from remote
$ git clone https://hconnect.hanyang.ac.kr/2020_ite1015_12522/{StduentID}.git

# open cloned directory
$ cd {StudentID}

# do something
# (create README.md file and write "# Hello")

# add test.txt to staging (not commited)
$ git add README.md

# commit with message
$ git commit -m "Update README.md"

# upload to remote server
$ git push -u origin master
```

GitLab

https://hconnect.hanyang.ac.kr/2020_ite1015_12522/{StudentID}



gcc/g++

open-source C++ compiler

Most formats and options are the same as default c-compiler (gcc, cc)

g++ [options] <infile> ...

- -c : compile and assemble, but do not link Create only object file (.o) without creating executable
- -g : debug info. Contains information necessary for debugging (source code, etc.)
- -o <outfile> : Place the output into <outfile>
- -I<dir> : include directory. (directory name to look for headers when compiling)
- -L<dir> : library directory. (Directory name to look for library files when linking)
- -D<symbol>[=def] : define a macro to use at compile time
- ... : There are numerous other options.

Example: Compile

Write main.cc file like below

main.cc

```cpp
#include <iostream>

int main() {
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

And compile it!
g++ generate executable using source code (main.cc)

```
# -o is output file name
$ g++ main.cc -o main

# run!
$ ./main
Hello World!
```

Example: Compile

Write main.cc, print.cc, print.h file like below

main.cc

```
#include "print.h"

int main() {
  print_hello();
  return 0;
}
```

import print.h

use function defined in print.h

b.cc

```
#include <iostream>
#include "print.h"

void print_hello() {
  std::cout << "Hello World!" << std::endl;
}
```

import for print

print.h

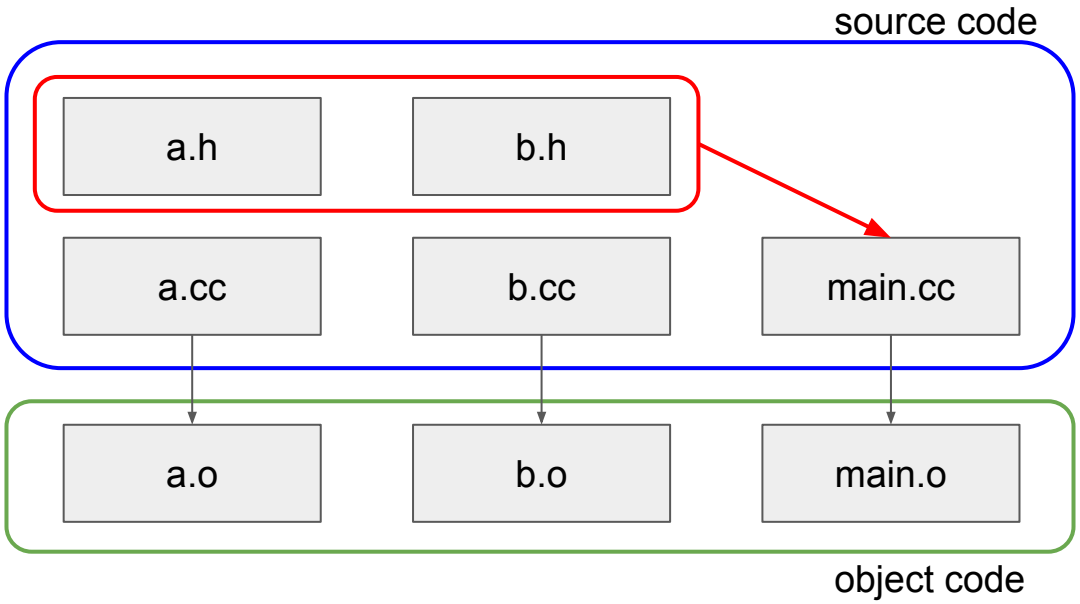
```
void print_hello();
```

Example: Compile

And compile it!
g++ generate executable using source code (main.cc, print.cc)

```
$ g++ main.cc print.cc -o main

$ ./main
Hello World!
```

C program build workflow



Make

Build tools that have been around for a long time on Unix.

- Rules for how to compile and link the source to create an executable
- When ‘make’ is run, find ‘Makefile’ (or ‘makefile’) in that directory and runs it

Makefile

```
<target> : <dependency>
    <recipe>
    <recipe>
    ...
```

Example: Make

Write code

main.cc

```cpp
#include "a.h"
#include "b.h"

int main() {
  function_a();
  function_b();
  return 0;
}
```

a.h

```cpp
void function_a();
```

b.h

```cpp
void function_b();
```

a.cc

```cpp
#include <iostream>
#include "a.h"

void function_a() {
  std::cout << "function a called" << std::endl;
}
```

b.cc

```cpp
#include <iostream>
#include "b.h"

void function_b() {
  std::cout << "function b called!" << std::endl;
}
```

Example: Make

Makefile

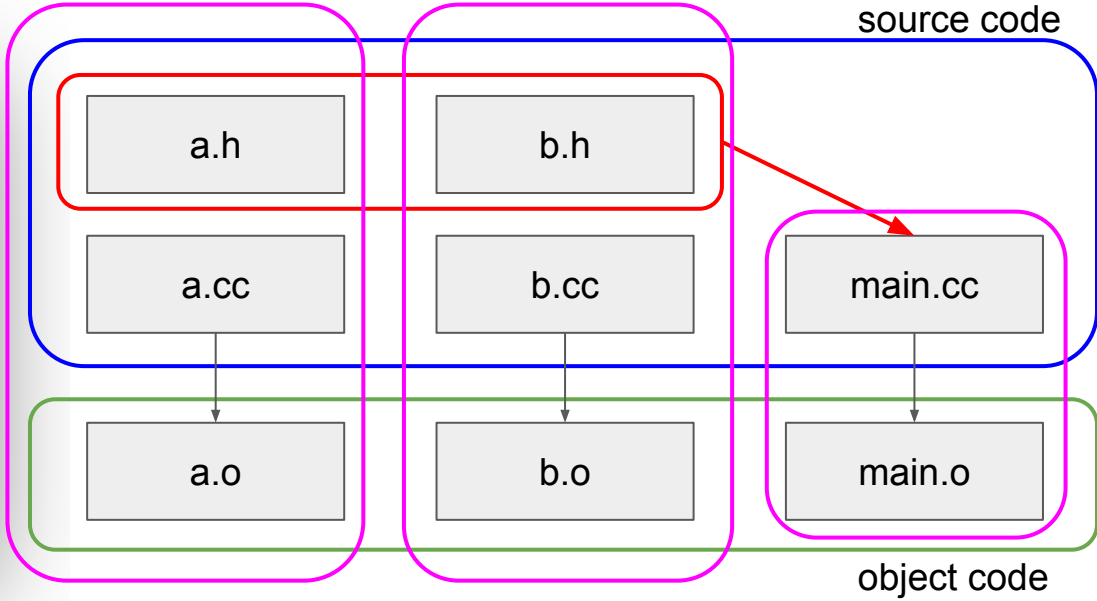
```
main : a.o b.o main.o
        g++ -o main a.o b.o main.o

a.o : a.cc
        g++ -c -o a.o a.cc

b.o : b.cc
        g++ -c -o b.o b.cc

main.o : main.cc
        g++ -c -o main.o main.cc

clean :
        rm *.o main
```

```
# make will create main output file
# main requires a.o, b.o, main.o
# It performs the each compilation.
$ make

$ ./main
function a called
function b called!
```

Example: Make

Makefile

```
main : a.o b.o main.o
        g++ -o main a.o b.o main.o

a.o : a.cc
        g++ -c -o a.o a.cc

b.o : b.cc
        g++ -c -o b.o b.cc

main.o : main.cc
        g++ -c -o main.o main.cc

clean :
        rm *.o main
```



TODO:

Change only b.cc

then make and run

make will compile only for changes.

b.cc

```
#include <iostream>
#include "b.h"

void function_b() {
  std::cout << "function b called!" << std::endl;
}
```

b.h

```
#include <iostream>
#include "b.h"

void function_b() {
  std::cout << "function b is changed!" << std::endl;
}
```

Assignment #1 (due: - today, but no scoring)

1. README.md must added in your repository.
2. create directory named ‘week-1’ in your repository.
3. And write code of “Example: Make” (main.cc, a.cc, a.h, b.cc, b.h, Makefile)
4. after all, upload your works to remote repository using ‘git push’

You should be able to check your results on the hconnect webpage.

The week-1 directory should contain the following files:

- {StudentID}.txt, main.cc, a.cc, a.h, b.cc, b.h, Makefile

Appendix

For students who have difficulty using Linux, I will be uploading a simple Linux usage today.

If you are not logged into GitLab, please log in today.

I'll invite you to the group and create a repository.

If you do not have personal storage, please tell me after logging in.

QA: https://open.kakao.com/o/glEhaRtc, mailto: maytryark@gmail.com

I prefer QA chat room because I want questions and answers to be visible fairly to everyone.