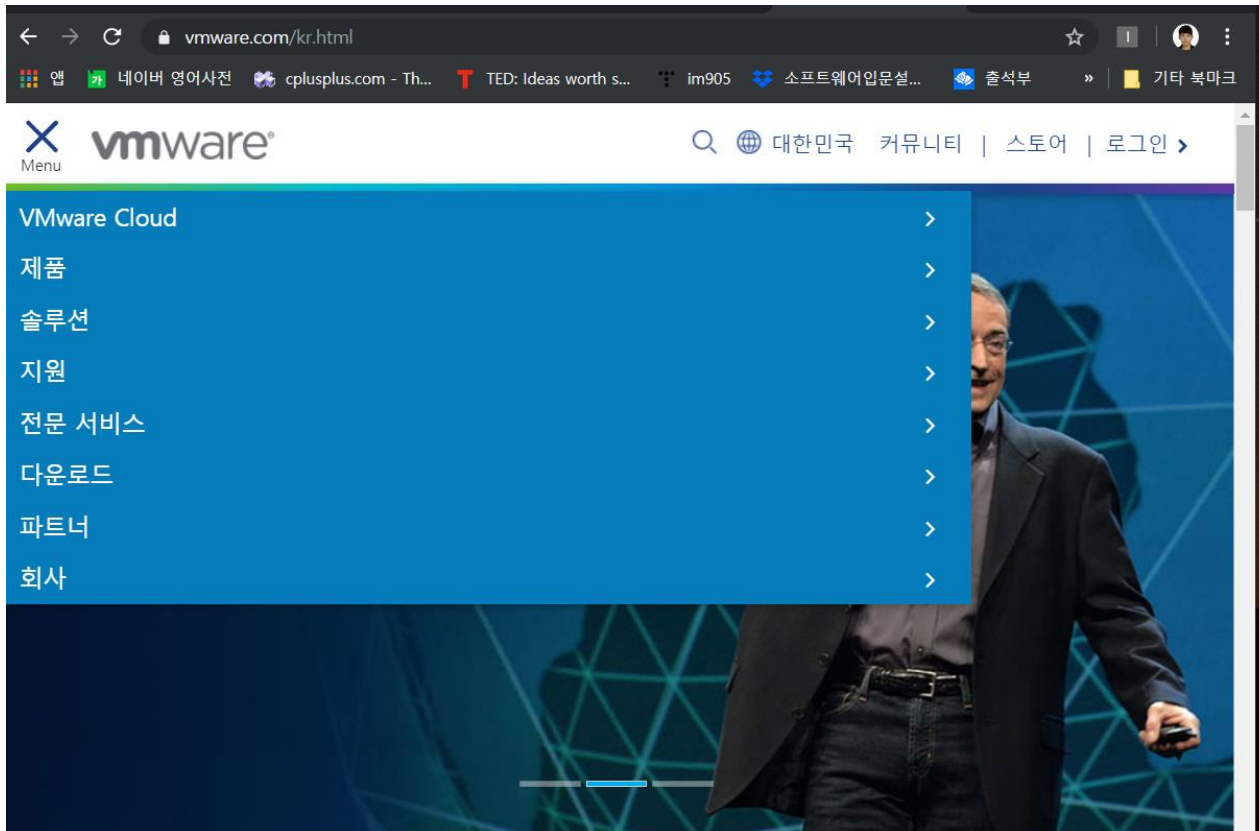


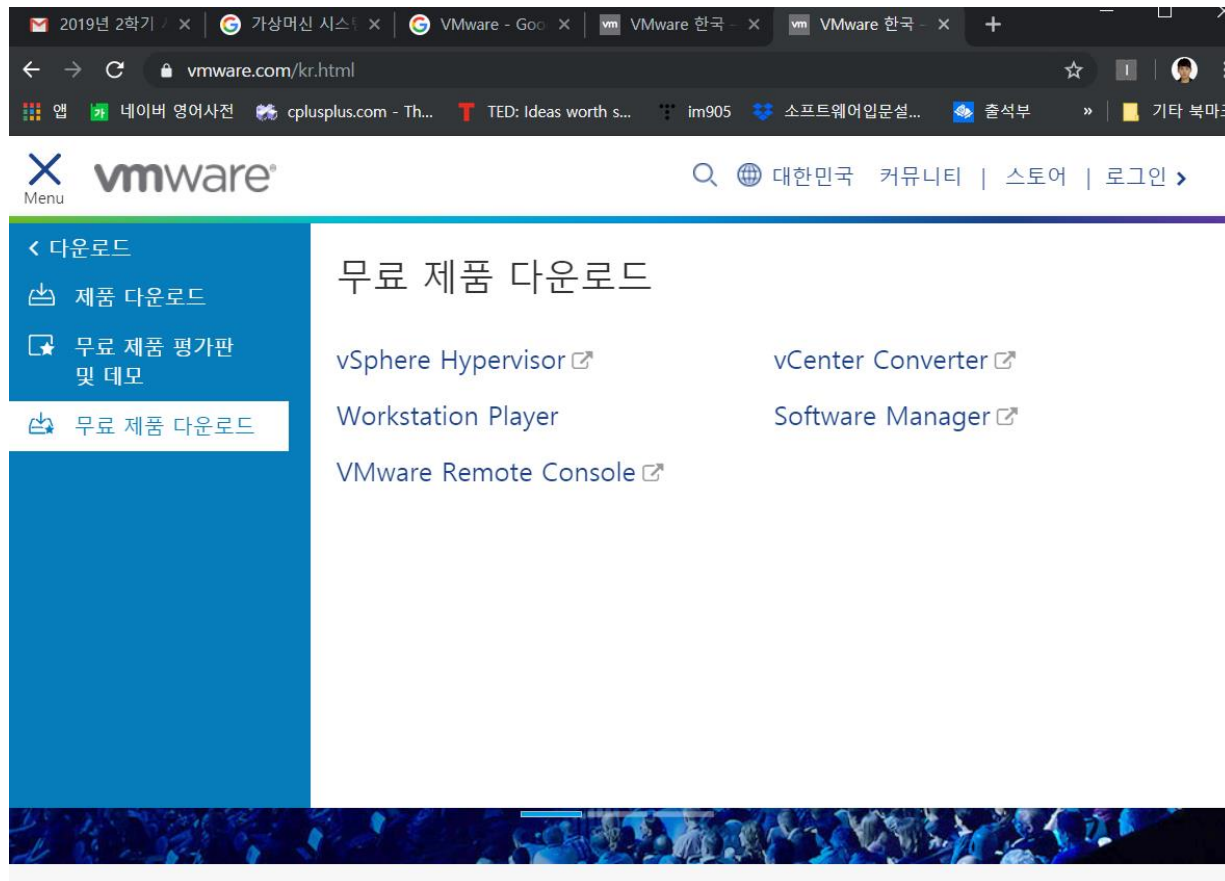
# VMware

- If you can't set the Virtual Box, try using VMware

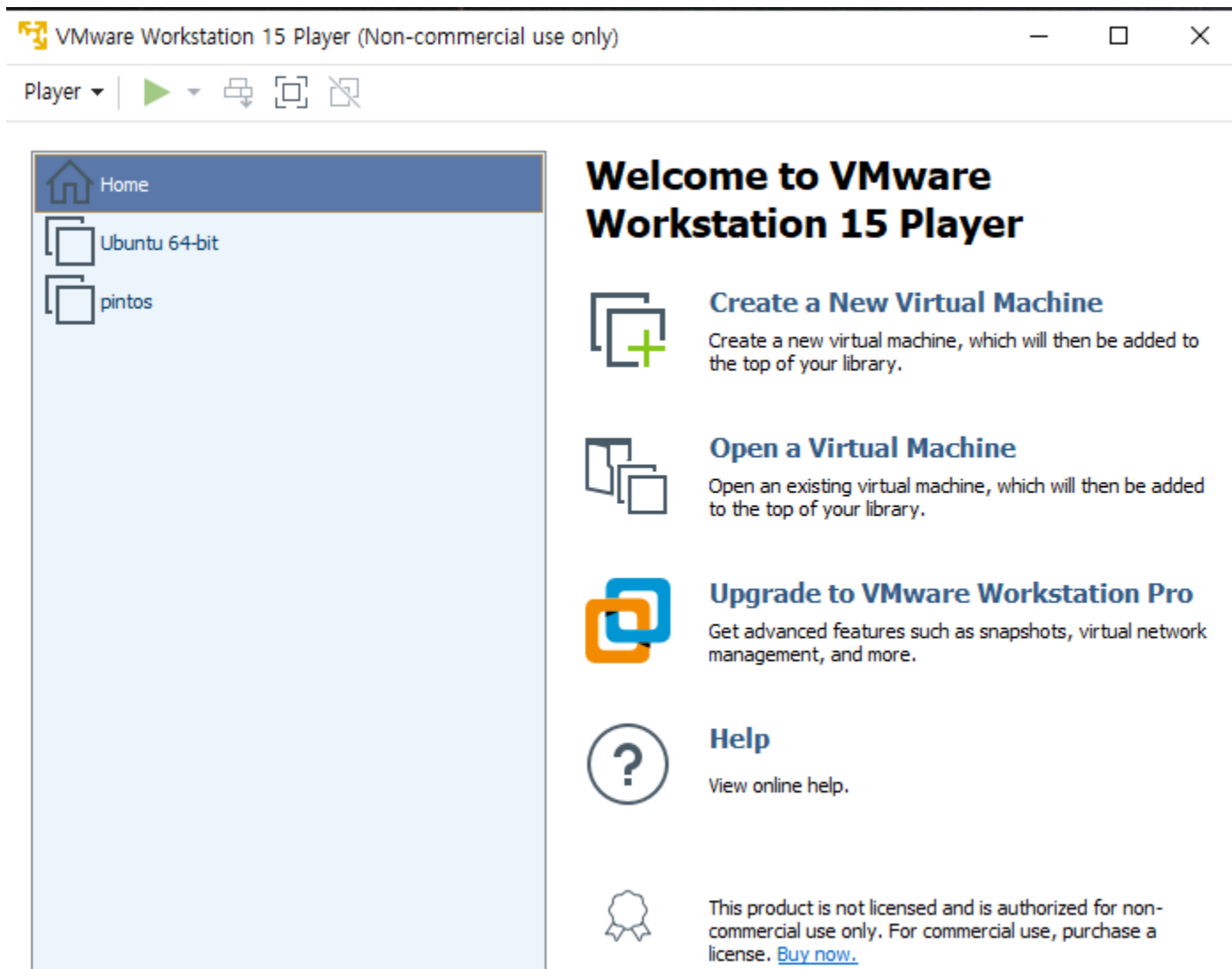


# VMware

- VMware workstation player is free software.



# VMware



VMware Workstation 15 Player (Non-commercial use only)

Player | [Play] [Full Screen] [Close]

**Home**

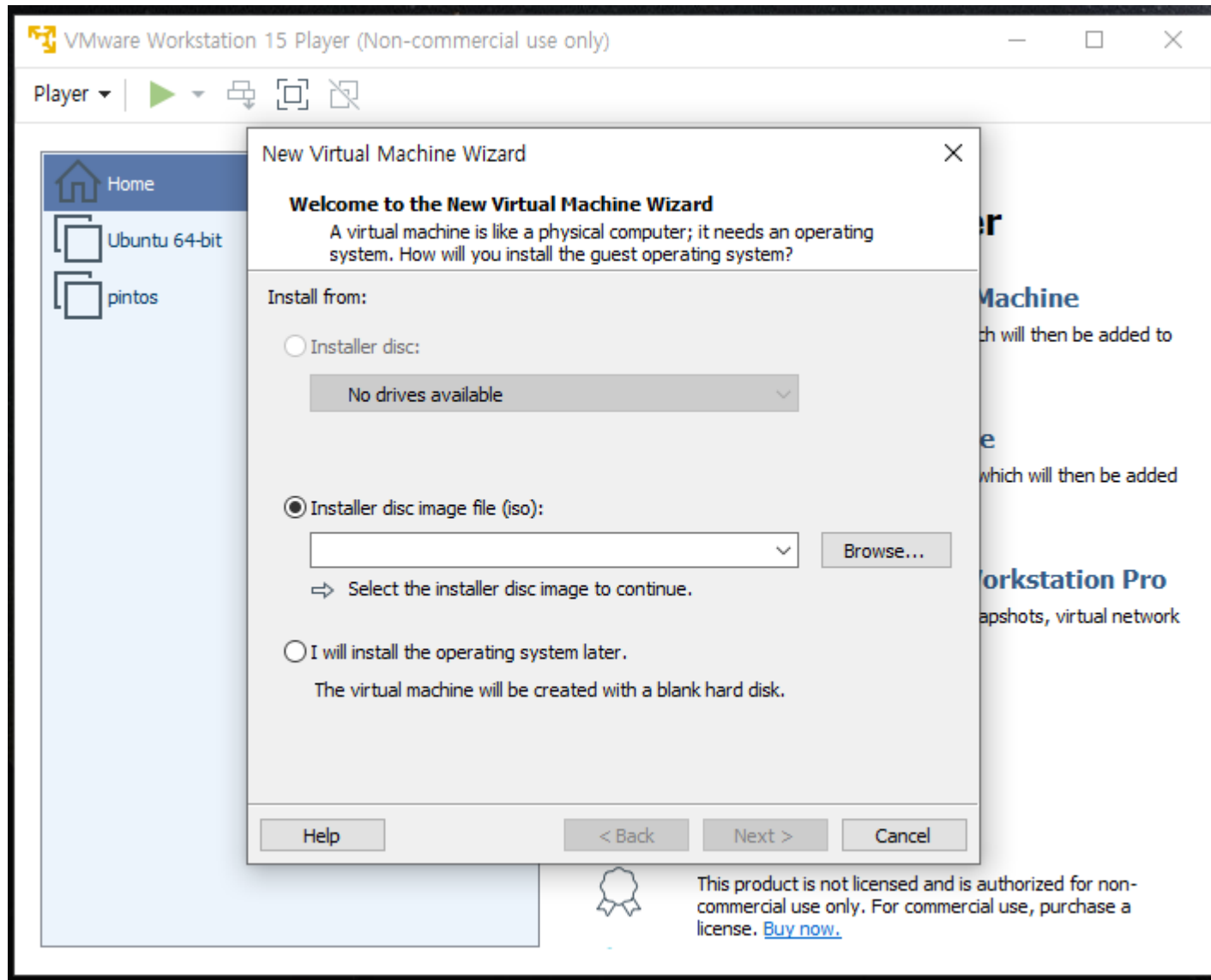
- Ubuntu 64-bit
- pintos

## Welcome to VMware Workstation 15 Player

- Create a New Virtual Machine**  
Create a new virtual machine, which will then be added to the top of your library.
- Open a Virtual Machine**  
Open an existing virtual machine, which will then be added to the top of your library.
- Upgrade to VMware Workstation Pro**  
Get advanced features such as snapshots, virtual network management, and more.
- Help**  
View online help.

This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. [Buy now.](#)

# VMware



# VMware

VMware Workstation 15 Player (Non-commercial use only)

Browse for ISO Image

바탕 화면 검색

바탕 화면

구성 새 폴더

이름	수정한 날짜	유형	크기
Project	2019-09-02 오후 11:23	파일 폴더	
Xilinx_Vivado_SDK_Win_2014.4_1119_1	2019-08-29 오후 12:27	파일 폴더	
서류	2019-08-27 오후 3:52	파일 폴더	
임시	2019-08-15 오후 3:24	파일 폴더	
ubuntu-18.04.2-desktop-amd64.iso	2019-04-10 오전 11:11	ALZip ISO File	1,949,696...

바로 가기

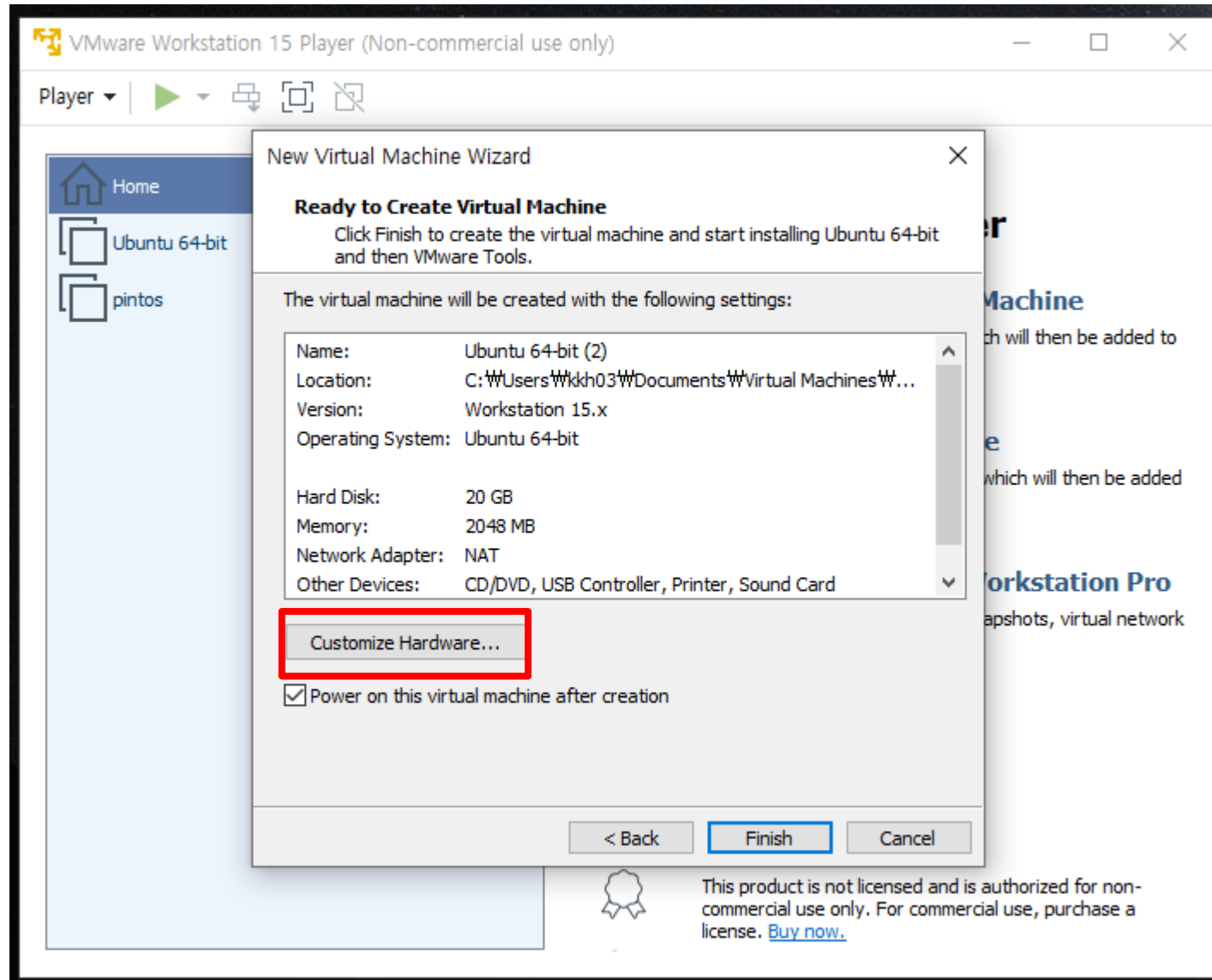
- 바탕 화면
- 다운로드
- 2019WDSC
- Assignment
- Lecture
- software Introdu
- OneDrive
- 내 PC
- USB 드라이브 (H:)
- .fseventsd
- .Spotlight-V100
- Lab1

파일 이름(N): ubuntu-18.04.2-desktop-amd64.iso

CD-ROM images (\*.iso)

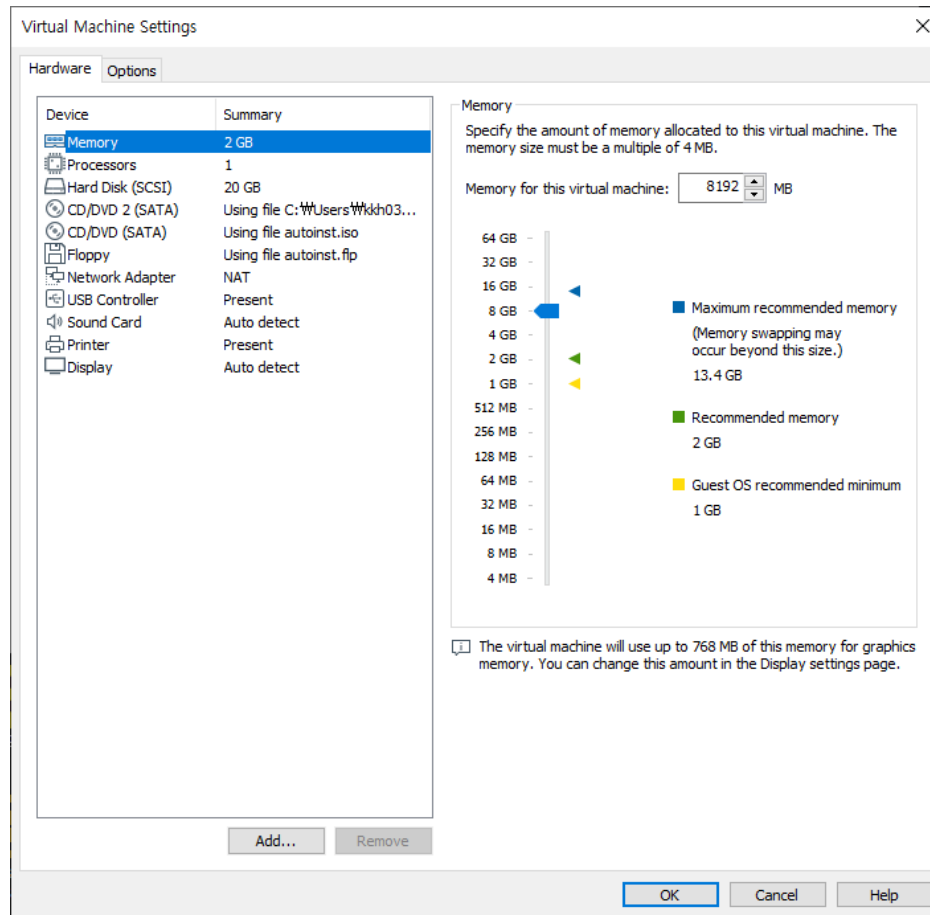
열기(O) 취소

# Open Terminal



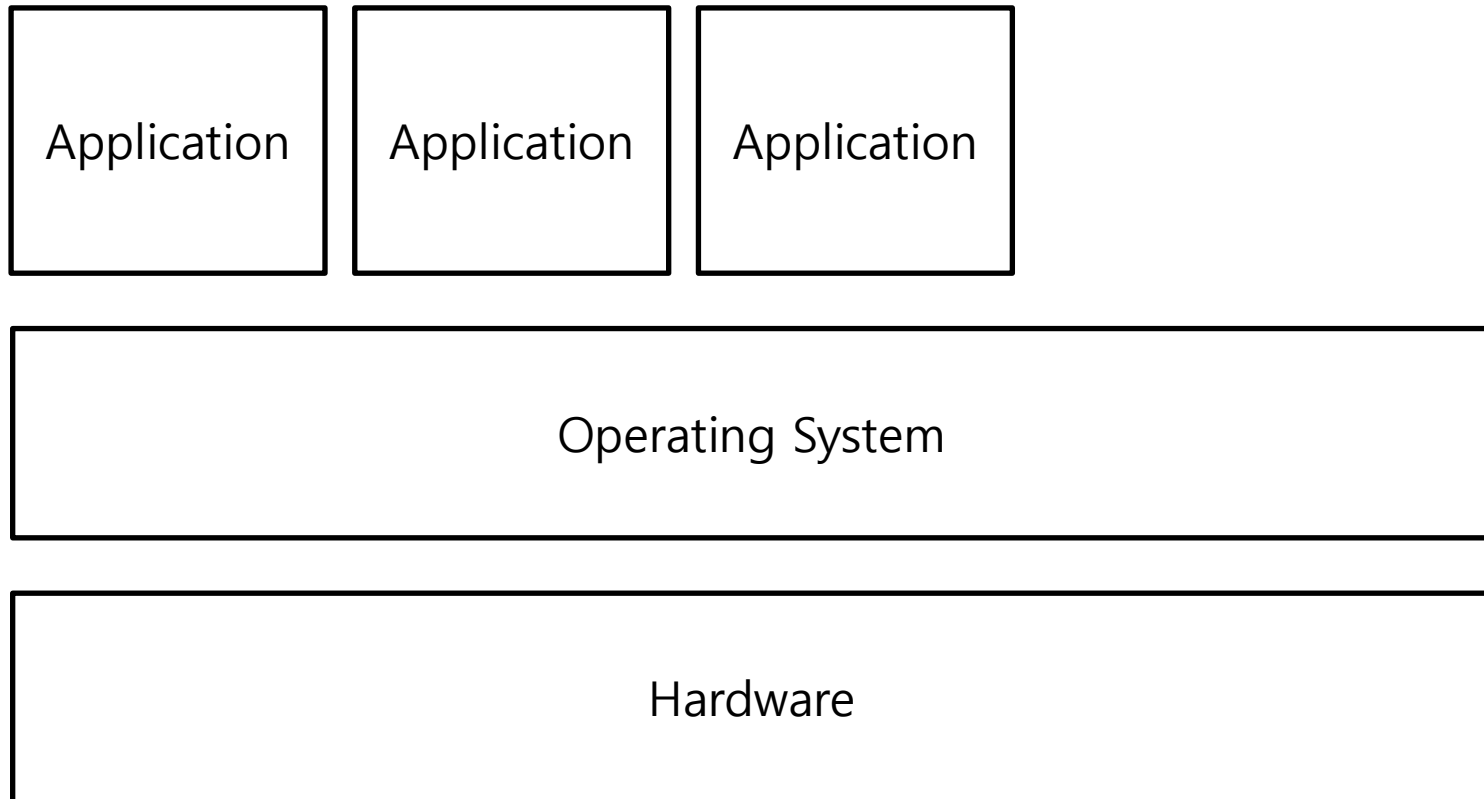
# VMware

- Set memory up to half of your laptop's capacity



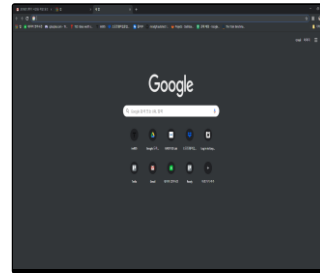
# What are we doing.

---





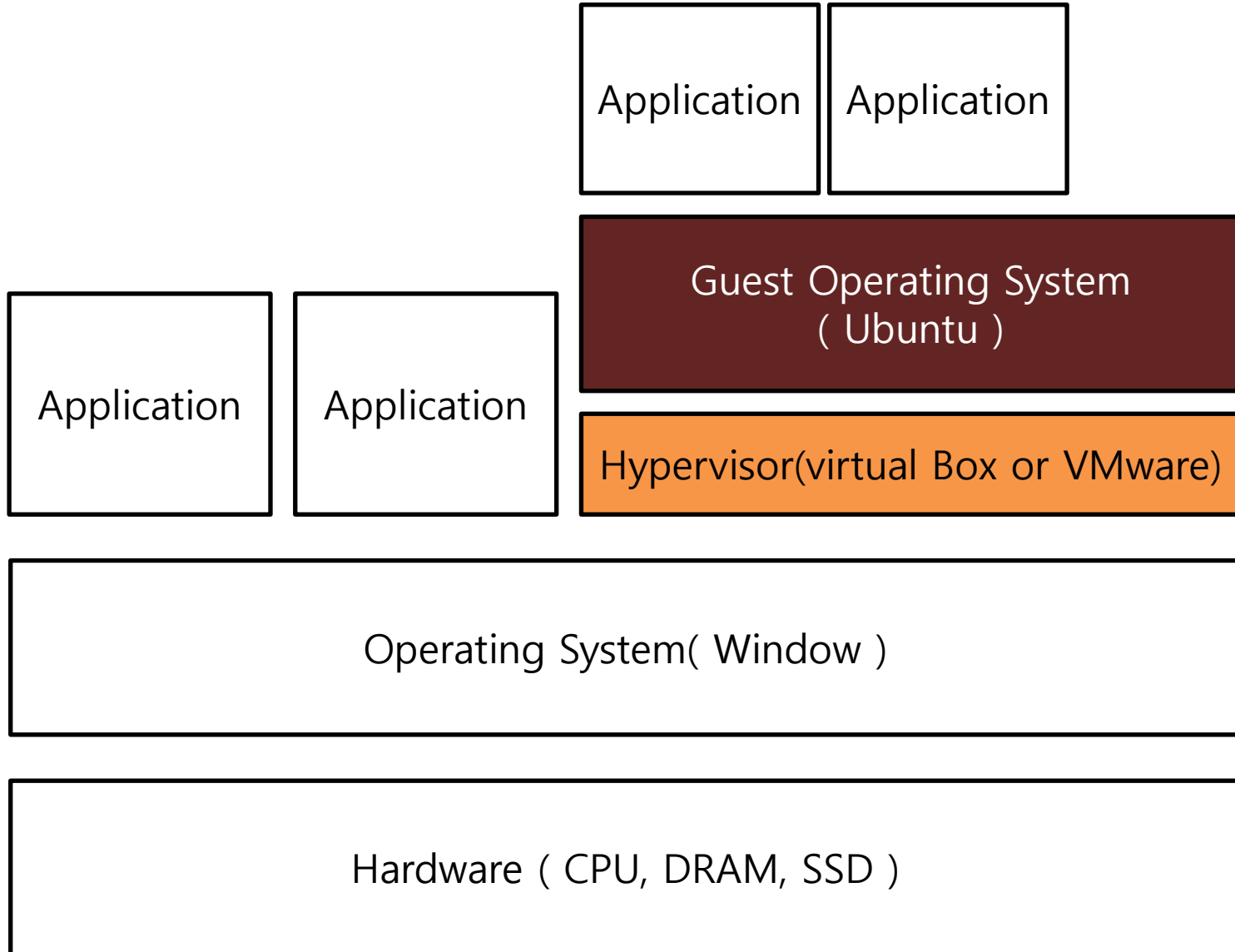
# What are we doing.



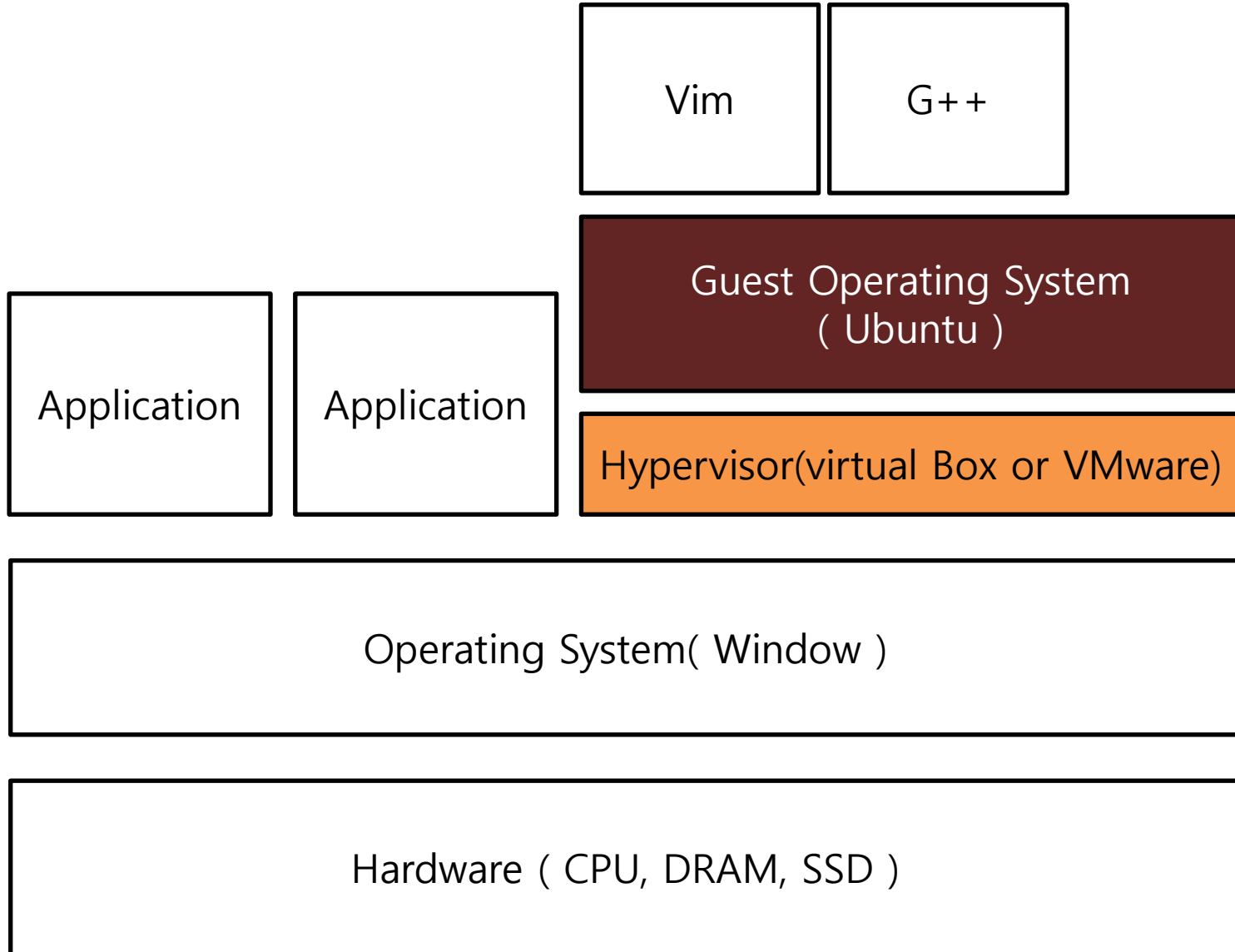
Operating System( Window )

Hardware ( CPU, DRAM, SSD )

# What are we doing.



# What are we doing.



---

# Creative Software Programming

## Lab2: g++, make, gdb

Yoonsang Lee

Fall 2019

# Today Topic

---

- How to use Terminal
- How to use Git
- G++
- Make
- GDB

# How to use Terminal

---

- Open Terminal (Shortcut CTRL + ALT + T)
- Retrieve file on current directory

```
(Shell – home directory)
```

```
$ ls
```

- Current Location

```
(Shell – home directory)
```

```
$ pwd  
/home/<user> # this is your Home Directory
```

# How to use Terminal

---

- Directory type
  - Normal directory : <dir-name>
  - Current directory : .
  - Parent directory : ..
  - Root directory : /
  - Home directory : ~
- Path type
  - Absolute address : /<dir1>/<dir2> ..
  - Relative address: : <dir1>/<dir2>

# How to use Terminal

---

- Make directory

(Shell)

```
$ mkdir <dir-name>
```

- Chang the shell working directory

(Shell)

```
$ cd <destination directory>
```

- Remove

(Shell)

```
$ rm <file-name>
```

(Shell)

```
$ rm -rf <dir-name>
```



# How to use Terminal

---

- Move source(s) to destination directory.

(Shell)

```
$ mv <source file> <destination directory>
```

(Shell)

```
$ mv <source directory> <destination directory>
```

- Rename SOURCE to DEST

(Shell)

```
$ mv <SOURCE> <DEST>
```

# How to use Terminal

---

- Copy

(Shell)

```
$ cp <source file> <destination directory>
```

(Shell)

```
$ cp <source file> <destination file>
```

(Shell)

```
$ cp -r <source directory> <destination directory>
```

# Git workflow overview

Working  
directory

Staging  
Area

Local  
Repository

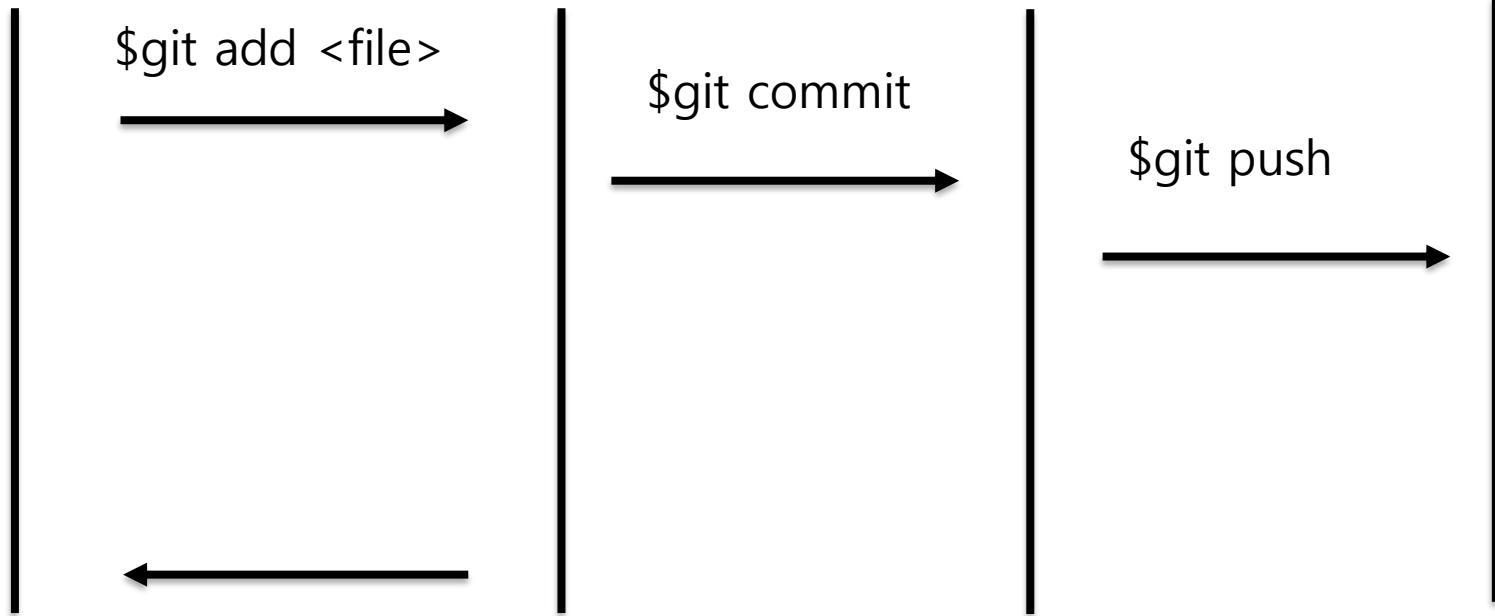
Remote  
Repository

`$git add <file>`

`$git commit`

`$git push`

`$git reset HEAD -- <file>`



# Git : staging

- Currently no modified files have been staged.

```
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        main
        main.cc

nothing added to commit but untracked files present (use "git add" to track)
```

- \$ git add \* means stage all currently modified files.

```
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git add *
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   main
        new file:   main.cc
```

# Git : unstaging

---

```
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git reset HEAD -- main
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   main.cc

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        main
```

# Git : commit

- When you commit, write briefly what you commit

```
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git commit -m "write briefly what you commit"
[master (root-commit) 9a62e89] write briefly what you commit
1 file changed, 8 insertions(+)
create mode 100644 main.cc
```

- All the commits can be found in the log.

```
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git log
commit 9a62e892988f5491056cb0abdbbf6ef9c8c7bd3c (HEAD -> master)
Author: koo <rnrudgh@gmail.com>
Date:   Wed Sep 4 10:57:25 2019 -0700

    write briefly what you commit
```

# Git : push

```
koo@ubuntu:~/Downloads/2019_ITE1015_2019193573$ git push
Username for 'https://hconnect.hanyang.ac.kr': rnrudgh@gmail.com
Password for 'https://rnrudgh@gmail.com@hconnect.hanyang.ac.kr':
warning: redirecting to https://hconnect.hanyang.ac.kr/2019_ITE1015_12226/2019_ITE1015_2019193573.git/
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://hconnect.hanyang.ac.kr/2019_ITE1015_12226/2019_ITE1015_2019193573
 * [new branch]      master -> master
```

Files · master · 2019\_ITE1015\_12226 / 2019\_ITE1015\_2019193573 · GitLab - Mozilla Firefox

Files · master · 2019\_ITE1015\_12226 / 2019\_ITE1015\_2019193573

Project Activity **Repository** Pipelines Graphs Issues 0 Merge Requests 0 Wiki

Files Commits Network Compare Branches Tags

master 2019\_ITE1015\_2019193573 / + Find File

Name	Last commit	History	Last Update
main.cc	write breifly what you commit		6 minutes ago

# What is G++ ?

---

- Open-sourced C++ compiler
- Most formats and options are the same as the default C compiler (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 & Link

- Write main.cc, print.cc

(Shell – working directory)

```
$ vi main.cc
```

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

(Shell – working directory)

```
$ vi main.cc print.cc
```

```
#include <iostream>  
  
void print_hello() {  
    std::cout << "hello world!" <<  
endl;  
}
```

# Example : Compile & Link

- Compile and link the two source files (main.cc, print.cc)

(Shell – working directory)

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

(Shell – working directory)

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

- Run the created executable

(Shell – working )

```
$ ./hello_world
```

# Make

---

- Build tools that have been around for a long time on Unix operating systems
  - Rules for how to compile and link the source to create an executable

# Makefile

---

- When “make” is run, find Makefile (or makefile) in that directory and runs it as usual
- How to write Makefile

```
target: prerequisites  
<TAB>command1  
<TAB>command2
```

- target : File or state to create( such as.o or executable) ≡)
- prerequisites : List of files needed to create target
- command(s) :Each step command to create a target. <Tab> must be placed before the command.

# Example: Writing / Running makefile

- Write makefile

(Shell – working directory)

\$ vi Makefile

```
hello_world: main.o print.o
    g++ -o hello_world main.o print.o
main.o: main.cc
    g++ -c main.cc
print.o: print.cc
    g++ -c print.cc
clean:
    rm hello main.o print.o
```

# Example: Writing / Running makefile

---

- Execute makefile (1) : generate executable file

(Shell – working directory)

```
$ make
```

- Execute makefile (2) : Remove Executable file and All object files

(Shell – working directory)

```
$ make clean
```

# GDB

---

Debugging tools - help you find the wrong parts of your program by checking its status when the program is running or when it crashes.

When you build a program, you need to give it the `-g` option to see the information you need.

**gdb** [options] <command>

- <command> : If the current directory is not in your PATH, you must include `./`.
- Basic command
  - `r` [arguments] : Run the given command.
  - `bt` : backtrack. Show current call stack status.
  - `up/down` [steps] : Move up / down a given step from the current position of the call stack.
  - `p` <variable> : Display the value of a given variable.
  - `q` : exit gdb process.
  - Use more easy-to-use improved programs such as `cgdb` and `ddd`

# Example

(Shell – working directory)

```
$ vi test.cc
```

```
void IncorrectAccess(int* array, int i, int n) {  
    if (i < n) {  
        array[i] = 0;  
        IncorrectAccess(array, i + 1, n);  
    }  
}  
int main() {  
    int array[10];  
    IncorrectAccess(array, 0, 20);  
    return 0;  
}
```

(Shell – working directory)

```
$ g++ -o test test.cc
```

```
$ gdb ./test
```

```
...
```

```
(gdb)
```

```
...
```