LINUX PROGRAMMING

Unix/Linux Fundamental Command
(1 Week)

UNIX SYSTEM

UNIX ARCHITECTURE

- *Applications :* 우리가 알고 있는 프로그램. *Ex)* 카카오톡
- *Shell : special application,* 다른 *application* 들을 위한 *interface*를 제공.
- *System calls : kernel interface.*
- *Library routines : system call* 들을 사용하기 쉽게 정의해 놓은 *function library.*
- *Kernel : Core software of OS.*

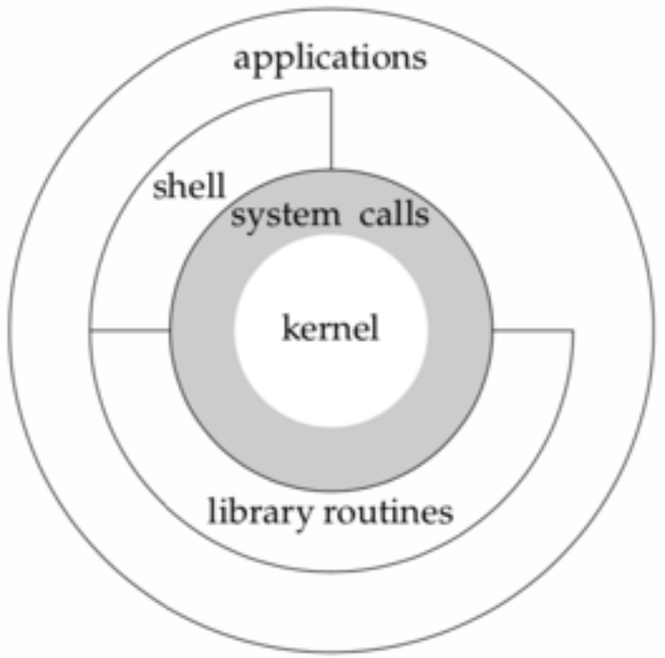
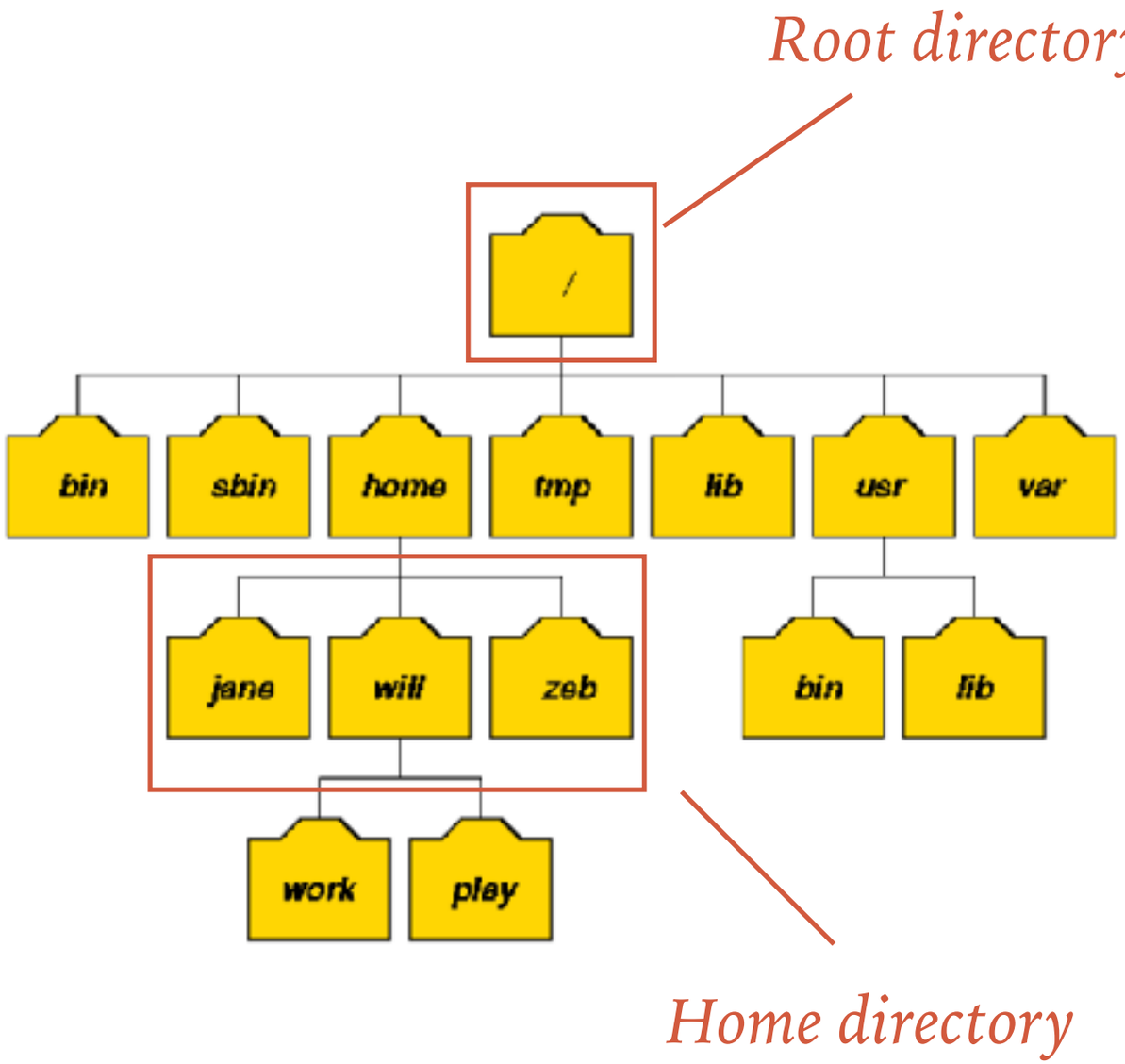


Figure 1.1 Architecture of the UNIX operating system

리눅스 는 *GNU operation system* 에서 주로 사용하는 커널로써, *linux* 와 *GNU*의 조합을 리눅스(*GNU/LINUX*) 라고 부른다.

FILE SYSTEM



➤ *Root directory (/)*

*Linux file system*의 최상의 *directory.*

➤ *Home directory (/home)*

유저들의 *directory* 존재한다.

➤ *Working directory*

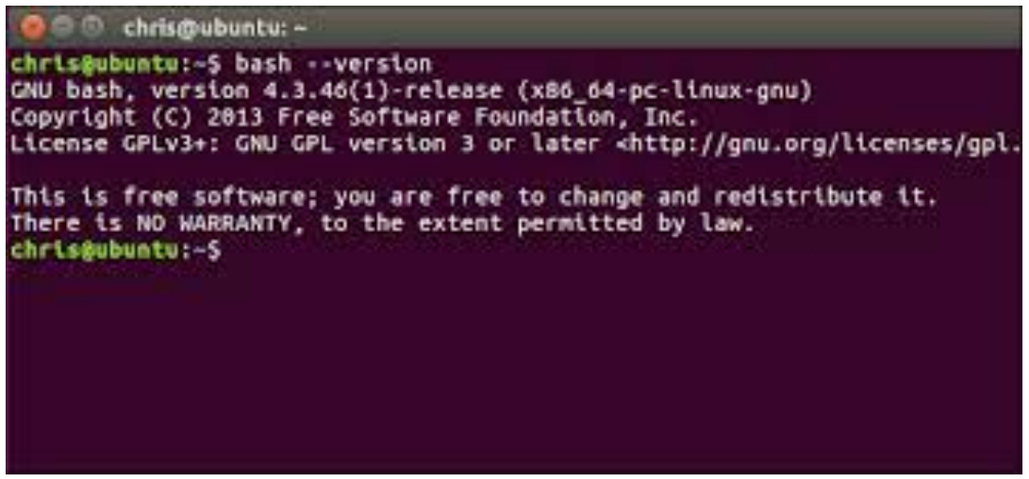
작업하고 있는 *directory*

'/' 를 루트로 가진 트리 구조

WORKING INTERFACE

- CLI (Command - Line Interface)란 ?

가상 터미널 또는 터미널을 통해 사용자와 컴퓨터 상호 작용하는 방식을 뜻한다.

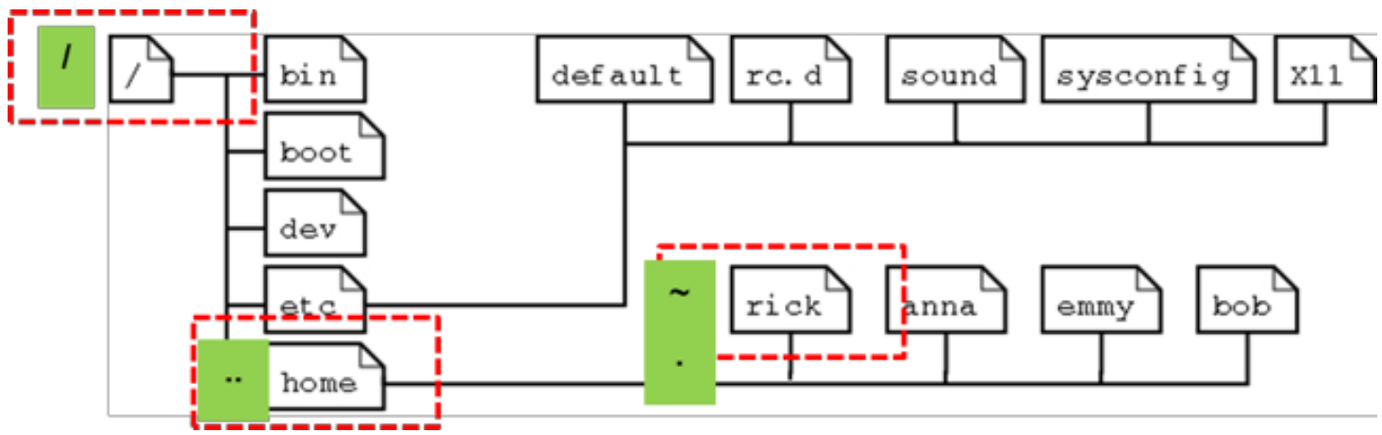


- GUI (Graphical - User Interface) 란 ?

사용자가 편리하게 사용할 수 있도록 입출력 등의 기능을 알기 쉬운 아이콘 따위의 그래픽으로 나타낸 것이다.

DIRECTORY SYMBOL IN LINUX

- Symbol can express directory
  - . : 현재 directory 를 나타낸다.
  - .. : 상위 directory 를 나타낸다.
  - / : 최상의 directory 를 나타낸다.
  - ~ : 유저의 home directory 를 나타낸다.

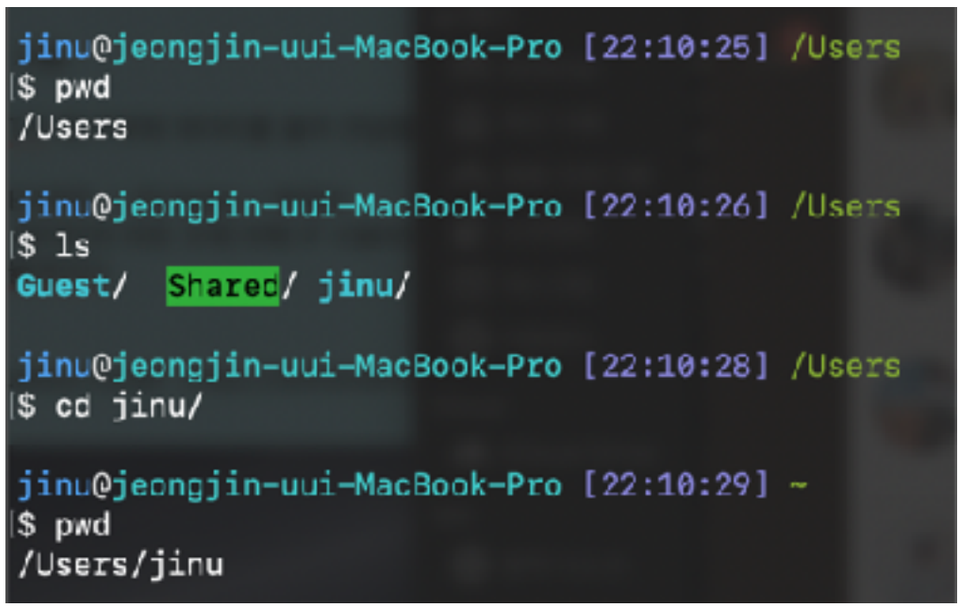



```
user@user-virtual-machine:~$ pwd
/home/user
user@user-virtual-machine:~$ cd ..
user@user-virtual-machine:/home$ pwd
/home
user@user-virtual-machine:/home$ cd /
user@user-virtual-machine:/$ pwd
/
user@user-virtual-machine:/$ cd ~
user@user-virtual-machine:~$ pwd
/home/user
```

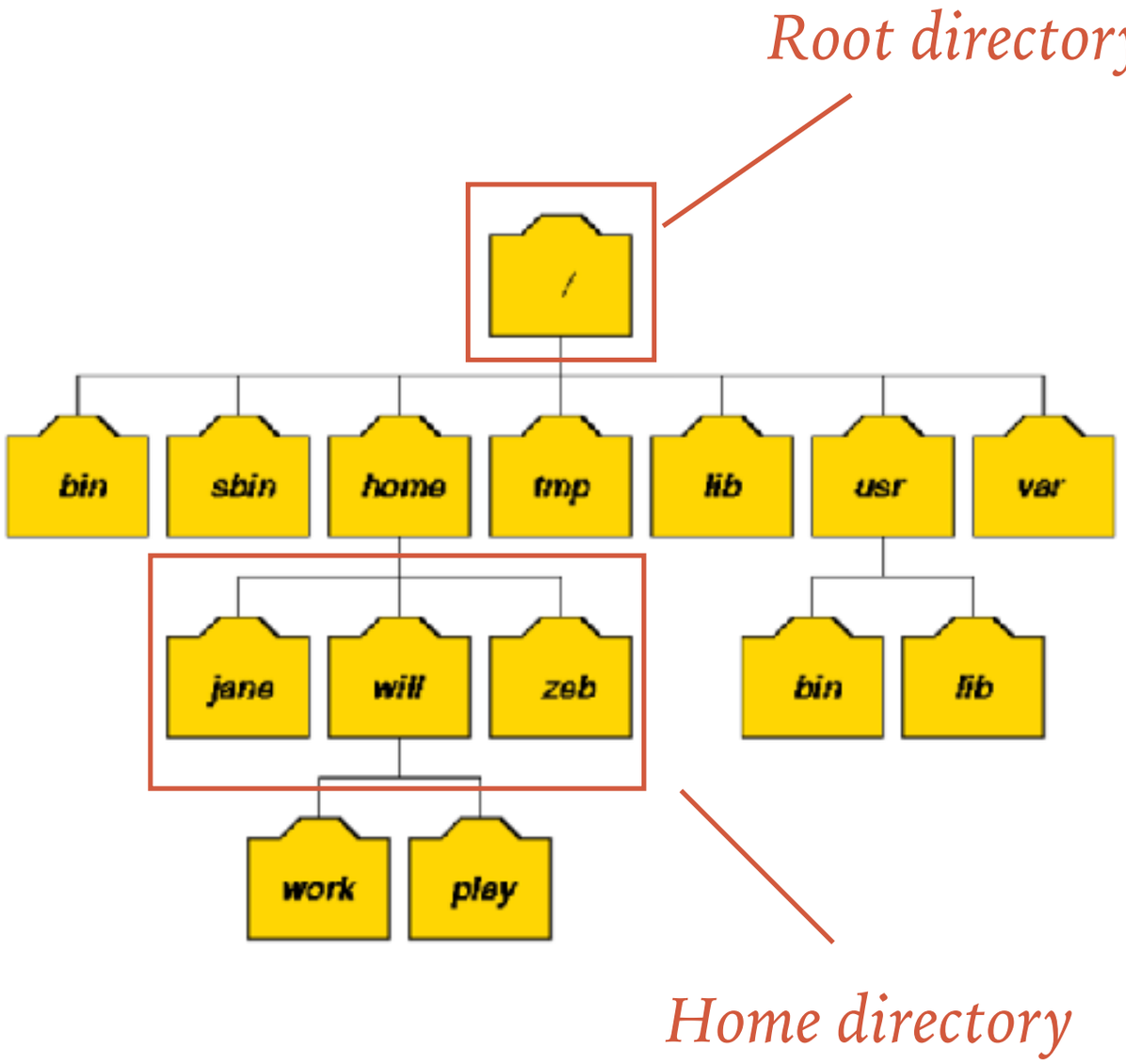
ABSOLUTE PATH & RELATIVE PATH

- Absolute path
  - / (root) directory로 부터 전체 경로를 표현
  - Ex) /home/user/

- Relative path
  - 현재 작업 directory 를 기준으로 경로를 표현
  - Ex) ../user Or



PATH



Working directory = play

Dirctory	절대 경로	상대 경로
work	/home/will/play	../play
Usr	/usr	../../../usr
Jane	/home/jane	../../jane
Play	/home/will/play	./ , ~/play

BASIC COMMANDS

LINUX'S BASIC COMMAND: LS

- ls — list
  - Directory 와 file들의 이름을 출력한다.
  - Option 들을 추가하여 더 많은 정보를 출력할 수 있다.

- How to use
  - - ls : 현재 작업 directory의 contents들을 출력한다.
  - - ls <directory> : 지정한 directory의 contents들을 출력한다.

```
jinu@jeongjin-uui-MacBook-Pro [21:34:52] ~/Desktop/git/Hantor_Page
$ ls
README.md  backend/   dist/      frontend/

jinu@jeongjin-uui-MacBook-Pro [21:35:03] ~/Desktop/git/Hantor_Page
$ ls ~/Desktop/git
Chat_Web/           WebRTC/             door/               sample-app/
Hantor_Page/        WebSystem_Project/  mongoose/           studyhacking/
SmartDoor/          comments_collector/ project3/
```

LINUX'S BASIC COMMAND : LS

Option:

```
-a      Include directory entries whose names begin with a dot (.).
```

```
-l      (The lowercase letter ``ell''.)  List in long format.  (See
        below.)  If the output is to a terminal, a total sum for all the
        file sizes is output on a line before the long listing.
```

-a : 모든 파일을 출력한다.

-l : 파일을 세부적으로 출력한다.

```
jinu@jeongjin-uui-MacBook-Pro [21:42:09] ~/Desktop/git/Hantor_Page
$ ls -a
./          .git/       README.md   dist/
../         .gitignore  backend/    frontend/

jinu@jeongjin-uui-MacBook-Pro [21:42:10] ~/Desktop/git/Hantor_Page
$ ls -l
total 8
-rw-r--r--   1 jinu  staff    14B  2 22 17:55 README.md
drwxr-xr-x   9 jinu  staff   288B  3  7 10:27 backend/
drwxr-xr-x  11 jinu  staff   352B  2 24 23:00 dist/
drwxr-xr-x  11 jinu  staff   352B  2 26 23:27 frontend/

jinu@jeongjin-uui-MacBook-Pro [21:42:13] ~/Desktop/git/Hantor_Page
$ ls -al
total 16
drwxr-xr-x   8 jinu  staff   256B  2 24 19:34 ./
drwxr-xr-x  13 jinu  staff   416B  2 28 21:02 ../
drwxr-xr-x  14 jinu  staff   448B  3 13 17:39 .git/
-rw-r--r--   1 jinu  staff    13B  2 22 17:55 .gitignore
-rw-r--r--   1 jinu  staff    14B  2 22 17:55 README.md
drwxr-xr-x   9 jinu  staff   288B  3  7 10:27 backend/
drwxr-xr-x  11 jinu  staff   352B  2 24 23:00 dist/
drwxr-xr-x  11 jinu  staff   352B  2 26 23:27 frontend/
```

FILE

```
-rw-r--r--   1 jinu  staff    14B  2 22 17:55 README.md
drwxr-xr-x   9 jinu  staff   288B  3  7 10:27 backend/
```

1. 파일 종류, *2.* 파일 허가, *3.* 파일 링크 수, *4.* 파일 소유자, *5.* 파일 소유 그룹, *6.* 파일 크기, *7.* 파일 수정 날짜. *8.* 파일 이름

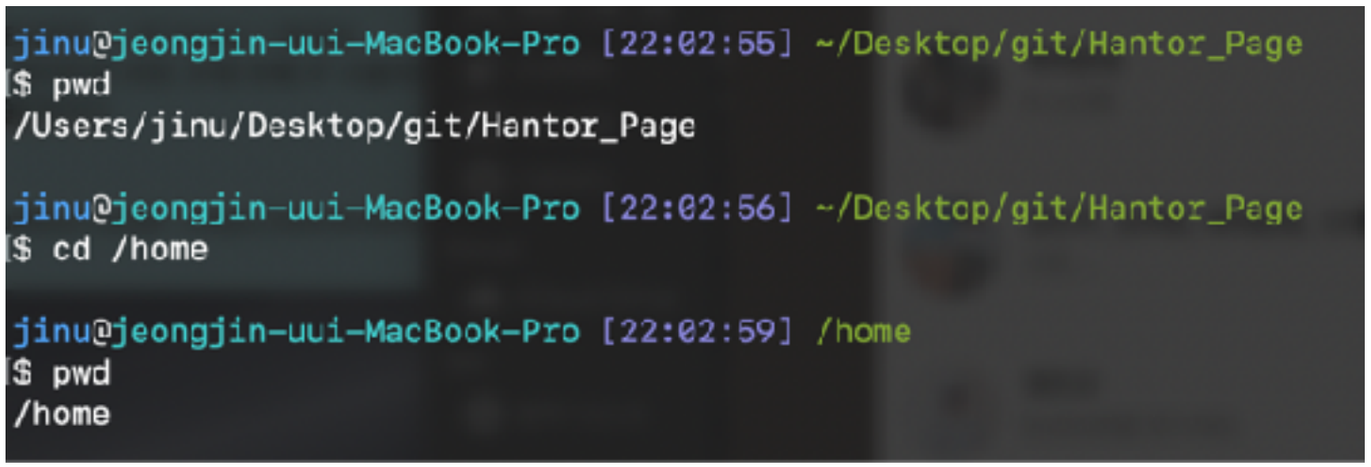
```
drwxr-xr-x  2 jinu  staff    64B  6 23 23:50 directory/
-rwxr-xr-x  1 jinu  staff     6B  6 23 23:50 execute_file*
-rw-r--r--  1 jinu  staff     6B  6 23 23:50 test_file.txt
```

User *Group* *Other user*

LINUX'S BASIC COMMAND: PWD, CD

- pwd — print working directory
  - 현재 작업 directory 경로를 출력한다.
- cd — change directory
  - 작업 directory를 바꾼다.

- How to use
  - $ pwd
  - $ cd <directory>



LINUX'S BASIC COMMAND : MKDIR

- mkdir — make directory
  - 새로운 directory를 만든다.

- How to use
  - $ mkdir <directory's name>

LINUX'S BASIC COMMAND : MV

- mv — move
  - File 의 이름을 변경 또는 이동시킨다.

- How to use
  - $ mv <source> <target>

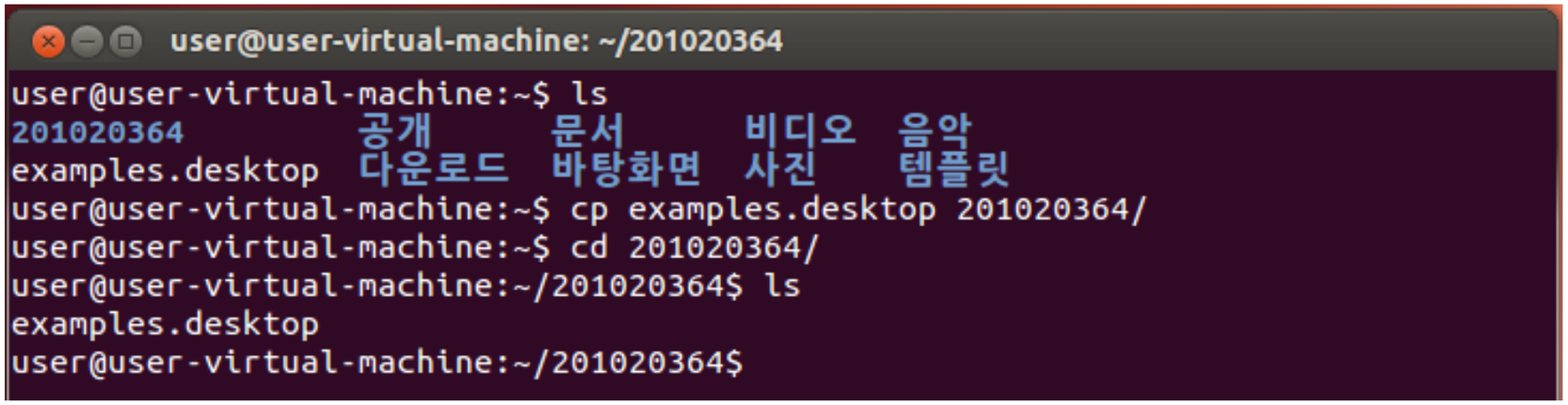
```
jinu@jeongjin-uui-MacBook-Pro [00:46:00] ~/Desktop/git/test/directory
$ ls
pre.txt

jinu@jeongjin-uui-MacBook-Pro [00:46:01] ~/Desktop/git/test/directory
$ mv pre.txt next.txt

jinu@jeongjin-uui-MacBook-Pro [00:46:11] ~/Desktop/git/test/directory
$ ls
next.txt

jinu@jeongjin-uui-MacBook-Pro [00:46:12] ~/Desktop/git/test/directory
$ mv next.txt ../

jinu@jeongjin-uui-MacBook-Pro [00:46:19] ~/Desktop/git/test/directory
$ ls

jinu@jeongjin-uui-MacBook-Pro [00:46:20] ~/Desktop/git/test/directory
$ cd ..

jinu@jeongjin-uui-MacBook-Pro [00:46:22] ~/Desktop/git/test
$ ls
directory/     execute_file*  next.txt       test_file.txt
```

LINUX'S BASIC COMMAND : CP

- cp — copy
  - 파일을 복사한다. (파일 이름을 바꿔서 복사할 수도 있다.)

- Major option
  - -r : copy every files include recursive, sub-directory

- How to use
  - $ cp (-r) <source> <destination>



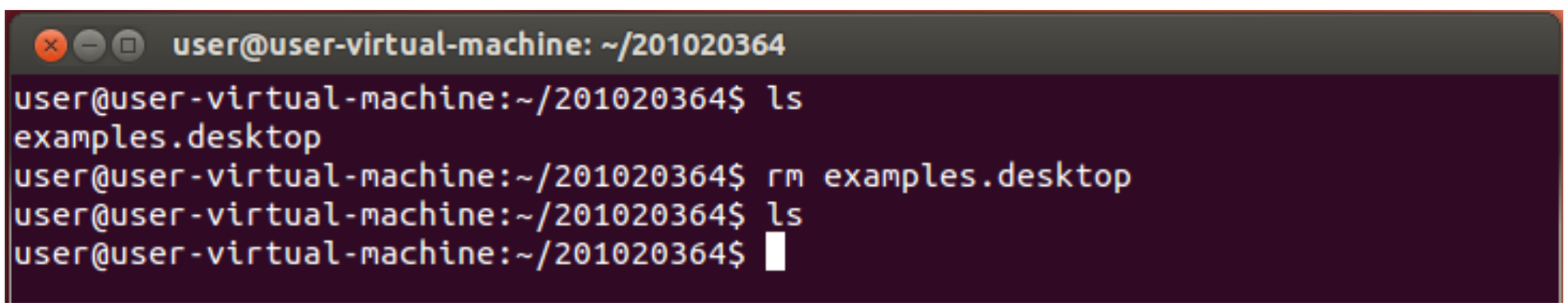
LINUX'S BASIC COMMAND: RM

- rm — remove
  - 파일 또는 directory를 삭제한다.

  (삭제할 때 조심!!!! 복구 거의 불가)

- Major option
  - -r : Delete every files include recursive, sub-directory
  - -f : force, do not ask again for deleting

- How to use
  - $ rm (-rf) <file name or directory)



LINUX'S BASIC COMMAND: CAT

- cat — concatenate and print files
  - 파일 안에 text를 보여준다.

- How to use
  - $ cat <file name>

```
$ cat main.c
#include<stdio.h>
int main(){
        printf("hello\n");
        return 0;
}
```

SHELL COMMAND

- | (pipe)
  - 두 개의 명령어를 연결한다.

```
jinu@jeongjin-uui-MacBook-Pro [22:21:53] ~/Desktop/git/Hantor_Page
$ ls
README.md  backend/   dist/      frontend/  test/

jinu@jeongjin-uui-MacBook-Pro [22:21:55] ~/Desktop/git/Hantor_Page
$ ls | grep e
backend/
frontend/
test/
```

- >, >> (redirectory)

EX) (command) > (file)

  - Command의 출력을 file의 contents로 저장된다.

SHELL COMMAND

```
jinu@jeongjin-uui-MacBook-Pro [22:25:56] ~/Desktop/git/Hantor_Page/test
$ ls -al
total 0
drwxr-xr-x  2 jinu  staff    64B  3 16 22:25 ./
drwxr-xr-x  9 jinu  staff   288B  3 16 22:25 ../

jinu@jeongjin-uui-MacBook-Pro [22:26:00] ~/Desktop/git/Hantor_Page/test
$ ls -al > example

jinu@jeongjin-uui-MacBook-Pro [22:26:13] ~/Desktop/git/Hantor_Page/test
$ cat example
total 0
drwxr-xr-x  3 jinu  staff    96B  3 16 22:26 ./
drwxr-xr-x  9 jinu  staff   288B  3 16 22:25 ../
-rw-r--r--  1 jinu  staff     0B  3 16 22:26 example
```

> : 파일 내용을 새로 쓴다.

>> : 파일안에 있는 내용을 유지한 상태로 그 뒤에 쓴다.

SUPER USER DO

- 관리자 권한을 획득하여, 권한이 필요한 시스템 작업들을 할 수 있다.

- How to Use
  - $ sudo <command>

APT-GET(ADVANCED PACKAGING TOOL)

- 우분투 (Ubuntu)를 포함안 데비안(Debian)계열의 Linux에서 쓰이는 Package 관리 명령어 도구
- apt-get 명령어를 사용하기 위해서는 sudo 권한이 있어야한다.

- How to Use
  - Update
    - sudo apt-get update
  - Install
    - sudo apt-get install <package name>
  - Remove
    - sudo apt-get remove <package name>
  - Reinstall
    - sudo apt-get reinstall <package name>

DEVELOPMENT TOOL

Vim, Gcc

INTRODUCE

- **Vim** : It is editor supported in Linux OS

- **GCC** : Gnu Compiler Collector, It is compiler system produced by the GNU project supporting various programming languages.

VIM

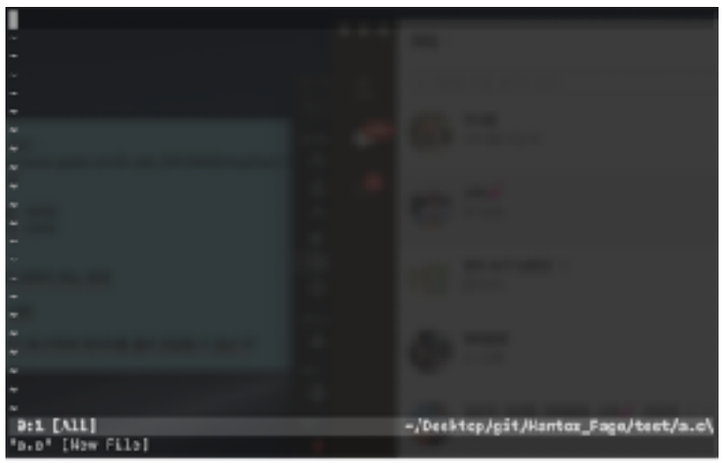
- Mode
  - Insert mode (i)
    - 파일 편집
  - Visual mode (v)
    - 드레그 역할을 한다.

VIM

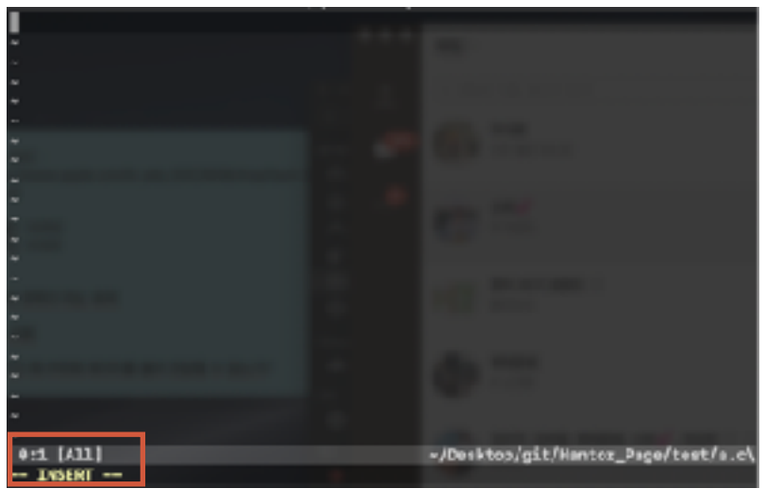
- Option
  - Find option
    - /(argument) => contents에서 argument를 찾는다(n = forward word, N = backward word)
  - Delet option
    - dd => 커서 위에 있는 line을 삭제한다.
  - Store option
    - :w => 현재 상태를 저장한다.
  - Quit option
    - :q => vim editor를 나간다.
  - Store & Quit option
    - :x => 현재 상태를 저장한 뒤 editor를 나간다.
  - Reset option
    - u => 이전 상태로 되돌린다.
  - Copy option
    - Y => 커서 위에 있는 text를 복사
  - Put Option
    - P => 복사 했던 text를 붙여넣기

VIM

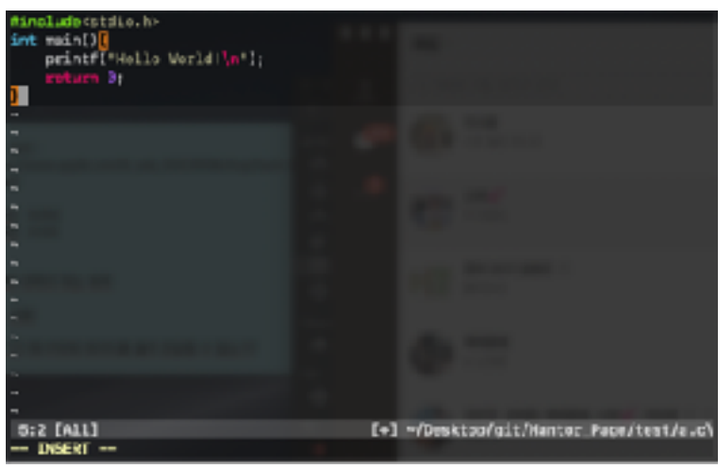
```
jinu@jeongjin-uui-MacBook-Pro [23:46:38] ~/Desktop/git/Hantor_Page/test
$ vim a.c
```



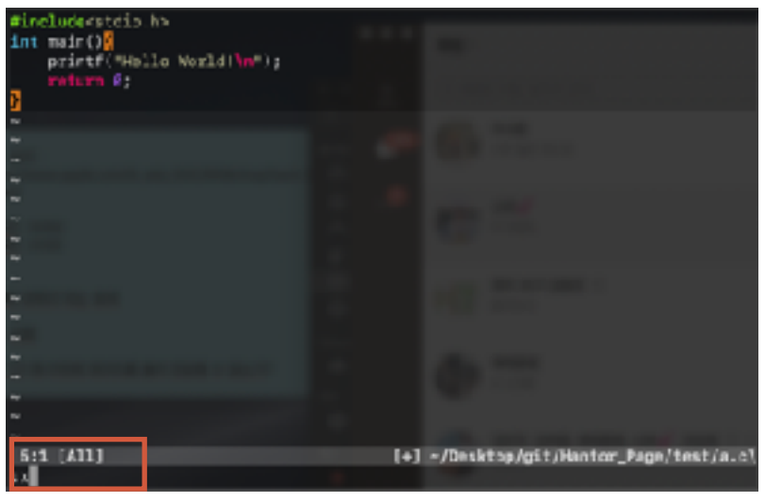
(1)



(2) 'i' 입력

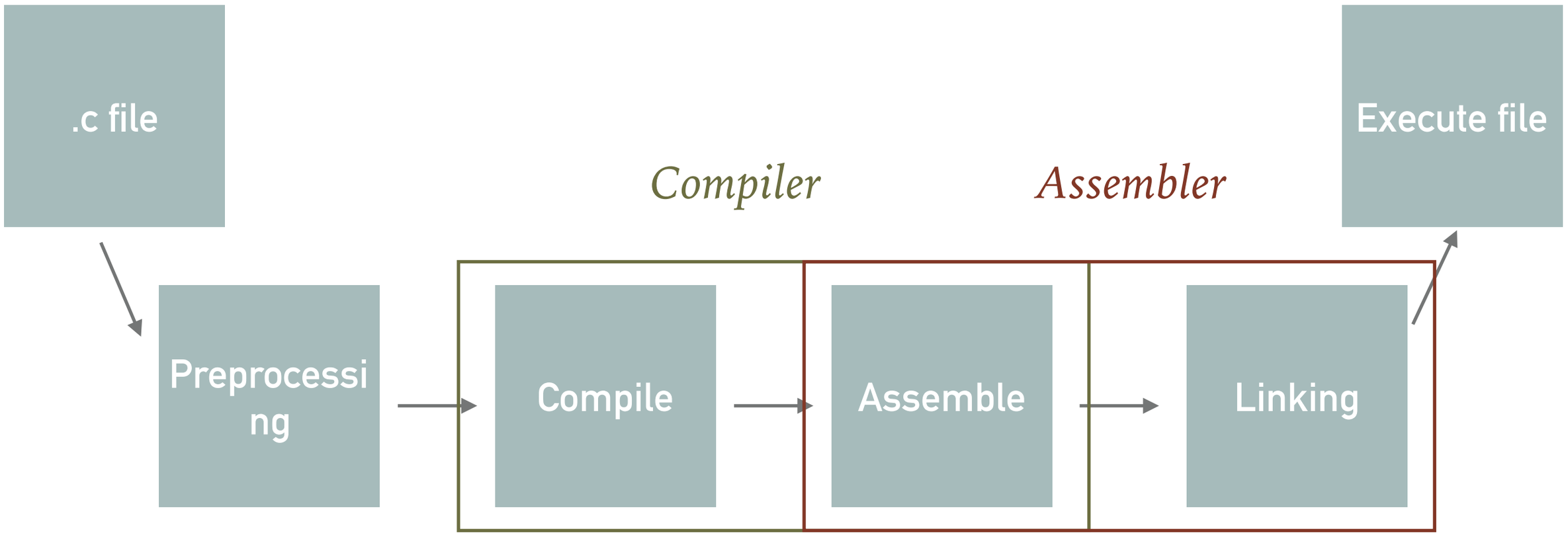


(3)



(4) Esc + ':x'

FROM C FILE TO EXECUTE FILE



Processing : Macro 과정을 처리한다.

*Compile : c file*을 *assembly code*로 변환한다.

*Assemble : assembly code*를 *object code*로 변환한다.

Linking : 여러 *object code*를 묶어서 실행 파일로 만든다.

GCC

➤ Assembly code 생성 (-S) :

$ gcc -S main.c

=> main.s

➤ Object code 생성 (-c) :

$ gcc -C main.c

=> main.o

➤ Execute file 생성 :

$ gcc -o main main.c

=> main

GCC

```
jinu@jeongjin-uui-MacBook-Pro [23:53:54] ~/Desktop/git/Hantor_Page/test
$ gcc -o a a.c

jinu@jeongjin-uui-MacBook-Pro [23:53:56] ~/Desktop/git/Hantor_Page/test
$ ls
a*        a.c       example

jinu@jeongjin-uui-MacBook-Pro [23:54:01] ~/Desktop/git/Hantor_Page/test
$ ./a
Hello World!
```

-o option :

$ gcc -o <out file> <source file> : source file을 out file이라는 실행파일로 들겠다.

How to execute :

$ PATH/<file>

PRACTICE

- https://drive.google.com/open?id=1mimtlkw5Cf6fkBpLNCxbngRaywkHdaOX
- 위 링크에서 practice0.zip 파일을 받고 알집을 푼다.
- Practice0 directory가 생성된다.
- Terminal 에서 practice0 directory로 이동한다.
- $make 명령어를 실행한다.
- 그러면, 여러 directory가 생긴다. 그 안에 특별한 파일 하나와 힌트 파일들을 만들었다. 힌트를 보고 특별한 파일을 찾고 특별한 파일을 box directory 위에 이동시킨다.
- 이동시킨 파일이 특별한 파일이 맞다면 Success 문구가 terminal에서 보일 것이다.