

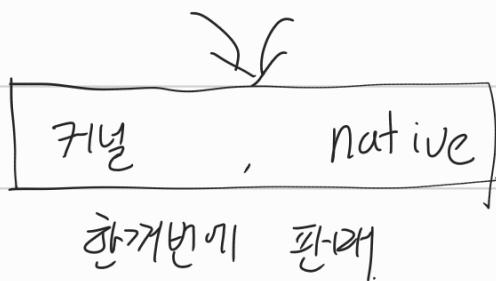
# LINUX

2. ⇒ 운영체제

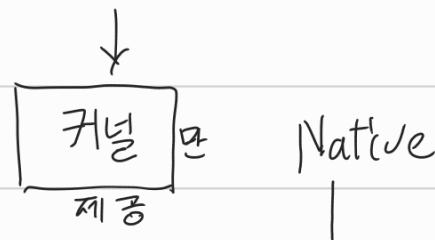
↳ 2. ⇒ 컴퓨터 단계 만들어 줍니다.

제어? → 풀겁지X ⇒ 서비스만 제공.

Windows vs IOS



Linux



사용자들이 다양하게 만들  
(package)

Kernel: OS의 기본 기능을 실행하는 부분

So, 원하는 package 을 골라서

네이티브: 특정 하드웨어나 OS에서 그대로 실행하는 부분

사용 ex) ubuntu

## Embedded Linux

↳ ex) 라즈베리 파이 → 하드웨어 범위↑

(Linux 전용↑)

통합현상 ⇒ RAM 고체

유닉스 → 리눅스

가상머신

WSL

(윈도우에서 리눅스를 서브로 사용하는 것)

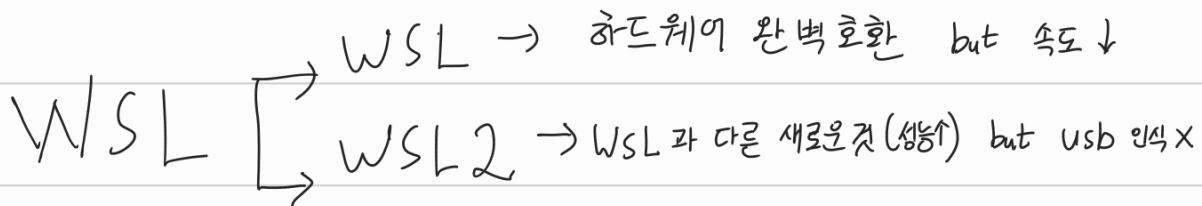
단, 완성된 기능X (usb 인식 풍선적X)

Virtual box ← 호환  
VMware player ← 호환

Python에서 numpy → third part package 놓고 함  
↳ 쉽다 ⇒ 제한적

프로그램 만들 때 고려 → 제한된 자원, 돈, 시간

프로그램 핵심 3가지



Console - 터미널 일종, 시스템 제어 등을 위한 특수목적 터미널 (제어장치)

Terminal - 시스템 접속하여 입출력을 가능하게 하는 단말 장치 (형태: 하드웨어 or 소프트웨어)

↳ 콘솔의 집합체

SSH 서버 - 보안이 강화된 telnet 이자고 생각, 텍스트 모드로 작성  
(secure shell) (원격접속)

Unix ————— Linux  
(GNU Linux)  
↳ 파생된 언어 (C언어)

Ubuntu 20.04 LTS  
↳ 기술 지원 해주는 오래  
(Long Term Service)  
최대 5년

# < 실질 LINUX 활용 >

\* 하는 법 : PowerShell에서 Ubuntu로 → WSL 명령어

cd → 홈

pwd → 현재경로

clear → 다 없애기

\* Mount — 물리적인 장치를 즉 디렉터리에 연결 시켜주는 것

└ Window : 윈도우가 mount를 자동으로 해줌

└ Linux : 관리자가 직접 특정 디렉터리에 붙이는 작업 수행

\* CPU에서는 양의 정수만 표현 가능

에스키코드는 영이만 사용 가능  
(6 bit)

(리눅스의 목적)

\* 소프트웨어 잘하는 법

→ 잘해야 일반화가 잘된다.

일반화 ←———— 추상화

(꼭 필요한 것만 선택)

이쁘게?

리눅스 시스템 사용하면서

할 것

ex) 80% ←———— 20%

개별자 입장 (고객 needs 파악)

소비자 입장

기술 → 의사, 일자, 등록 프로그램 → IT들

일반화

개별고, 일자O

```
user@DESKTOP-M4IU114: ~
user@DESKTOP-M4IU114: ~$ ls -l
total 8
drwxr-xr-x 2 user user 4096 Mar 30 17:09 project
drwxr-xr-x 2 user user 4096 Mar 30 17:10 workspace
user@DESKTOP-M4IU114: ~$ chmod 700 project
user@DESKTOP-M4IU114: ~$ ls -l
total 8
drwx----- 2 user user 4096 Mar 30 17:09 project
drwxr-xr-x 2 user user 4096 Mar 30 17:10 workspace
user@DESKTOP-M4IU114: ~$
```

## Chmod

### 명령어 사용법 #

```
chmod [옵션] [모드] [파일]
```

- if, 현재 어떤 허가들이 있는지 보기 위해서는 다음과 같이 입력한다 :

```
ls -l 파일이름
```

### [옵션] #

-R : 하위 파일과 디렉토리 모든 권한을 변경한다.

-v : 실행되고 있는 모든 파일을 나열한다.

-c : 권한이 변경된 파일내용을 출력한다.

## 1. 문자열 모드 #

Chmod 옵션 (reference)(operator)(modes) 파일

- reference(대상) :

- u : user의 권한 (사용자의 권한)
- g : group의 권한 (파일의 group 멤버인 사용자의 권한)
- o : other의 권한 (user, group의 멤버가 아닌 사용자의 권한)
- a : all의 권한 (위의 셋을 포함하는 모든 사용자의 권한)

- operator :

- + : 해당 권한을 추가한다.
- : 해당 권한을 제거한다.
- = : 해당 권한을 설정한데로 변경한다.

- modes :

- r : read 권한 (읽기)
- w : write 권한 (쓰기)
- x : execute 권한 (실행)
- : 사용권한없음

- 예제 :

chmod ug+rw sample : sample파일의 user나 group 멤버들에게 읽기, 쓰기 권한을 추가  
chmod u=rwx,g+x sample : sample파일의 user는 읽기,쓰기,실행 권한 부여, group 멤버들에게 실행권한 추가

## 2. 8진법 수 모드 # ↴각인

Chmod 옵션 (8진법 수) 파일

- 잠깐! 참고 사항

- -rwxr-xr-x : 파일 접근 권한 분류 표기로, 처음 -는 파일 분류 타입.
- 파일 타입 : d 디렉토리, l(소문자 L) 링크 카운터, s 소켓, p 파일, - 일반, c 특수문자, b 특수 블럭

- rwxr-xr-x = 755

처음3개문자 = user의 권한

중간3개문자 = group의 권한

마지막3개문자 = other의 권한

r은 파일 읽기 = 4, w는 파일 쓰기 = 2, x는 파일 실행 = 1로, 3개문자씩 수를 더해서 쓴다.

- 추가 사항 : 특수 권한

777 = 일반적인 8진법 형태

4777= SetUid 설정 때 4000을 더함

2777= SetGid 설정 때 2000을 더함

1777= Sticky bit 설정 때 1000을 더함

- 예제 :

chmod 777 test : test 파일의 user, group, other의 권한을 모두 rwx로 변경.

chmod 4755 test : test파일의 user id설정을 지정하고, user에게 rwx 권한 부여, group과 other에게 r-x권한부여

링크 9.4

```

user@DESKTOP-M4IU114: ~
user@DESKTOP-M4IU114: $ ls -l
total 8
drwxr-xr-x 2 user user 4096 Mar 30 17:09 project
drwxr-xr-x 2 user user 4096 Mar 30 17:10 workspace
user@DESKTOP-M4IU114: $ chmod 700 project
user@DESKTOP-M4IU114: $ ls -l
total 8
drwxr----- 2 user user 4096 Mar 30 17:09 project
drwxr-xr-x 2 user user 4096 Mar 30 17:10 workspace
user@DESKTOP-M4IU114: $

```

VI → 편집모드

```

user@DESKTOP-M4IU114: /mnt/c/Users/PC-16
user@DESKTOP-M4IU114: /mnt/c/Users/PC-16 $ vi my.py
user@DESKTOP-M4IU114: /mnt/c/Users/PC-16 $ python
Command 'python' not found, did you mean:
  command 'python3' from deb python3
  command 'python' from deb python-is-python3
user@DESKTOP-M4IU114: /mnt/c/Users/PC-16 $ vi hello.py
user@DESKTOP-M4IU114: /mnt/c/Users/PC-16 $ apt search

```

A, I (단축키) - 입력모드

ESC (단축키) - 빠져나가기

:q! → 저장X 빠져나가기

~~~~~  
저장무시

:X → 저장X ⇒ 저장후 나가기

~~~~~  
저장O ⇒ 나가기

단축키



## vi 단축키

- 자주 쓰는 단축키

### 삽입

키	기능
i	커서 위치에 Insert
I	줄 맨 앞에서 Insert
a	커서 다음에 Insert
A	줄 맨 뒤에서 Insert
o	커서 아래로 한 줄 띄우고 Insert
O	커서 위로 한 줄 띄우고 Insert

### 이동

키	기능
w	단어 첫 글자 기준으로 다음으로 이동
W	공백 기준으로 다음(단어의 시작)으로 이동
b	단어 첫 글자 기준으로 이전으로 이동
B	공백 기준으로 이전으로 이동
e	단어 마지막 글자 기준으로 다음으로 이동
E	공백 기준으로 다음(단어의 끝)으로 이동
gg	문서 맨 앞으로 이동
G	문서 맨 아래로 이동
^	문장 맨 앞으로 이동
\$	문장 맨 뒤로 이동

## 이동

키	기능
w	단어 첫 글자 기준으로 다음으로 이동
W	공백 기준으로 다음(단어의 시작)으로 이동
b	단어 첫 글자 기준으로 이전으로 이동
B	공백 기준으로 이전으로 이동
e	단어 마지막 글자 기준으로 다음으로 이동
E	공백 기준으로 다음(단어의 끝)으로 이동
gg	문서 맨 앞으로 이동
G	문서 맨 아래로 이동
^	문장 맨 앞으로 이동
\$	문장 맨 뒤로 이동

## 검색

키|기능 -|- /| 해당 word를 검색, `n`과 `N`으로 다음/이전 찾기

## 편집

키	기능
dd	현재 줄 잘라내기
yy	현재 줄 복사하기
p	붙여넣기
u	실행취소 (Undo)
ctrl + r	재실행 (Redo)
v	Visual모드
y	복사
c	잘라내기

# 저장

키	기능
:w	저장
:q	닫기
:q!	저장하지 않고 닫기
:wq	저장하고 닫기
:수자	지정한 줄 번호로 이동

version 1.1  
April 1st, 06

## vi / vim 단축키 모음

<b>Esc</b>	명령 모드																	
~ 대소문자 전환 ↖ 마크로 이동	! 외부 명령 ↖ 마크로 실행	@ 매크로 실행	# 이전 검색	\$ 줄 끝으로 이동 % 일치하는 패턴 찾기	^ 줄의 첫 글자	& 반복	* 다음 검색	( 문장 시작	) 문장 끝	아래줄로 이동 0 줄의 처음	- 이전 줄 + 다음 줄 = 들어쓰기							
1	2	3	4	5	6	7	8	9	0	-	+							
Q 실행 모드 ↖ 마크로 기록	W 다음 WORD W 단어 단어	E 끝 WORD e 단어 끝	R 수정 모드 r 한 문자 교체	T 뒤로 검색 t 한 문자 검색	Y 줄단위 복사 y 텍스트 복사	U 줄단위 실행취소 u 실행취소	I 줄 시작에서 삽입 i 편집 모드	O 행 위에 삽입 o 행 아래에 삽입	P 커서 이전에 봉이날기 P 커서 이후에 봉이날기	{ 문단 시작 } 문단 끝	{ 기타 } 기타							
A 줄 끝에 빛깔 이미지 a 엿볼이기	S 줄 삭제후 편집모드 S 단어 삭제후 편집모드	D 줄 끝까지 삭제 d 삭제	F 뒤로 검색 f 한 문자 찾기	G 파일끝/줄로 이동 g 확장 명령	H 화면 상단 h ← 화면 하단	J 줄 합치기 j ↓	K 도움말 k ↑	L 화면 하단 l →	: ex 명령줄 : t/T/F ; 명령 반복	" 레지스터 지정 ' 마크로 이동	열 이동 / 사용 안함							
Z 종료 Z 확장	X 백스페이스 X 삭제	C 줄 끝까지 바꾸기 C 바꾸기	V 줄단위 비주얼모드 V 비주얼 모드	B 이전 WORD b 이전 단어	N 이전 (찾기) n 다음 단어	M 화면 가운데 m 설정	< 내어쓰기 t/T/F ; 역순 검색	> 들어쓰기 .	? 찾기 (뒤로)									

**동작** 커서를 이동하거나, 연산자를 동작할 범위를 지정합니다.

**명령** 바로 동작하는 명령, **빨간색**은 편집 모드로 변경됩니다.

**연산자** 이동 관련 문자(숫자나 커서 이동)와 함께 사용하여야 하며, 커서의 위치부터 목적지까지 연산합니다.

**확장** 특별한 기 함수로, 주 가격인 키 입력이 필요합니다.

입력 후 (숫자를 제외한, 으로 끝날 수 있는) 글자를 입력하여 합니다.

**words:** 구분자로 공백, 특수기호 모두 사용  
**WORDs:** 구분자로 공백 문자만 사용

**words:** `quux(foo, baz, baz);`  
**WORDs:** `quux (foo, bar, baz);`

**주요 명령행 명령 ('ex'):**  
**:w** (저장), **:q** (종료), **:q!** (저장하지 않고 종료)  
**:e f** (파일 f 열기),  
**:s/x/y/g** (파일 전체에서 'x'를 'y'로 교체),  
**:h (vim) 도움말**, **:new (새 파일)**

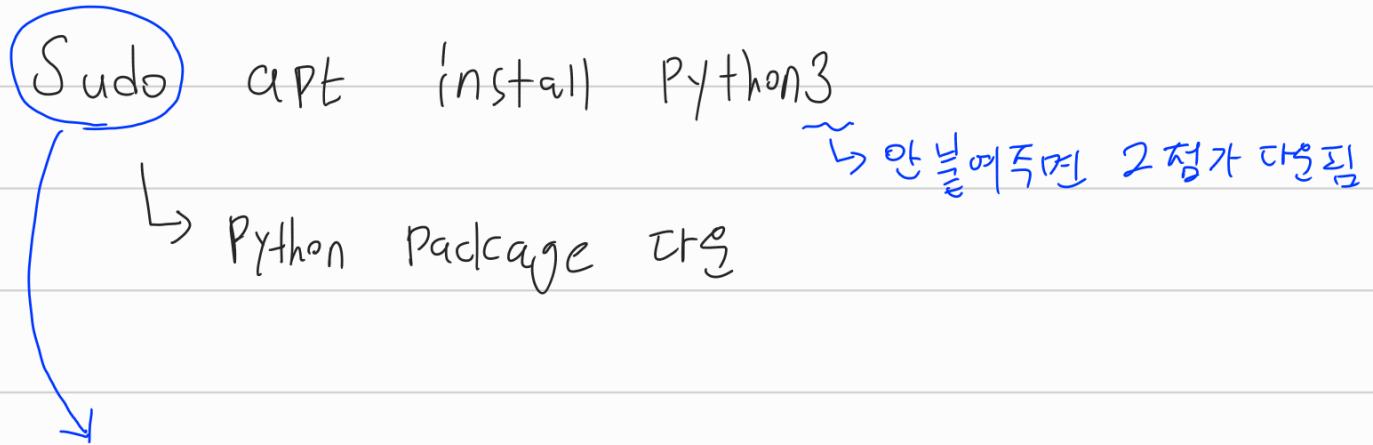
**그외 중요한 명령들:**  
**CTRL-R:** 재실행 (vim),  
**CTRL-F/-B:** 페이지 위로/아래로,  
**CTRL-E/-Y:** 줄 스크롤 위로/아래로,  
**CTRL-V:** 복사-비주얼 모드 (vim 전용)

**비주얼 모드:** 커서를 움직여 지정한 범위에 연산자를 적용합니다. (vim 전용)

**참고:**

- (1) 복사/붙이날기/지우기 명령어를 사용하기 전에 "x"를 입력하여 레지스터(클립보드)를 저장하세요. (x는 a에서 z 또는 \*을 사용할 수 있음) (예: "ay\$"를 입력하면 현재 커서에서 라인 끝까지의 내용을 레지스터 'a'에 저장합니다.)
- (2) 어떤 명령을 입력하기 전에 횟수를 지정하면, 횟수만큼 반복하게 됩니다.(예: 2p, d2w, 5i, d4j)
- (3) 연속으로 입력하는 명령은 현재의 라인에 반영됩니다. 예시: dd(현재 라인 지우기), >>(들어쓰기)
- (4) ZZ는 저장후 종료, ZQ는 저장하지 않고 종료.
- (5) zt : 커서가 위치한 곳을 제일위로 옮리기,  
zb : 바닥으로, zz : 가운데로
- (6) gg : 파일의 처음으로(Vim 전용),  
gf : 커서가 위치한 곳의 파일 열기(Vim 전용)

vi/vim에 대한 더 많은 강좌나 팁을 얻으려면 [www.viemu.com](http://www.viemu.com) (ViEmu, MS 비주얼 스튜디오를 위한 vi/vim 에뮬레이션)을 방문하십시오.



## sudo (superuser do) 명령어

현재 계정에서 **root** 권한을 이용하여 명령어를 실행할 때 사용

**sudo apt-get update**

sudo 다음에 실행할 명령을 입력하면 root 권한으로 명령어를 실행한다.

실행하기 전 현재 사용자의 비밀번호를 물어본다.

(root 사용자의 암호를 물어보는 su 명령어와 차이가 있다.)

apt-get(Advanced Packaging Tool)은 우분투(Ubuntu)를 포함한 데비안(Debian) 계열의 리눅스에서 쓰이는 팩키지 관리 명령어 도구입니다. 우분투에는 GUI로 되어 있는 시냅틱 꾸러미 관리자도 있지만 이런 저런 개발관련 패키지를 설치할 때는 커맨드기반인 apt-get이 더 편하기도 합니다. sudo는 superuser 권한으로 실행하기 위함입니다.

#!  
 ↳ 스크립트 해더