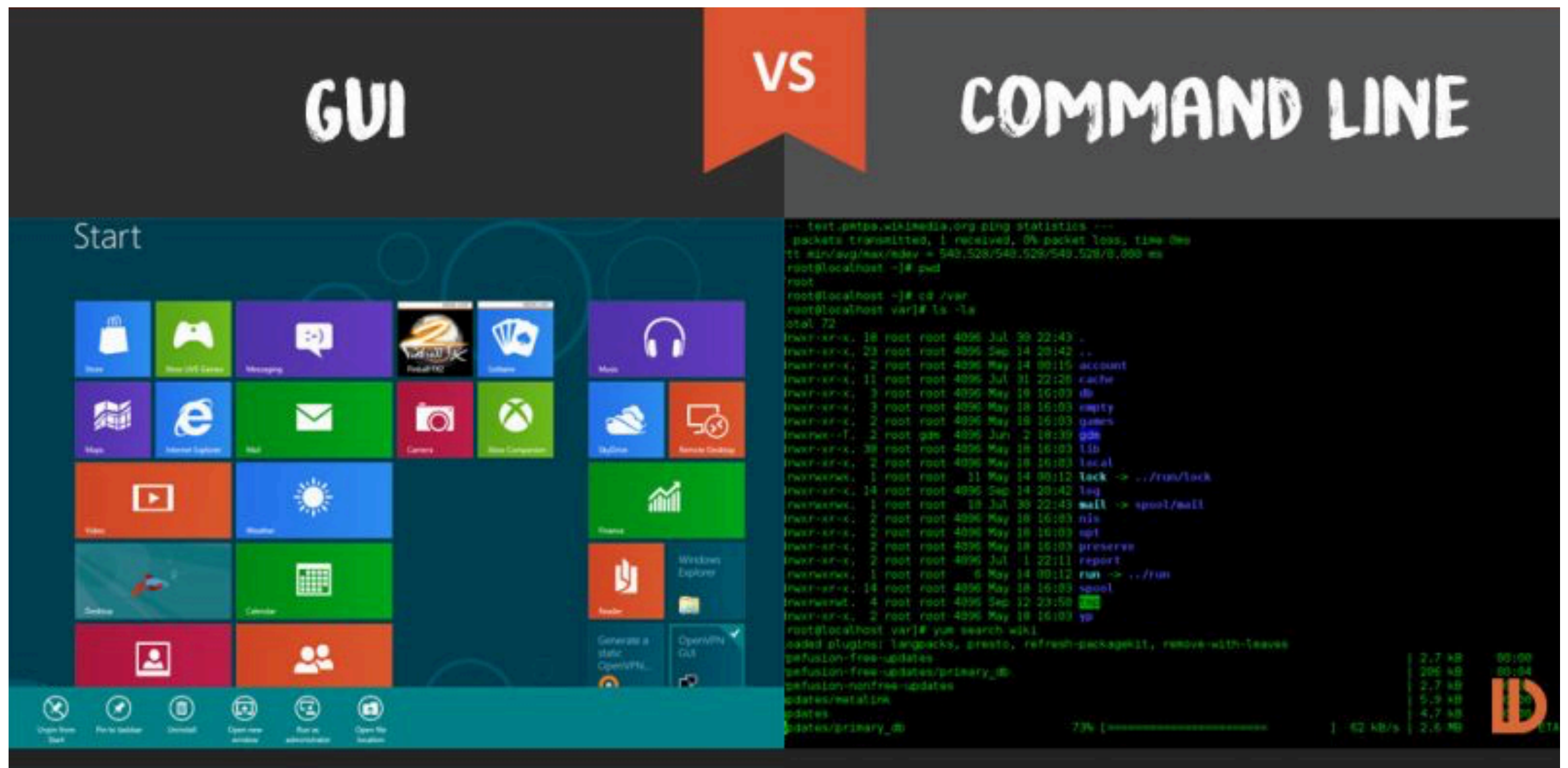


# Terminal

# GUI vs CLI

GUI : Graphic User Interface (macOS, Windows)

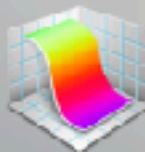
CLI : Command Line Interface (MS-DOS, Linux, Terminal)



## 기타



텍스트 편집기



Grapher



DVD 플레이어



Time Machine



서체 관리자



체스



스티커



이미지 캡처



VoiceOver 유틸리티



AirPort 유틸리티



이그레이션 자원



활성 상태 보기



론습



키체인 접근



시스템 정보



Automator



스크립트 편집기



Boot Camp 자원



디지털 컬러 측정기



ColorSync 유틸리티



화면 캡처



Bluetooth 파일 교환



오디오 MIDI 설정



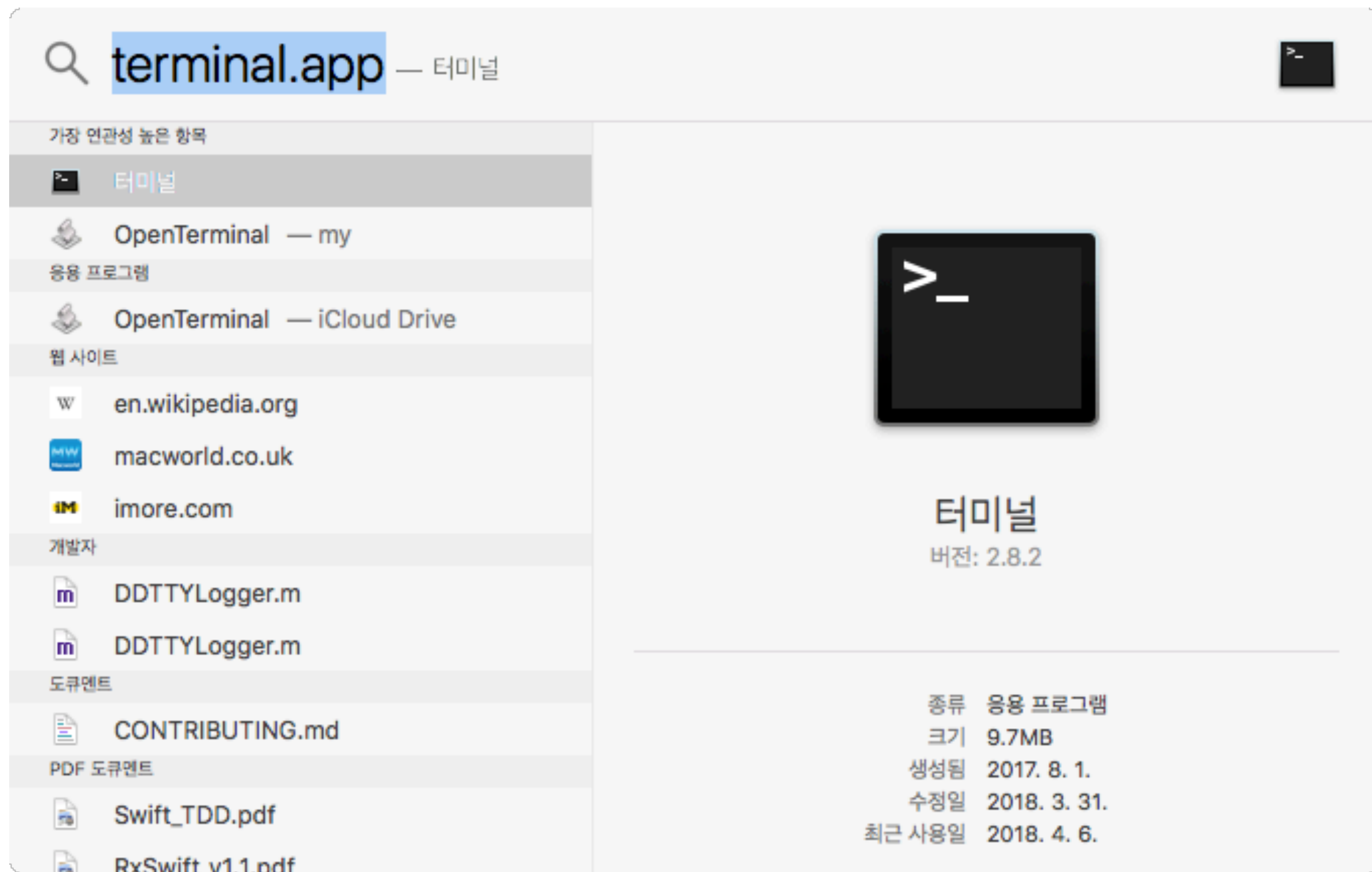
디스크 유틸리티



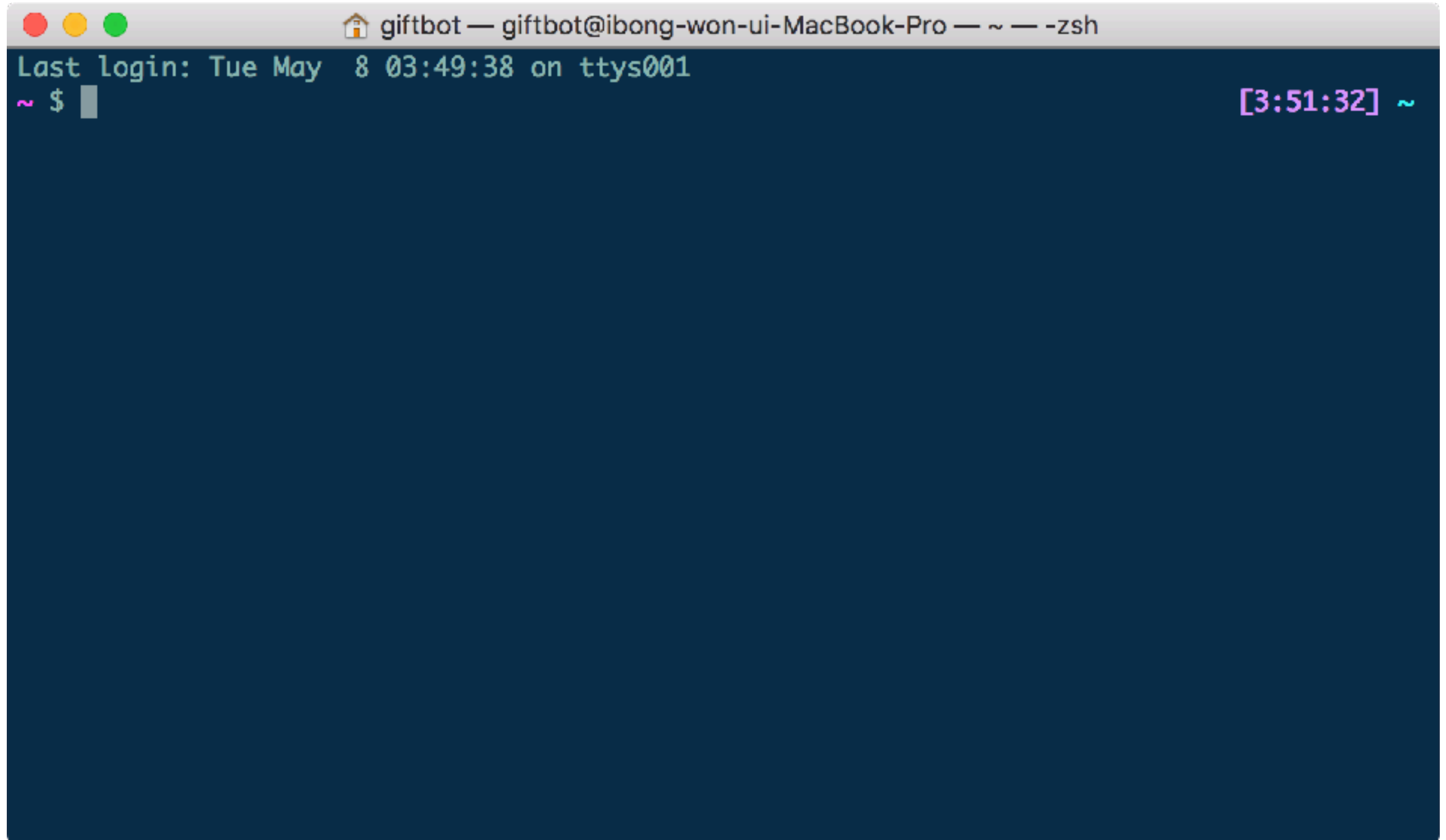
XQuartz



터미널



# Terminal



```
giftbot — giftbot@ibong-won-ui-MacBook-Pro — ~ — -zsh
Last login: Tue May  8 03:49:38 on ttys001
~ $ [3:51:32] ~
```

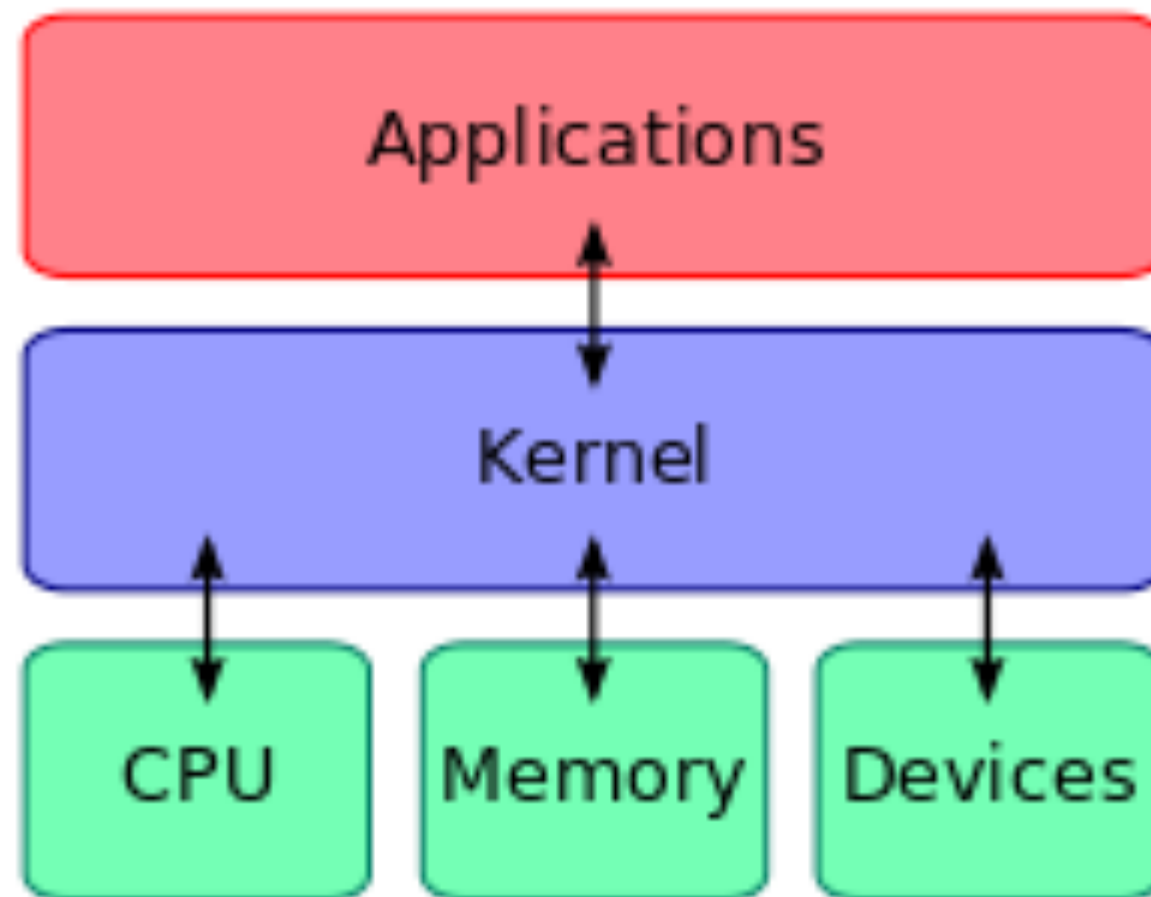
A screenshot of a macOS Terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left, followed by the text "giftbot — giftbot@ibong-won-ui-MacBook-Pro — ~ — -zsh". The main content area has a dark blue background. The first line of text is "Last login: Tue May 8 03:49:38 on ttys001" in a light green monospace font. The second line shows a prompt "~ \$" in light blue, followed by a grey cursor bar. In the top right corner of the terminal area, the text "[3:51:32] ~" is displayed in light blue.

echo \$SHELL - 현재 사용 중인 셸 출력  
cat /etc/shells - 사용 가능한 셸 목록 출력

```
[~] $ cat /etc/shells
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/bash
/bin/csh
/bin/ksh
/bin/sh
/bin/tcsh
/bin/zsh
```

운영 체제의 핵심 역할 : 하드웨어를 관리하고 필요한 프로세스에 나눠주는 등 시스템 여러 시스템 자원을 제어  
컴퓨터 부팅시 부트로더에 의해 로드되어 항상 메모리에 상주  
프로세서, 프로세스, 하드웨어, 메모리, 파일 시스템, 네트워크 등을 관리  
인터럽트 처리기, 스케줄링 (처리 순서 결정), 수퍼바이저 (사용권 부여) 등이 포함



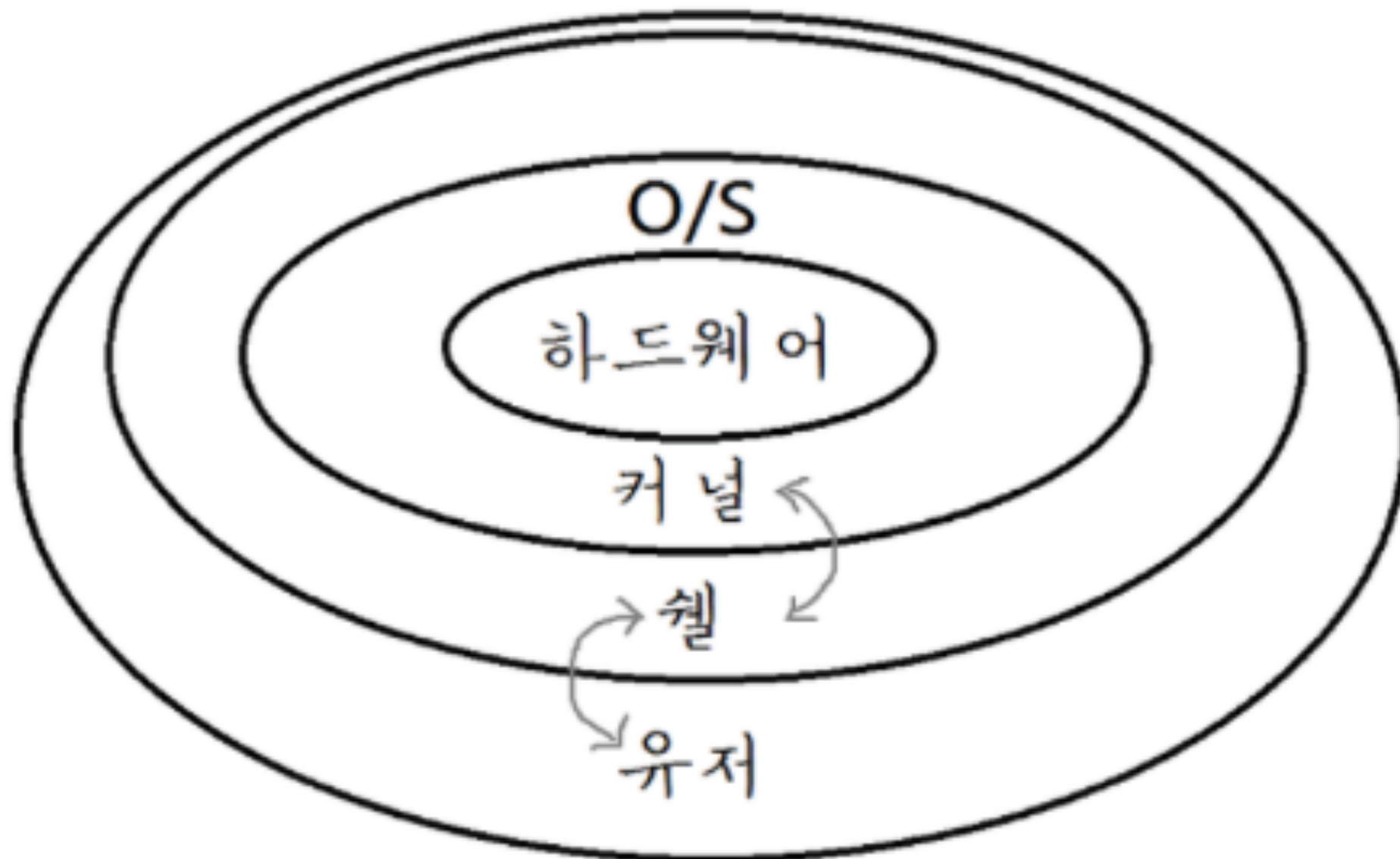


초기 O/S에서는 커널이 존재하지 않기도 함

종류	커널 기반의 운영체제 Linux, Unix, 윈도우 NT	커널이 없는 운영체제 MS-DOS
형태	하드웨어와 응용프로그램 사이에 커널이 중간에 한 번 개입을 함	응용 프로그램이 직접 하드웨어에 접근함
응답시간	응용 프로그램 입장에서 볼 때 응답 시간은 약간 지연	응용 프로그램의 응답 시간은 빨라짐
통제	잘못된 하드웨어에의 접근을 거부할 수 있고 하드웨어 접근 창구가 일원화 되기 때문에 시스템의 안정성 향상을 기대할 수 있음	응용 프로그램이 하드웨어에 대한 잘못된 조작을 하더라도 이것을 저지할 방법이 없게 됨



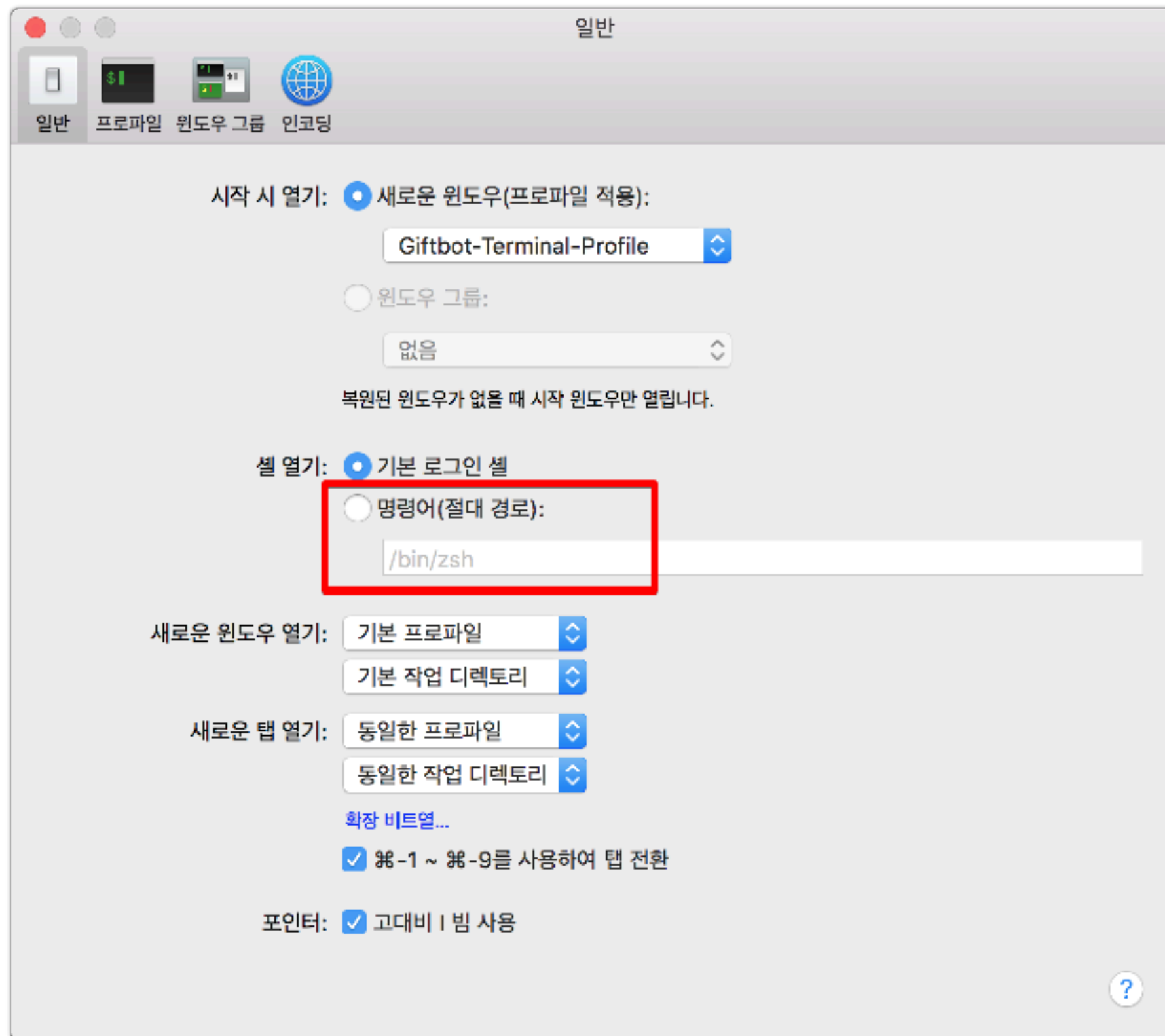
Shell : 사용자와 운영체제 (커널) 간에 대화를 가능하게 해주는 명령어 해석기 역할. 대화식 인터페이스 구조  
사용자 (명령) -> 셸 (해석) -> 커널 (명령 수행 후 결과 전송) -> 셸 (해석) -> 사용자 (결과 확인)



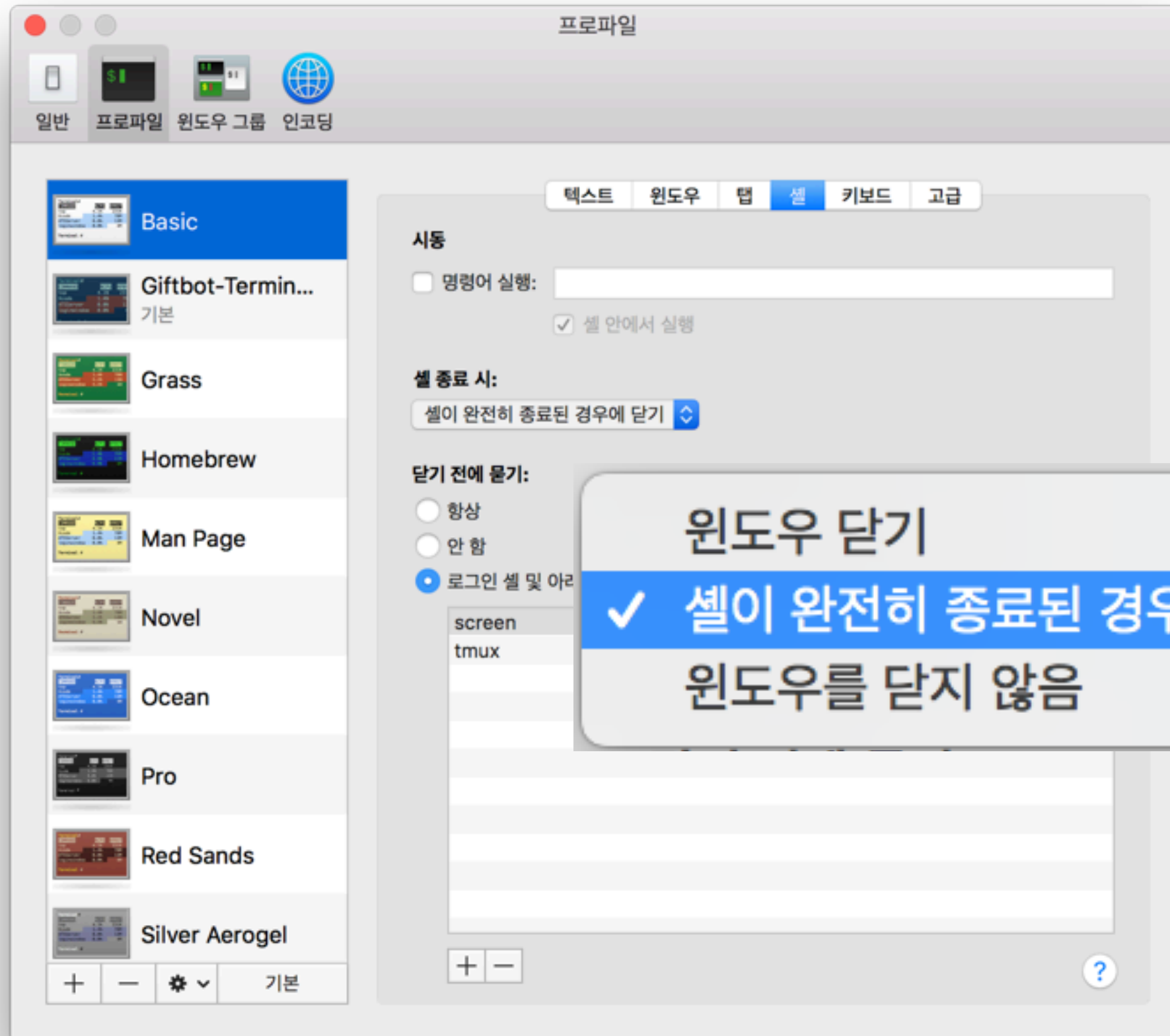
# Shell

셸 이름	실행 명령	설명
Bourne Shell	sh	벨 연구소의 스티브본 (Stephen Bourne) 개발, 많은 셸 스크립트의 기반이 되는 셸
C Shell	csch	C언어 구문과 유사, Bourne Shell 을 확장하여 히스토리, 작업제어, 엘리어스 등 기능 추가 개발자들에게 유용한 기능들을 제공한다.
TC Schell	tcsh	C Shell 에 명령 행 완성 과 명령 행 편집 기능을 추가
Korn Shell	ksh	Bourne Shell 가 호환되며 C Shell 의 많은 기능을 포함, Unix 계열에서 많이 사용된다.
Bourne Again Shell (bash)	bash	리눅스에서 가장 많이 사용되는 셸로 Bourne 셸을 토대로 C셸과 Korn Shell 의 기능들을 통합시켜 개발되었다.

# Base Shell



# Preference





[Home](#) [News](#) [Features](#) [FAQ](#) [Documentation](#) [Downloads](#) [Donate](#)

## Stable Releases

Stable releases update rarely but have no serious bugs.

[Download](#) **iTerm2 3.1.6 (OS 10.10+)**

This is the recommended build for most users.

- ▶ [Show Changelog](#)
- ▶ [Show Older Versions](#)

- 목록 조회 (ls), 경로 이동 (cd), 경로 확인(pwd)
- 디렉토리 생성 (mkdir), 디렉토리 삭제 (rmdir)
- 파일 생성 (touch), 복사 (cp), 이동 (mv), 삭제 (rm),
- 에디터 실행 (vi / nano 등등), 파일 내용 출력 (cat), 파일 상단/하단 내용 출력 (head / tail)
- 명령어 매뉴얼 확인 (man), 입력한 명령어 목록 출력 (history), 다중명령어 실행 (;)
- 지정한 경로를 Finder 로 열거나 파일 실행 (open)
- 파일 속성 변경 (chmod)
- 화면 클리어 (clear), 종료 (exit)

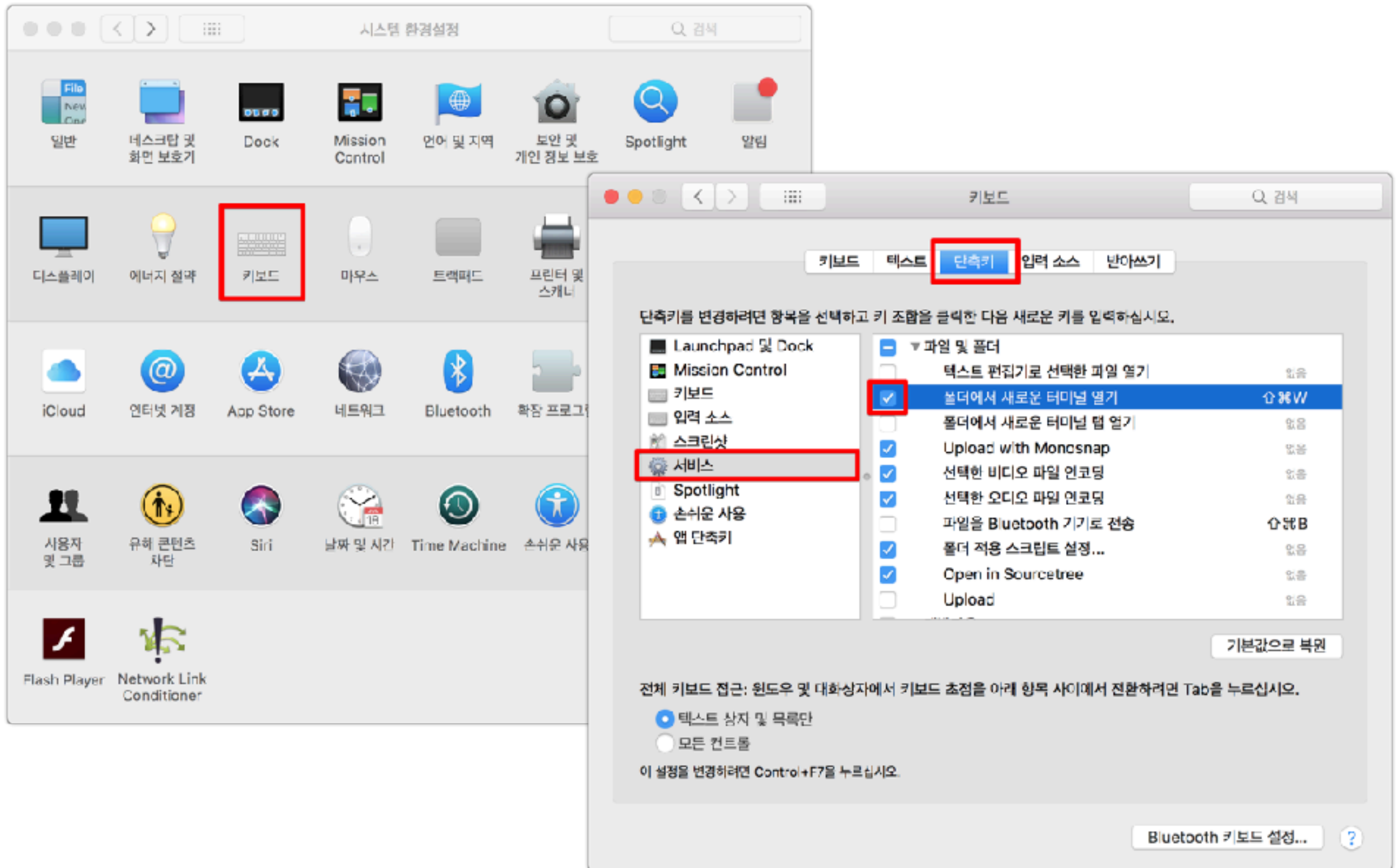
# Change Directory

---

1. cd 명령어, e.g. `$ cd ~/Desktop`
2. Drag & Drop
3. 서비스 단축키 등록



# Change Directory



# Homebrew

macOS 용 패키지 매니저 ( <https://brew.sh> )



# Homebrew

The missing package manager for macOS

English

## Install Homebrew

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

# Check Ruby Version

---

```
$ brew upgrade ruby
```

```
$ ruby -v
```

$V_i(m)$

**VI (Visual Editor, 1976) / Vim (Vi IMproved, 1991)**

- 오픈 소스 텍스트 에디터 (라인 기반)
- ex 라는 기존 에디터의 단점을 보완하기 위해 개발
- 현재 대부분의 리눅스 기반 운영체제에서 vi 를 쳐도 alias 되어 있어 vim 으로 실행
- 초기 러닝 커브 / 숙달 후 빠른 속도로 편집 가능

**주요 에디터 : Emacs, Nano, Visual Studio Code, Sublime Text, Atom 등**



Esc

명령 모드

## Vim 명령어 단축키

손에 잡히는 Vim 입문서

~ 대소문자 전환 , 포식으로 이동	! 외부 명령 1	@ 매크로 실행 2	# 이전 검색 3	\$ 행 끝으로 이동 4	% 짝 끝 찾기 5	^ 행의 첫 글자 6	& :g 반복 7	* 다음 검색 8	( 문장 시작 9	) 앞으로 문장 끝	- 아래 행 이전 행	+ 다음 행 = 자동 들여쓰기
Q 실행 모드 q 매크로 기록	W 다음 단어 (의미상) w 다음 단어	E 단어 (의미상) 끝으로 e 단어 끝으로	R 수정 모드 r 한 문자 교체	T 앞에서 후방 검색 t 행에서 전방 검색	Y 행 단위 복사 y 복사	U 행 단위 실행 취소 u 실행 취소	I 행 시작에 삽입 i 커서 앞에 삽입	O 행 위에 삽입 o 행 아래에 삽입	P 커서 이전에 붙여넣기 p 커서 이후에 붙여넣기	{ 문단 시작 [	} 문단 끝 ]	
A 행 끝에 삽입 a 커서 뒤에 삽입	S 삭제 후 입력 모드 s 단어 삭제 후 입력 모드	D 행 삭제 d 삭제	F 행에서 후방 검색 f 행에서 전방 찾기	G 파일 끝으로 이동 g 확장 명령	H 화면 상단 h ←	J 아래 행 합치기 j ↓	K 다음 줄 k ↑	L 화면 하단 l →	: 명령행 모드 ; /T/t/F 빈번	레지스터 지정 ' . 포식으로 이동	열 이동 \ 사용하지 않음	
Z 종료 z 확장 명령	X 백스페이스 x 글자 삭제	C 한 줄 복사/바꾸기 c 바꾸기	V 비주얼 라인 모드 v 비주얼 모드	B 이전 단어 (의미상) b 이전 단어	N 이전 (찾기) n 다음 (찾기)	M 화면 가운데 m 표시 설정	< 내어 쓰기 , /T/t/F 반대 방향 검색	> 들여 쓰기 . 명령 반복	? 찾기 (위쪽) / 찾기 (아래쪽)			

## 동작

커서를 이동하거나,  
연산자가 동작할 범위를 지정합니다.

## 명령

바로 동작하는 명령입니다.  
빨간색은 입력 모드로 변경됩니다.

오래래이션  
펜딩 모드

커서 위치부터 목적지까지를 대상으로  
명령을 실행합니다.

## 확장

추가적인 키 입력이 필요합니다.

q.

입력 후 글자를 입력해야 합니다.

단어

공백 문자나 특수 문자로 구분된 단어  
test(123, 456, 789)

단어  
(의미상)

공백 문자로 구분된 단어  
test(123 456 789):

## • 명령행 모드의 주요 명령어

:w 저장  
:q 종료  
:q! 저장없이 종료  
:e f 파일 열기  
:%a/x/y/g 파일 전체에서 'x'를 'y'로 교체  
:h name name 명령에 대한 도움말  
:new 새 파일

## • 일반 모드의 주요 명령어

CTRL-R 재실행  
CTRL-F/-B 페이지 위로/아래로  
CTRL-E/-Y 스크롤 위로/아래로  
CTRL-V 비주얼 모드

## • 참고

- 1 복사/붙여넣기/지우기 명령을 사용하기 전에 "a"를 입력하여 레지스터 a를 지정할 수 있습니다. (레지스터 0부터 a부터 z까지 사용 가능)  
예를 들어 "ay\$는 커서 위치부터 행 끝까지의 내용을 레지스터 a에 저장합니다.
- 2 명령어 입력 전 숫자를 지정하면, 해당 숫자만큼 명령어가 반복됩니다.
- 3 연속으로 입력하면, 현재 행이 반영됩니다.  
예를 들어 dd는 현재 행이 지워집니다.
- 4 ZZ는 저장 후 종료, ZQ는 저장 없이 종료입니다.
- 5 zt는 커서가 위치한 부분을 화면 상단으로 스크롤합니다.  
zb는 바닥으로, zz는 가운데로 스크롤합니다.
- 6 gg는 커서를 파일 처음으로 이동합니다.  
g'는 커서 위치의 파일명을 인식하여 파일을 엽니다.



**I Am Devloper**

@iamdevloper



Following

I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it.

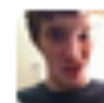
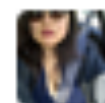
[↩ Reply](#) [↻ Retweet](#) [★ Favorite](#) [... More](#)

RETWEETS

4,846

FAVORITES

2,105



4:56 AM · 18 Feb 2014



# Vim vs Emacs

## VIM

usable in just about  
any environment.

does one thing, well.



## EMACS

flexible, customizable, and  
packed with every feature  
known to man.

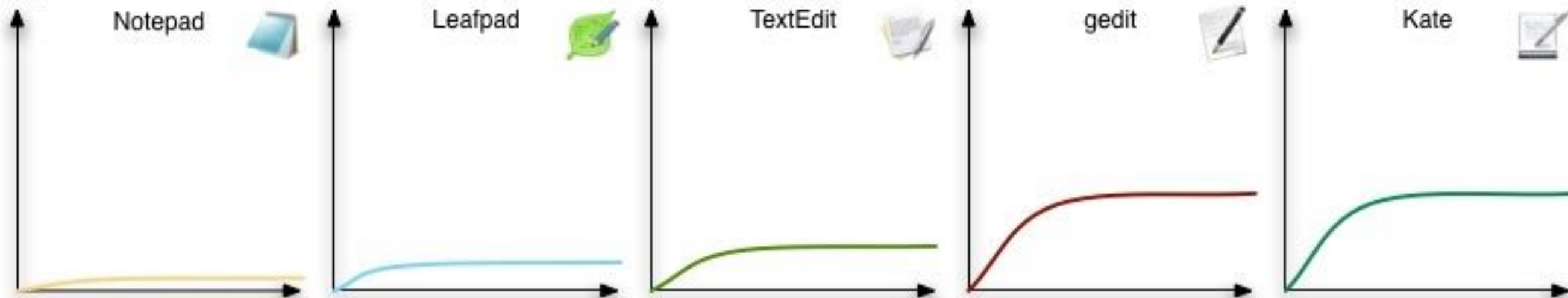


## NANO

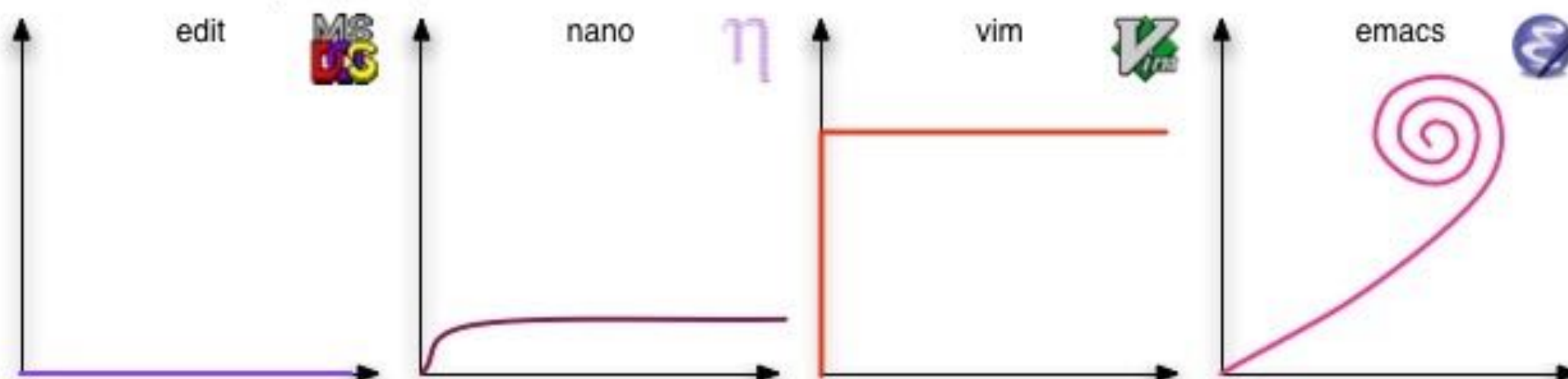
mostly used by people  
who do not know  
what they are doing;  
or psychopaths.



### System default editors



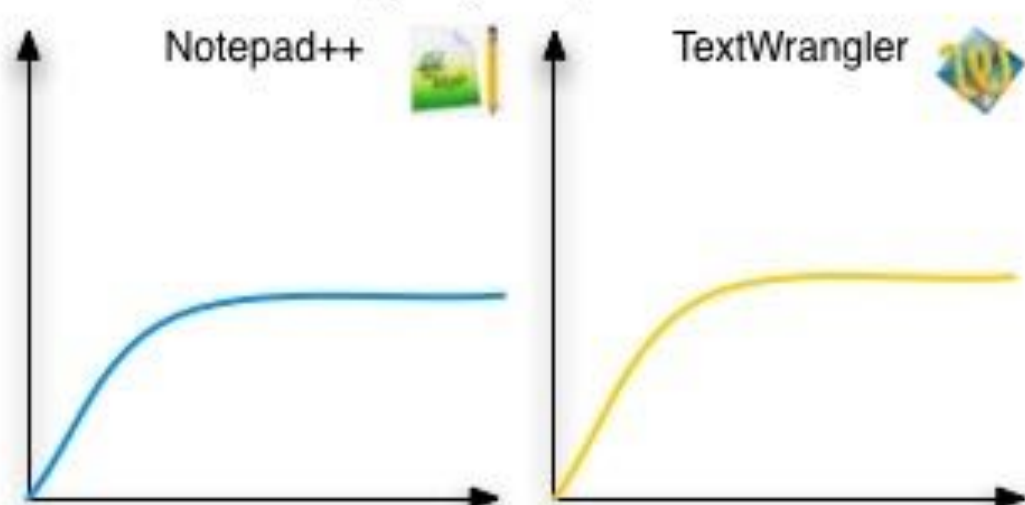
### Terminal editors (nerdy stuff)



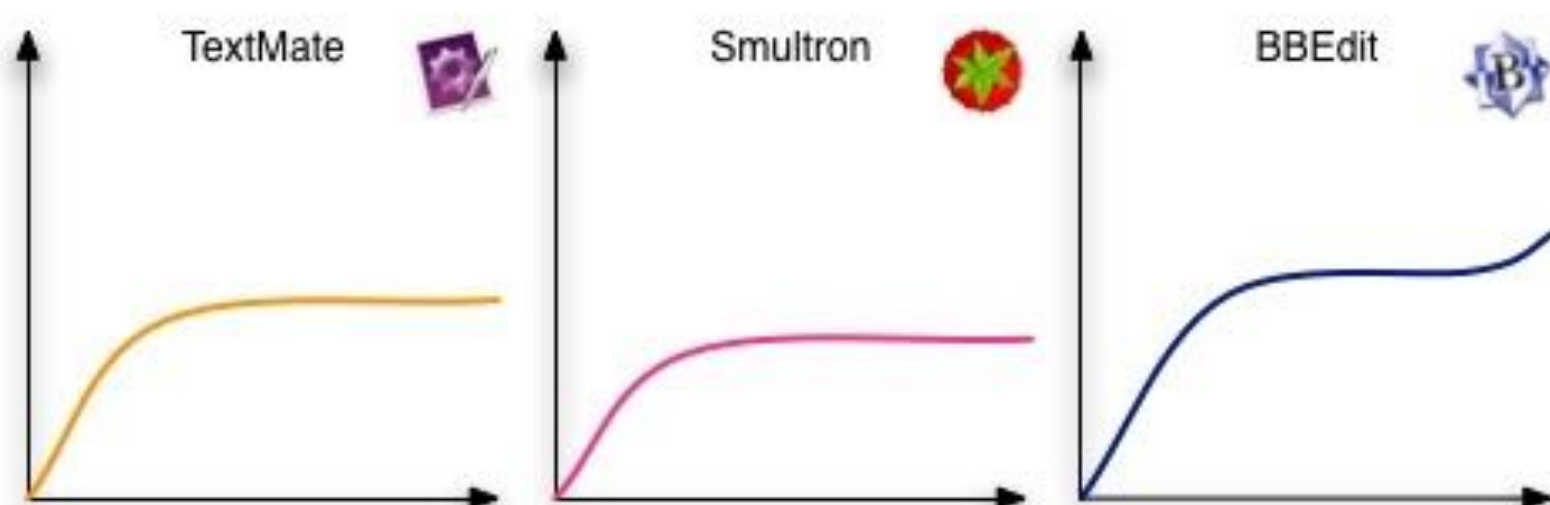
Typical learning curves for some common text editors

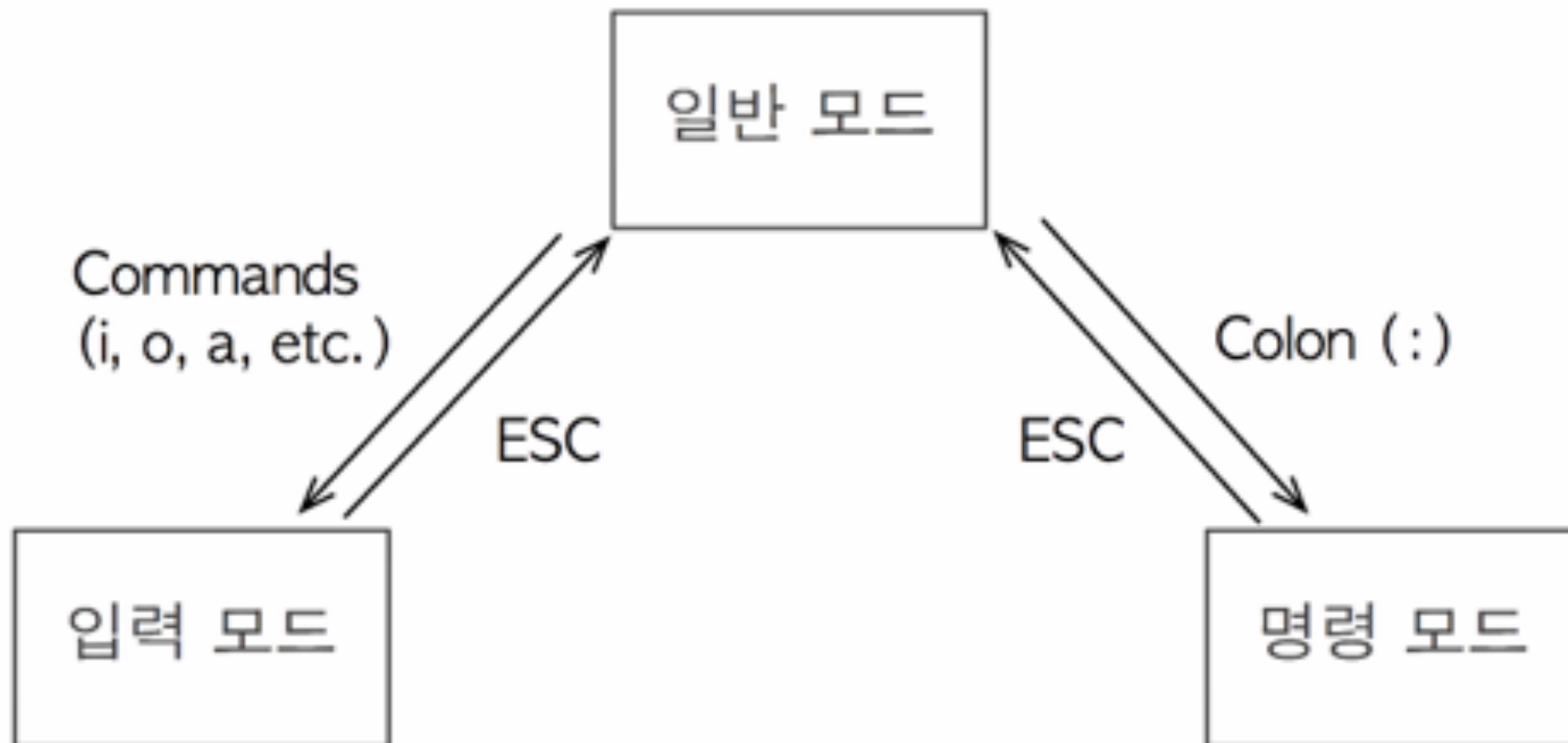
[manuelmagic.me/geek/texteditors](http://manuelmagic.me/geek/texteditors)

### Freeware editors (geeky stuff)

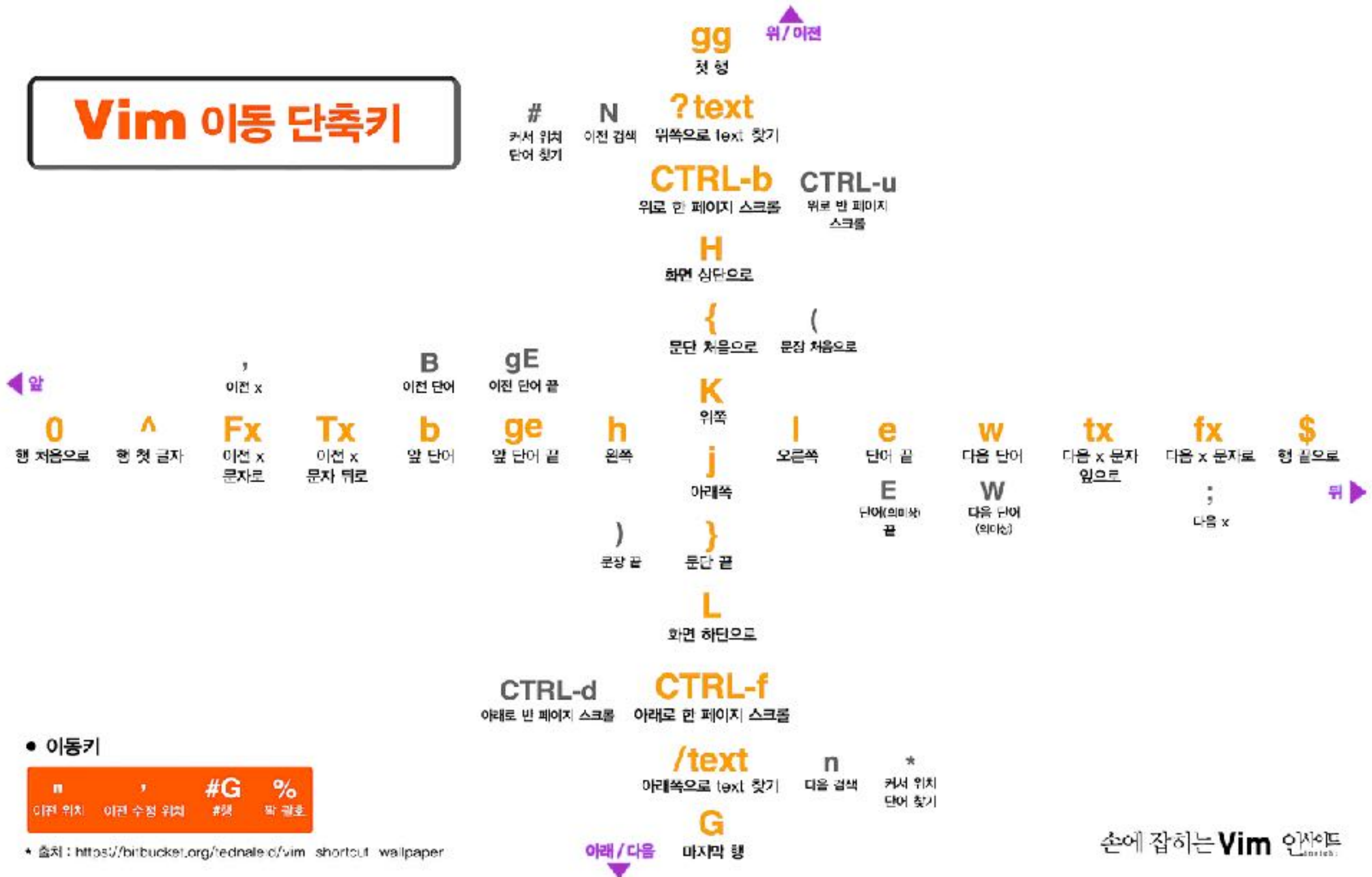


### Shareware editors





# Vim 이동 단축키



## • 이동키

<b>''</b>	<b>,</b>	<b>#G</b>	<b>%</b>
이전 위치	이전 수정 위치	#행	짝 괄호

★ 출처 : <https://bitbucket.org/hednale/d/vim-shortcut-wallpaper>

손에 잡히는 Vim 인사이트