Vim Editor

ABC 멘토링 5조
멘토 정현준
2021 / 01 / 18

ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY

공지 사항

- **2/21까지 모든 주제를 끝내야 한다고 합니다...**

차시	주제	차시	주제
1 (12/31)	Course Overview (TODAY)	7 (1/28)	Command-line Environment
2 (1/4)	Git	8 (2/2)	Debugging and Profiling
3 (1/7)	Shell 1 (기본 명령어)	9 (2/4)	Metaprogramming
4 (1/14)	Shell 2 (각종 툴 및 스크립트)	10 (2/8)	Security and Cryptography
5 (1/18)	Text editor(Vim)	11 (2/15)	Potpourri
6 (1/21)	Data Wrangling	12 (2/18)	Special Topic and Q&A

Vim

- Vi Improved의 약자 (라고 합니다.)
- 1976년에 만들어진 Vi editor에서 출발했습니다.
- 현재까지도 Linux OS에서 많이 사용됩니다.
- Vimscript를 이용해 다양한 커스터마이징이 가능합니다.
- 가끔 오래된 시스템의 경우 vim이 없는 경우가 있습니다.
  - 그럴 땐 “sudo apt-get install vim” 명령어를 통해 설치 해줍시다.
  - Uni 서버의 경우엔 vim이 이미 설치되어 있습니다.

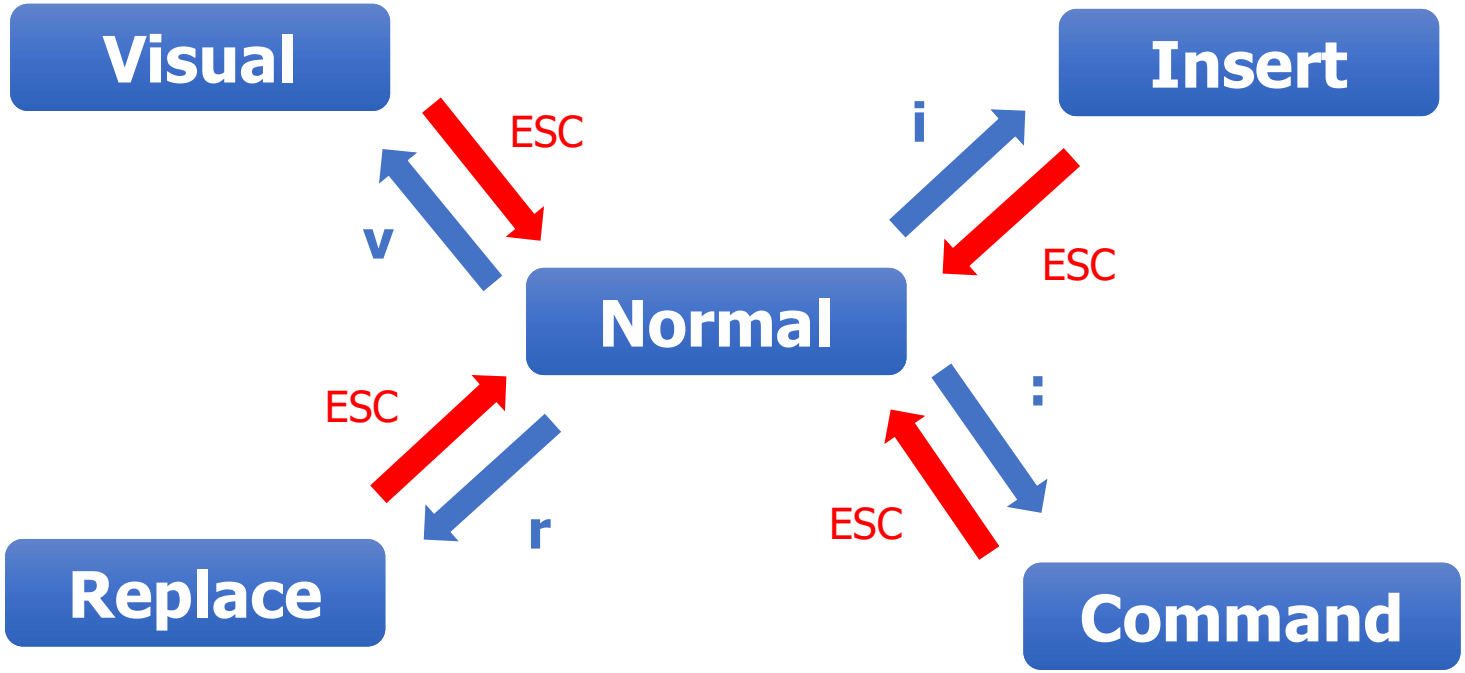
Vim modal editing

- Vim은 text 편집 외에 다양한 모드를 제공합니다.

Normal	파일 내부에서 커서를 움직이거나 간단한 편집을 할 때
Insert	Text를 추가할 때
Replace	Text를 바꿀 때
Visual	Text를 블록 단위로 선택할 때
Command-line	Vim command를 사용할 때

Vim mode 바꾸기

- 다른 모드에서 ESC를 누르면 항상 normal 모드가 됩니다.



Vim buffer

- Vim은 파일을 "buffer"라는 단위로 관리합니다.
- Vim session은 "tab"이라 부르는 창에서 각 파일을 하나씩 보여줍니다.
- 즉, 동일한 tab은 여러 개의 buffer를 열 수 있습니다.

Buffer = 메모리 상에 올라온 text file

Window = buffer의 출력 영역 (view point)

Tab = window의 집합

Vim command-line

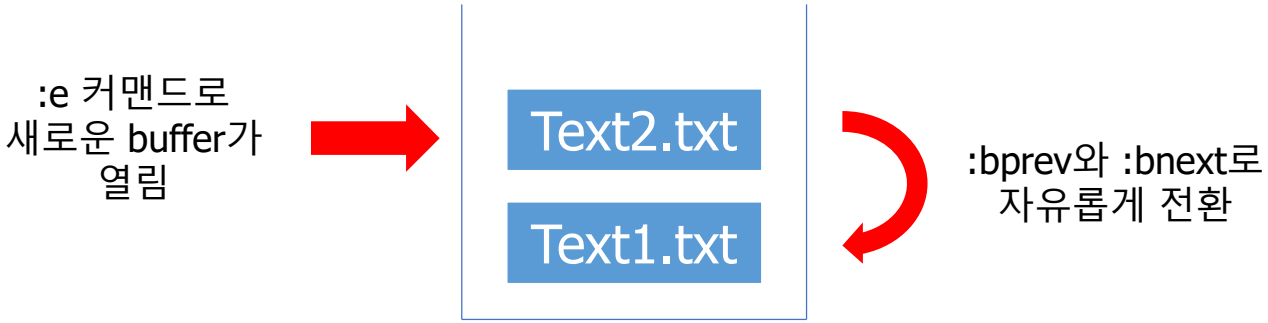
- :q => vim을 종료합니다.
- :w => 수정한 내역을 저장합니다.
- :wq => 저장 후 종료
- :ls => 현재 열려 있는 buffer들을 보여줍니다.
- :set [setting command] => setting command를 적용합니다.
  - :set number => 파일에서 라인 넘버를 보여줍니다.
  - :set ts=4 => tab space를 4칸으로 조정합니다.

Vim command-line

- :help [command] => 커맨드에 대한 정보를 출력
  - Ex) :help :w => 저장 커맨드에 대한 정보 출력
- :숫자 => 해당 숫자에 맞는 라인으로 이동
  - Ex) :10 => 파일의 10번째 줄로 이동
- :/"문자열" (또는 정규표현식)=> 해당 문자열을 파일에서 검색 (ctrl+F 역할)
- :nohl => highlight 표시 제거
- :%s/문자열1/문자열2/g => 파일 안에 문자열 1 패턴을 문자열 2로 모두 바꿉니다.

Vim buffer 이동

- :e [파일 이름] => 새로운 파일을 buffer로 엽니다.
- :ls => 현재 열려 있는 모든 buffer를 출력합니다.
- :bnext => 다음 buffer로 이동합니다.
- :bprev => 이전 buffer로 이동합니다.



Vim 화면 나누기

- :split (또는 :sp) => 화면을 세로로 나눕니다.
- :vsplit (또는 :vs)=> 화면을 가로로 나눕니다.
- :vs(sp) 파일이름 => 특정 파일을 열어서 창 분할을 합니다.
- :q => 현재 커서 위치의 창을 닫습니다.
- Ctrl + w, w => 다음 창으로 커서 이동
- Ctrl + w, shift + w => 이전 창으로 커서 이동
- Ctrl + w, = => 모든 창의 크기를 균일하게 맞춘다.

Vim Macro

- Normal mode 또는 visual mode에서 사용합니다.
- gg (키보드 g 두 번 입력) : 파일의 첫 줄로 이동합니다.
- Shift + g : 파일의 마지막 줄로 이동합니다.
- dd : 커서가 위치한 라인을 삭제합니다.
- u (Undo) : 이전에 실행한 편집을 되돌립니다. (ctrl+z 역할)
- Ctrl+r (Redo) : undo 했던 내용을 되돌립니다.
- y (yank) : 선택된 내용을 복사합니다.
- p (past) : yank로 복사한 내용을 붙여 넣습니다.

Vim Macro

- Normal mode 또는 visual mode에서 사용합니다.
- n : 문자열 찾기 상태에서 다음으로 찾은 문자열로 넘어 갑니다.
- Shift + n : 문자열 찾기 상태에서 이전에 찾은 문자열로 넘어 갑니다.
- Ctrl + e : 아래로 한 줄 내려 갑니다.
- Ctrl + y : 위로 한 줄 올라 갑니다.

그 외 자잘한 팁

- Command 모드를 제외한 모든 모드에서 사용 가능합니다.
- Home : 줄의 처음 위치로 커서를 옮깁니다.
- End : 줄의 마지막 위치로 커서를 옮깁니다.
- Page up : 이전 화면의 코드를 보여줍니다.
- Page down : 다음 화면의 코드를 보여줍니다.
- Insert : insert 모드와 replace 모드를 서로 전환합니다.
- Delete : 커서가 위치한 자리의 글자 하나를 삭제합니다.

.vimrc

- Vim 내부에서 :set 명령어를 이용하면 vim을 커스터마이징 할 수 있습니다.
- 하지만 set 명령어를 통해 실행한 커스터마이징은 vim을 종료하는 순간 설정이 저장되지 않습니다.
- 그럴 땐 자신의 리눅스 계정 홈 디렉토리(~ 디렉토리)에 .vimrc 파일을 만들고 그 안에 set 명령어를 저장하여 영구적으로 vim에 자신만의 설정을 영구적으로 저장할 수 있습니다.
- Vim에는 set 명령어를 통해 다양한 커스터마이징이 가능합니다.
- http://vimdoc.sourceforge.net/htmldoc/options.html (set 관련 참고 문서)

Vim script

- Vim script는 vim 명령들을 자동으로 처리 하도록 작성한 vim용 명령어 입니다.
- 변수, 함수, 조건문 등의 기능도 있기 때문에 vim script language라고 합니다. (여기서는 시간상 다루지 않겠습니다.)
- :로 시작하는 command-line 명령어 역시 vim scrip를 통해 사용됩니다.
- 또한 .vimrc 파일도 vim script에 포함됩니다.
- 즉, 여러분들이 vim 설정을 바꾸는 것은 vim script를 통해 바꾸는 것입니다.

Vim plugin

- 앞서 언급한 vim script 중에 가장 처음 vim이 시작될 때 한 번 로드가 되는 script가 있습니다.
- 이러한 vim script를 vim plugin이라고 합니다.
- Vim plugin을 이용하면 마치 새로운 기능이 vim에 추가된 것처럼 보입니다.
- 다른 사람이 만든 vim plugin을 설치만 하면 자유롭게 사용하고 수정할 수 있습니다.
- 이러한 plugin 관리를 위해서 Vundle이라는 프로그램이 있습니다.
- https://nolboo.kim/blog/2016/09/20/vim-plugin-manager-vundle/

Assignment

- Uni07 서버에 자신만의 .vimrc 파일을 만들어봅시다.
- 만든 .vimrc를 자신의 github로 올려봅시다.

- 다음 시간에 할 Data Wrangling에 대한 강의를 보고 옵시다.
- https://missing.csail.mit.edu/2020/data-wrangling/

Thank you


UNIST
ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY
2009