# vi 편집기

# 리눅스 문서 편집기

- 리눅스 편집기의 종류
  - ✓ 유닉스에서부터 사용했던 편집기
    - 행 단위 편집기(한 번에 한 행씩 작성하거나 수정하는 편집기): ed, ex, sed
    - 화면 단위 편집기: vi(vim – 일반적으로 사용하는 화면 편집기), emacs, nano(메뉴 기반 에디터)
    - GUI 편집기: gedit

# 리눅스 문서 편집기

- ❖ 리눅스 편집기의 종류
  - ✓ 모드형과 비모드형 편집기
    - 모드형
      - ◆ 입력 모드와 명령 모드가 구분
      - ◆ 입력 모드는 텍스트를 입력할 수 있는 모드이고 명령 모드는 텍스트를 수정하거나 삭제하고 복사와 붙이기 등 편집을 하는 모드
      - ◆ 같은 글자라도 입력 모드에서는 텍스트로 처리하여 입력되고 명령 모드에서는 텍스트로 입력되는 것이 아니라 편집 명령으로 사용
      - ◆ vi는 모드형 편집기
    - 비모드형
      - ◆ 입력 모드와 명령 모드가 구분되어 있지 않음
      - ◆ 편집 기능을 CTRL 이나 ALT 같은 특수 키와 함께 사용
      - ◆ nano는 비모드형 편집기

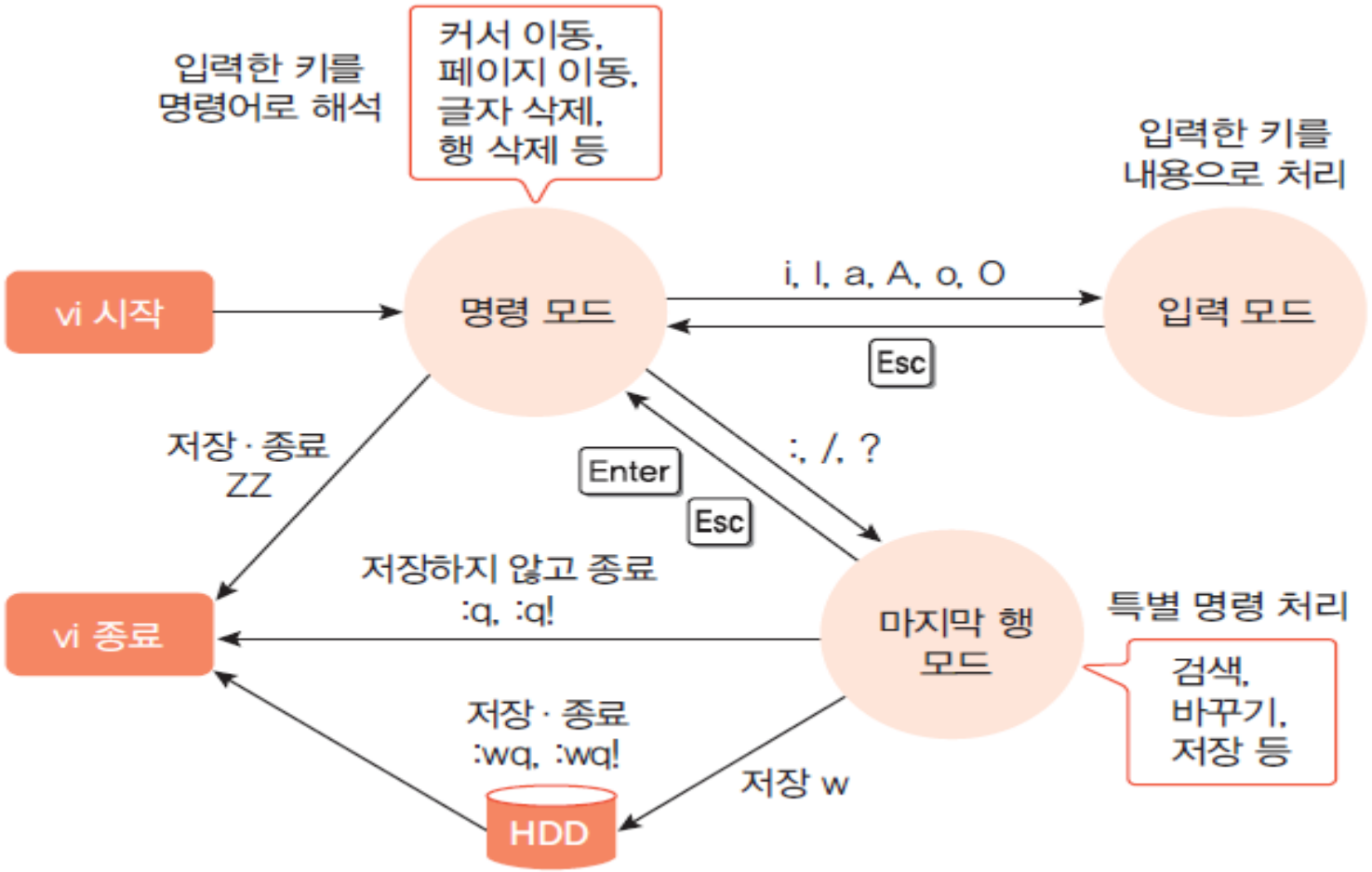
# vim

- vim
  - ✓ vi 이후에 개발된 Editor
  - ✓ 설치

    sudo apt install vim

# vim

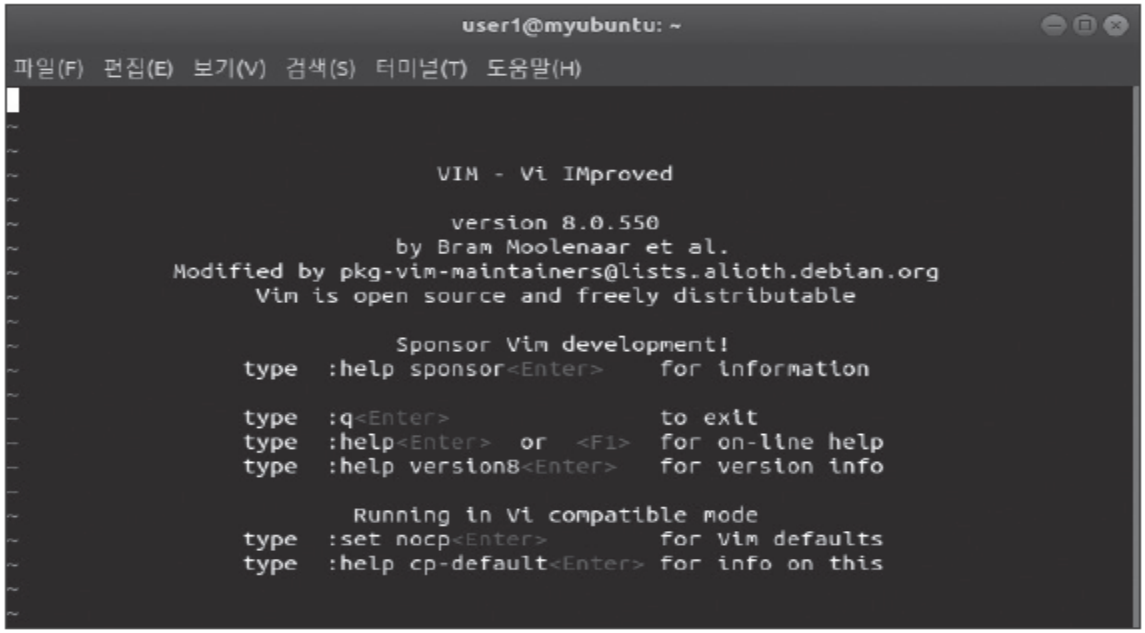
❖ 동작 모드


입력한 키를
명령어로 해석
커서 이동,
페이지 이동,
글자 삭제,
행 삭제 등
입력한 키를
내용으로 처리
vi 시작
명령 모드
i, I, a, A, o, O
입력 모드
Esc
저장 · 종료
ZZ
Enter
Esc
:, /, ?
저장하지 않고 종료
:q, :q!
vi 종료
마지막 행
모드
특별 명령 처리
검색,
바꾸기,
저장 등
저장 · 종료
:wq, :wq!
저장 w
HDD

# vim

- ❖ 시작
  - ✓ 기본 형식

    vim [파일 경로]
    - 해당 파일이 존재하면 파일의 내용이 보이고 존재하지 않은 파일이면 빈 파일이 열림
    - 파일 경로를 생략하면 아래와 같은 화면이 출력됨
    - 수정 권한이 없으면 [읽기 전용] 으로 열림



# vim

❖ 종료

✓ 명령 모드나 마지막 행 모드에서 저장하고 종료 가능

| 모드 | 명령 키 | 기능 |
|---|---|---|
| 마지막 행 모드 | :q | vi에서 작업한 것이 없을 때 그냥 종료한다. |
| | :q! | 작업한 내용을 저장하지 않고 종료한다. |
| | :w [파일명] | 작업한 내용을 저장만 한다. 파일명을 지정하면 새 파일로 저장한다. |
| | :wq, :wq! | 작업한 내용을 저장하고 vi를 종료한다. |
| 명령 모드 | ZZ(Shift+zz) | 작업한 내용을 저장하고 vi를 종료한다. |

- 마지막 행 모드에서 !는 강제 실행을 의미

# vim

❖ 모드 전환

✓ 입력 모드 전환

| 명령 키 | 기능 |
|---|---|
| i | 커서 앞에 입력한다(현재 커서 자리에 입력한다). |
| a | 커서 뒤에 입력한다(현재 커서 다음 자리에 입력한다). |
| o | 커서가 위치한 행의 다음 행에 입력한다. |
| I(대문자 i) | 커서가 위치한 행의 첫 칼럼으로 이동하여 입력한다. |
| A | 커서가 위치한 행의 마지막 칼럼으로 이동하여 입력한다. |
| O | 커서가 위치한 행의 이전 행에 입력한다. |

# vim

❖ 모드 전환

✓ vim test.txt를 실행한 뒤 명령 모드에서 i 명령 키를 입력하고 나서 다음 내용을 입력

```
ubuntu linux study      → Enter 키를 누르면 다음 행으로 이동한다.
I like linux            → Esc 키를 누르면 명령 모드로 전환된다.
~
~
~
```

✓ 입력 모드에서 다시 명령 모드로 전환하기 위해 esc 키를 누르면 커서가 x 자 위로 이동

```
ubuntu linux study
I like linux                → 명령 모드로 전환되고 커서가 x로 이동한다.
~
~
~
```

# vim

- ❖ 모드 전환
  - ✓ i 와 a 명령 키의 차이
    - 명령 키 i는 커서 앞에, a는 커서 뒤에 입력
    - 커서가 마지막 글자인 x 자 위에 있는 상태에서 i를 입력하고 Space Bar+ubuntu를 입력한 후 Esc 키를 누르면 명령 모드로 전환되어 마지막 입력 글자인 u 위에 커서가 놓임

```
ubuntu linux study
I like linu ubuntux          ➡ 명령 모드로 전환되고 커서가 u로 이동한다.
~
~
~
```

# vim

- ❖ 모드 전환
  - ✓ i 와 a 명령 키의 차이
    - ● a 명령 키로 입력 모드로 전환한 후 Space Bar+linu를 입력하고 키를 누르면 커서가 위치한 a 다음부터 글자가 입력되고, Esc키를 입력하면 명령 모드로 전환되면서 마지막 입력 글자인 u 위에 커서가 놓임

```
ubuntu linux study
I like linu ubuntu linux    → 명령 모드로 전환되고 커서가 u로 이동한다.
~
~
~
```

# vim

❖ 모드 전환

✓ o 명령 키를 사용해 입력 모드로 전환하기

● 명령 모드에서 현재 커서가 위치한 그 다음 행에 글자를 입력

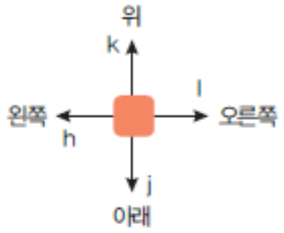
```
ubuntu linux study
I like linu ubuntu linux    → 명령 모드에서 o를 누르면 커서가 아랫행으로 이동한다.

~
~
```

# vim

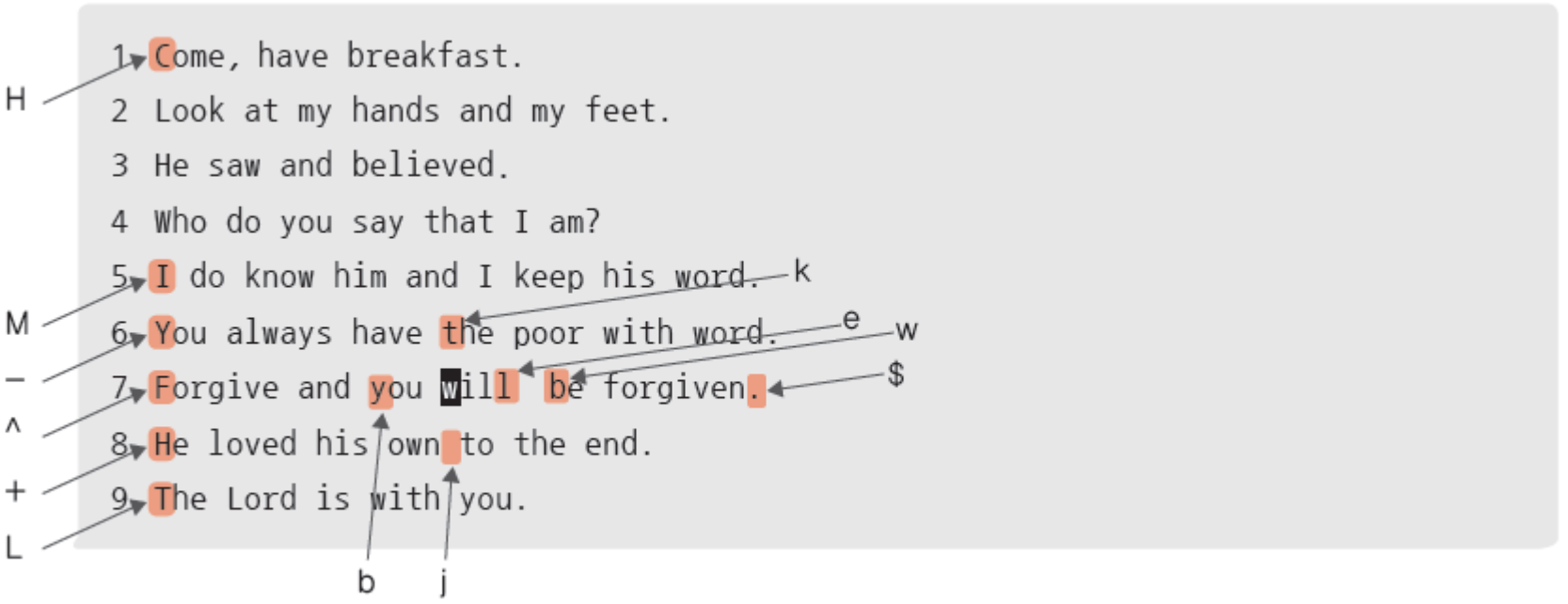
❖ 커서 이동

| 명령 키 | 기능 |
|---|---|
| k | 커서를 한 행 위로 이동한다. |
| j | 커서를 한 행 아래로 이동한다. |
| l | 커서를 한 글자 오른쪽으로 이동한다. |
| h | 커서를 한 글자 왼쪽으로 이동한다. |
| ^ 또는 0 | 커서를 현재 행의 처음으로 이동한다. |
| $ | 커서를 현재 행의 마지막으로 이동한다. |
| – | 커서를 앞 행의 처음으로 이동한다. |
| + 또는 Enter | 커서를 다음 행의 처음으로 이동한다. |
| H | 커서를 화면의 맨 윗행으로 이동한다. |
| M | 커서를 화면의 중간 행으로 이동한다. |
| L | 커서를 화면의 맨 아랫행으로 이동한다. |
| w | 커서를 다음 단어의 첫 글자로 이동한다. |
| b | 커서를 앞 단어의 첫 글자로 이동한다. |
| e | 커서를 다음 단어의 마지막 글자로 이동한다. |



# vim

❖ 커서 이동


1 Come, have breakfast.
2 Look at my hands and my feet.
3 He saw and believed.
4 Who do you say that I am?
5 I do know him and I keep his word.
6 You always have the poor with word.
7 Forgive and you will be forgiven.
8 He loved his own to the end.
9 The Lord is with you.
H
M
–
^
+
L
b
j
k
e
w
$

# vim

❖ 커서 이동

✓ 리눅스 vim은 화살표 키로도 커서를 이동할 수 있음

| 명령 키 | 화살표, 페이지 업·다운 키 | 명령 키 | 화살표, 페이지 업·다운 키 |
|---|---|---|---|
| k | 위 화살표 키(↑) | h | 왼쪽 화살표 키(←) |
| j | 아래 화살표 키(↓) | ^ 또는 0 | Home |
| l | 오른쪽 화살표 키(→) | $ | End |

# vim

❖ 화면 이동

✓ 파일 크기가 터미널의 화면 크기보다 클 경우 화면을 이동시키기 위한 명령

| 기존 명령 키 | 기능 | 추가 명령 키 |
|---|---|---|
| ^u (Ctrl+u) | 반 화면 위로 이동한다. | |
| ^d (Ctrl+d) | 반 화면 아래로 이동한다. | |
| ^b (Ctrl+b) | 한 화면 위로 이동한다. | Page Up |
| ^f (Ctrl+f) | 한 화면 아래로 이동한다. | Page Down |
| ^y (Ctrl+y) | 화면을 한 행만 위로 이동한다. | |
| ^e (Ctrl+e) | 화면을 한 행만 아래로 이동한다. | |

# vim

❖ 특정 행으로 바로 이동

✓ 원하는 행으로 커서를 바로 이동

- 50G -> 50행으로 이동
- :30(enter키) -> 30행으로 이동

| 명령 키 | 기능 |
|---|---|
| G(Shift+g) | 파일의 마지막 행으로 커서가 이동한다. |
| 행 번호G(Shift+g) | 지정한 행 번호로 커서가 이동한다. |
| :행 번호 | 지정한 행 번호로 커서가 이동한다(마지막 행 모드). |
| :$ | 파일의 마지막 행으로 커서가 이동한다(마지막 행 모드). |

# vim

❖ 내용 수정

| 명령 키 | 기능 |
|---|---|
| r | 커서가 위치한 글자를 다른 글자로 수정한다. |
| cw, #cw | 커서 위치부터 현재 단어의 끝까지 수정한다. #에는 수정할 단어의 수를 지정한다. 예를 들면 3cw는 커서 위치부터 세 단어를 수정한다. |
| s, #s | 커서 위치부터 Esc 키를 입력할 때까지 수정한다. #에는 수정할 글자의 수를 지정한다. 예를 들면 5s는 커서 위치부터 다섯 글자를 수정한다. |
| cc | 커서가 위치한 행의 내용을 모두 수정한다. |
| C | 커서 위치부터 행의 끝까지 수정한다. |

# vim

❖ 내용 삭제

| 명령 키 | 기능 |
|---|---|
| x, #x | 커서 위치의 글자를 삭제한다. #에는 삭제할 글자 수를 지정한다. |
| dw, #dw | 커서 위치의 단어를 삭제한다. #에는 삭제할 단어 수를 지정한다. |
| dd, #dd | 커서 위치의 행을 삭제한다. #에는 삭제할 행의 수를 지정한다. |
| D(Shift+d) | 커서 위치부터 행의 끝까지 삭제한다. |

# vim

❖ 명령 취소

| 명령 키 | 기능 |
| --- | --- |
| u | 명령을 취소한다. |
| U | 해당 행에서 한 모든 명령을 취소한다. |
| :e! | 마지막으로 저장한 내용 이후의 것을 버리고 새로 작업한다. |

# vim

❖ 내용 편집

✓ 복사, 붙이기 잘라내기

| 명령키 | 기능 |
|---|---|
| yy, #yy | 커서가 위치한 행을 복사한다. #에는 복사할 행의 수를 지정한다. |
| p | 커서가 위치한 행의 아래쪽에 붙인다. |
| P | 커서가 위치한 행의 위쪽에 붙인다. |
| dd, #dd | 커서가 위치한 행을 잘라둔다. 삭제와 같은 기능이다. #에는 잘라둘 행의 수를 지정한다. |

✓ 이름을 만들어서 사용

- 복사를 할 때 버퍼이름+yy를 사용하면 현재 행의 내용을 버퍼이름으로 저장
- 붙여넣을 때는 버퍼이름+p를 이용

# vim

❖ 실습

- ✓ vi로 새로운 파일인 exec.txt 파일을 생성
- ✓ 아래 내용을 입력

  Good morning everyone.

  Nice to meet you.

  I am a linux beginner.

  Now introduce yourself.
- ✓ 파일의 내용 저장
- ✓ 파일을 다시 열어서 3행의 커서 이동
- ✓ 파일의 내용 중 begginer 를 expert 로 수정
- ✓ 2행의 meet 를 삭제(dw)
- ✓ 2행의 you 부터 끝까지 삭제(D)
- ✓ 삭제한 내용 모두 복원(U)
- ✓ 내용 저장

# vim

- 실습
  - ✓ exec.txt 파일을 열기
  - ✓ 현재 행의 아래 행에 다음 내용을 추가

    Welcome Linux World.
  - ✓ 커서를 4행의 첫 글자인 I로 이동
  - ✓ I am을 We are로 수정
  - ✓ 커서를 마지막 행으로 이동
  - ✓ 마지막 행을 삭제
  - ✓ 명령 모드의 명령을 사용하여 파일을 저장하고 종료

# vim

❖ 내용 검색

✓ 검색은 명령어 입력 줄인 마지막 행으로 이동해서 입력

✓ : 명령어가 아닌 /이나 ?를 눌러서 검색할 키워드를 입력

| 명령 키 | 기능 |
|---|---|
| /문자열 | 문자열을 아래 방향으로 검색한다. |
| ?문자열 | 문자열을 위 방향으로 검색한다. |
| n | 원래 찾던 방향으로 다음 문자열을 검색한다. |
| N | 반대 방향으로 다음 문자열을 검색한다. |

# vim

- ❖ 내용 검색
  - ✓ 실습
    - 커서가 6행에 있을 때 검색하기 위해 /을 입력하면 마지막 행으로 이동
    - 검색할 문자열인 autoever 를 입력하고 Enter키를 누르면 커서 위치보다 뒤쪽에 위치한 같은 행의 autoever로 커서가 이동

```
I like  autoever linux
I like  autoever linux
Autoever editor vim study
Autoever editor vim study
I like  autoever linux
I like  autoever linux
~

/autoever
```

# vim

❖ 치환

| 명령 키 | 기능 |
|---|---|
| :s/문자열1/문자열2/ | 커서가 위치한 행에서 첫 번째로 나오는 문자열1을 문자열2로 바꾼다. |
| :%s/문자열1/문자열2/g | 파일 전체에서 모든 문자열1을 문자열2로 바꾼다. |
| :〈범위〉s/문자열1/문자열2/ | 범위 내 모든 각 행에서 첫 번째로 나오는 문자열1을 문자열2로 바꾼다. |
| :〈범위〉s/문자열1/문자열2/g | 범위 내 모든 행에서 문자열1을 문자열2로 바꾼다. |
| :〈범위〉s/문자열1/문자열2/gc | 범위 내 모든 행에서 문자열1을 문자열2로 바꿀 때 수정할지 여부를 묻는다. |

-- 대소문자 가리지 않고 바꾸기 수행(검색어 뒤에 /c를 붙여도 됨)
:set ignorecase

-- 대문자를 소문자로 변경할 때
: %s/.*/\L&/g

-- 소문자를 대문자로 변경할 때
: %s/.*/\U&/g

# vim

❖ 파일 읽어오기

| 명령 키 | 기능 |
|---|---|
| :r 파일 | 지정한 파일을 읽어들여 현재 커서 위치에 삽입한다. |
| :e 파일 | 지정한 파일로 전환한다(기존 파일을 :w로 저장한 뒤에 실행해야 한다). |
| :n | vi 시작 시 여러 파일을 지정했을 경우 다음 파일로 작업을 이동한다. |

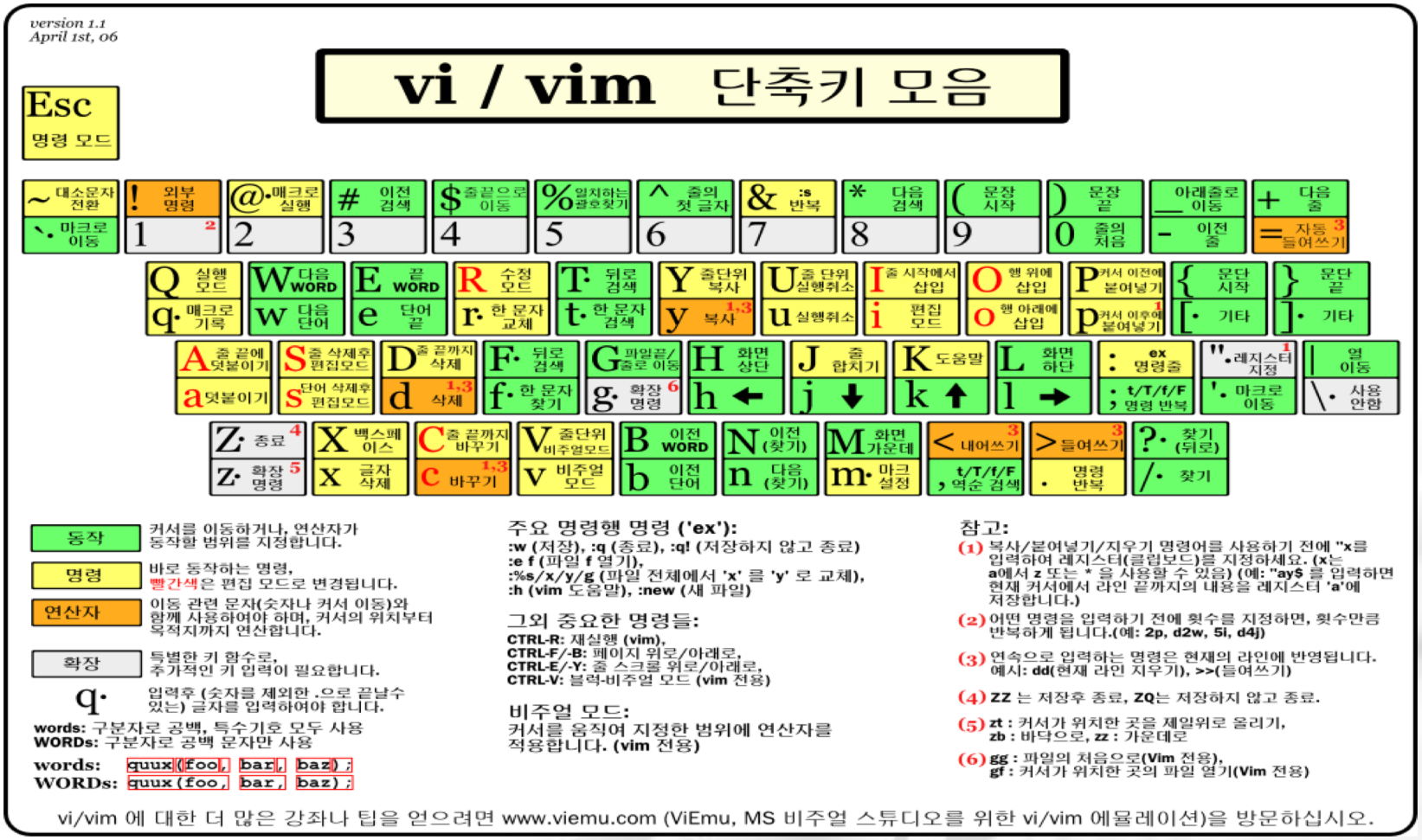
# vim

❖ Shell 명령 수행

| 명령 키 | 기능 |
|---|---|
| :! 셸 명령 | vi 작업을 잠시 중단하고 셸 명령을 실행한다(vi로 돌아오려면 Enter 키를 누른다). |
| :sh | vi를 잠시 빠져나가서 셸 명령을 실행한다(vi로 돌아오려면 exit 명령을 입력한다). |

# vim

❖ 단축키


version 1.1
April 1st, 06
vi / vim 단축키 모음
Esc
명령 모드
동작
명령
연산자
확장
q·
words: 구분자로 공백, 특수기호 모두 사용
WORDs: 구분자로 공백 문자만 사용
words: quux(foo, bar, 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: 블럭-비주얼 모드 (vim 전용)
비주얼 모드:
커서를 움직여 지정한 범위에 연산자를 적용합니다. (vim 전용)
참고:
(1) 복사/붙여넣기/지우기 명령어를 사용하기 전에 "x를 입력하여 레지스터(클립보드)를 지정하세요. (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), >>(들여쓰기)
(4) ZZ 는 저장후 종료, ZQ는 저장하지 않고 종료.
(5) zt : 커서가 위치한 곳을 제일위로 올리기,
zb : 바닥으로, zz : 가운데로
(6) gg : 파일의 처음으로(Vim 전용),
gf : 커서가 위치한 곳의 파일 열기(Vim 전용)
vi/vim 에 대한 더 많은 강좌나 팁을 얻으려면 www.viemu.com (ViEmu, MS 비주얼 스튜디오를 위한 vi/vim 에뮬레이션)을 방문하십시오.

# vim

❖ 기타 명령

| 명령 키 | 기능 |
|---|---|
| Ctrl+l(소문자 L) | 현재 화면을 다시 출력한다. |
| Ctrl+g | 현재 커서 위치의 행 번호를 마지막 행에 출력한다. |
| Shift+j(대문자 J) | 현재 행과 아랫행을 연결하여 한 행으로 만든다. |
| .(마침표) | 바로 직전에 했던 명령을 반복한다. |

# vim

- vi의 환경 설정 방법
  - ✓ 사용자 홈 디렉터리에 .exrc 파일에 설정 내용을 작성
  - ✓ 환경 변수 EXINIT에 지정
  - ✓ vi의 마지막 행 모드에서 명령으로 설정

# vim

❖ 환경 설정 옵션

| set 명령과 옵션 | 기능 |
|---|---|
| set nu | 파일 내용의 각 행에 행 번호를 표시한다(보이기만 할 뿐 저장되지는 않는다). |
| set nonu | 행 번호를 감춘다. |
| set list | 눈에 보이지 않는 특수문자를 표시한다(tab:^I, eol:$ 등). |
| set nolist | 특수문자를 감춘다. |
| set showmode | 현재 모드를 표시한다. |
| set noshowmode | 현재 모드를 감춘다. |
| set | set으로 설정한 모든 vi 환경 설정 값을 출력한다. |
| set all | 모든 vi 환경 변수와 현재 값을 출력한다. |

# vim

- 환경 설정 옵션
  - ✓ :set ai (autoindeant)
    - 윗라인과 같이 자동으로 들여쓰기.
  - ✓ :set si
    - 코딩 할때 if, for 같은 것을 입력 하고 다음 라인으로 이동시 자동으로 들여쓰기
  - ✓ :set paste
    - set ai, set si 같은 옵션을 사용할 경우 붙여 넣기를 하면 계단 현상이 발생함으로 붙여넣기를 사용할 경우에 이 옵션을 켜주면 계단 현상을 방지할 수 있음
  - ✓ :set ts=4 (tabstop)
    - [tab] 키를 입력 하였을때 이동하는 크기를 조정 합니다.
  - ✓ :set sw=4 (shiftwidth)
    - set si 했을 경우 들여쓰기 하는 깊이를 설정
  - ✓ :set et (expandtab)
    - [tab] 키를 입력 하였을 때 tab에 해당하는 space 만큼 이동

# vim

- 환경 설정 옵션
  - ✓ :set encoding=cp949 or utf8
    - 작업하는 컴퓨터 또는 개발 언어에서 기본 인코딩 타입을 설정
  - ✓ :set fenc=cp949 or utf8
    - 다른 인코딩으로 저장 하고 싶을때 명령을 내리고 저장하면 해당 인코딩으로 저장
  - ✓ :set t_ti= t_te=
    - 터미널 환경에서 vi를 종료 할 때 편집 하던 화면이 그대로 남도록 하는 것으로 BSD 에서는 기본값인데 Linux 에서는 화면이 지워짐
  - ✓ :set ruler
    - 우측 하단에 라인 및 컬럼 위치 표시및 전체 문서의 위치를 %로 표시
  - ✓ :set ff=unix  (dos, mac)
    - 라인변경 문자를 변경합니다.
  - ✓ :set key=<password>, set key=
    - 문서를 암호화 시키고 암호를 풀 수 있음
  - ✓ :set ic (ignorecase)
    - 검색 패턴 사용시 대소문자를 구별 하지 않음

# vim

- 환경 설정 옵션
  - ✓ :set wam
    - 저장하지 않고 종료시에 경고메시지 출력
  - ✓ :set sm (showmatch)
    - 괄호를 닫을 때 열기 괄호를 보여줌
  - ✓ 모든 명령의 취소는 no를 앞에 추가
    - :set noai

# vim

- vi의 환경 설정
  - ✓ 환경 변수를 이용한 설정 설정

```
adam@help:~$ EXINIT='set nu'
adam@help:~$ export EXINIT
```

# vim

- vi의 환경 설정
  - ✓ vi ~/.exrc

```
set number
set ts=3
set ai
```

# nano

❖ 개요

- ✓ 명령줄에서 작업할 때 텍스트 파일을 생성하거나 편집해야 하는 경우가 많은데 가장 강력하고 인기 있는 명령줄 편집자는 Vim과 Emacs
- ✓ GNU nano는 유닉스 및 리눅스의 메뉴 기반의 명령 줄 텍스트 편집기
- ✓ 구문 강조 표시, 다중 버퍼, 검색 및 정규식 지원으로 대체, 맞춤법 검사, UTF-8 인코딩 등과 같은 일반 텍스트 편집기에서 기대할 수 있는 모든 기본 기능이 포함되어 있음
- ✓ 나노 텍스트 편집기는 macOS 및 대부분의 Linux 배포판에 미리 설치되어 있음
- ✓ 설치 확인

```
adam@itstudy:~$ nano --version
 GNU nano, version 7.2
 (C) 2023 the Free Software Foundation and various contributors
 Compiled options: --disable-libmagic --enable-utf8
```

- ✓ 설치

```
sudo apt install nano
```

# nano

❖ 작업 화면


GNU nano 7.2 itstudy.txt
[ New File ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line