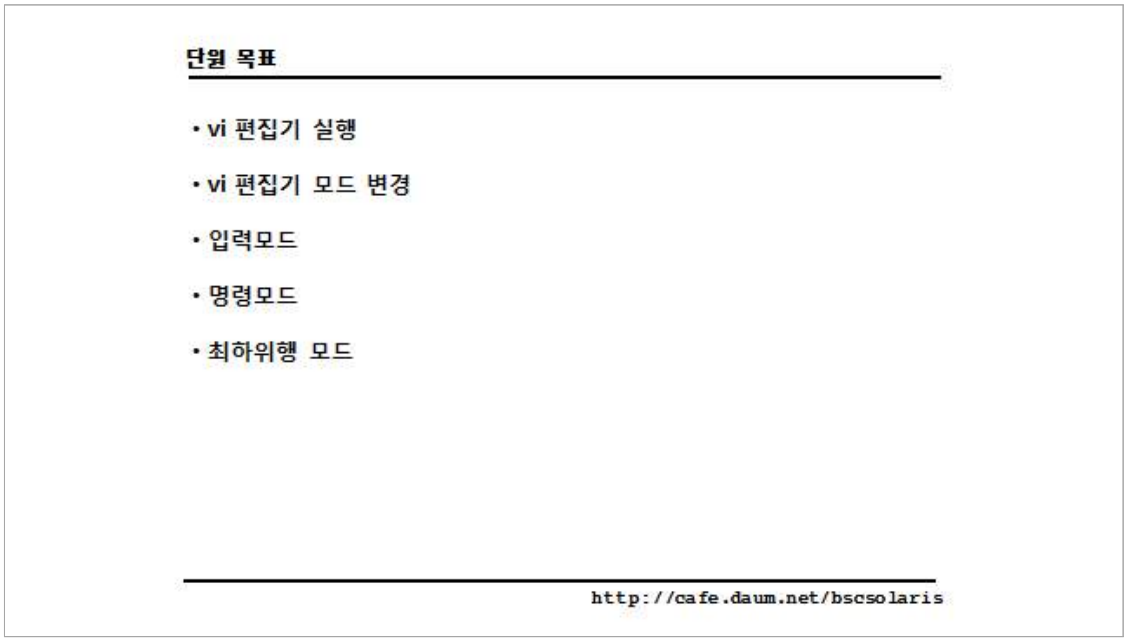
redhat CentOS

Unit 8
Improved Visual Editor

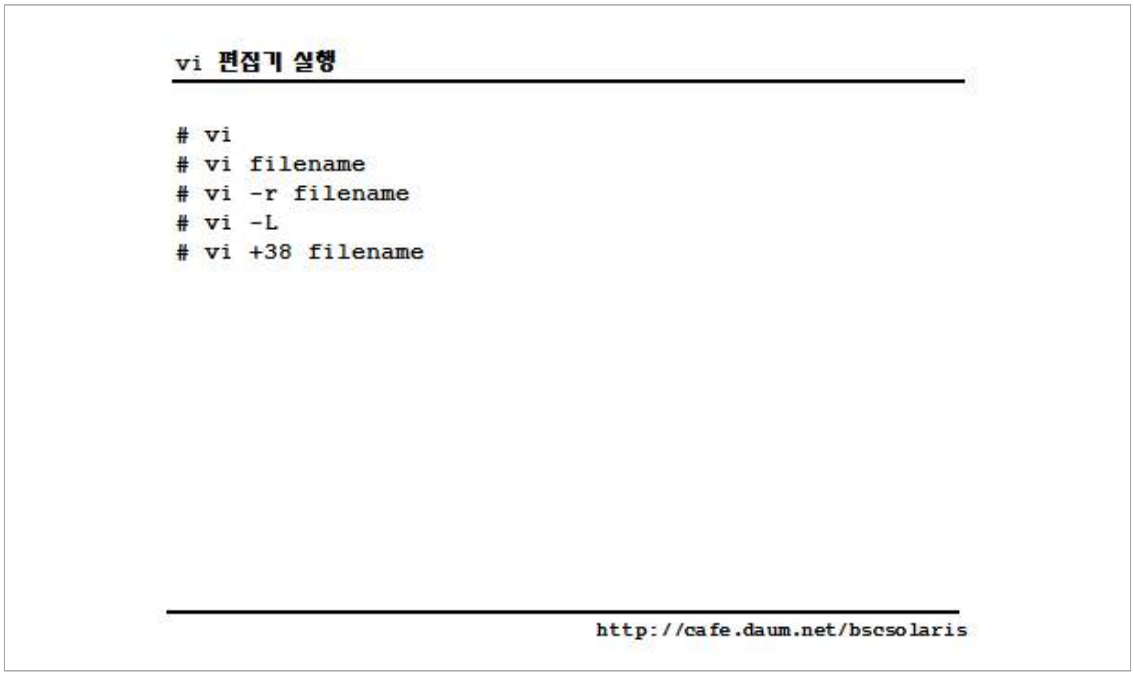
http://cafe.daum.net/bscsolaris



■ 현재 많이 사용되는 편집기 종류가 존재한다.

대표적인 편집기	편집기 설명
vi (Visual Editor)	vi 편집기는 유닉스 계열에서 가장 많이 사용되는 편집기이다. 1976년 빌 조이(Bill Joy)가 개발하였다. vi 편집기는 한 화면을 편집하는 비쥬얼 에디터(Visual Editor)이다.
vim (vi improved)	브람 무레나르(Bram Moolenaar)가 vi 편집기와 호환되면서 독자적으로 다양한 기능 추가하여 만든 편집기이다. 편집시에 다양한 색상을 이용하여 가시성을 높였으며, 패턴 검색시에 하이라이트 기능을 제공하여 빠른 검색이 가능하게 해준다. (http://www.vim.org)
emacs (Editor Macros)	리처드 스톨만이 개발한 고성능 문서 편집기이다. 매크로 기능이 있는 텍스트 교정 및 편집기이다. 초기 리처드 스톨만에 의해 개발 되었고, 이후에 제임스 고슬링(James Gosling)이 LISP 언어에 의한 환경 설정 및 에디터 기능을 확장시킬 수 있는 기능을 포함하여 '고슬링 이맥스'라는 이름으로 배포하였다.
pico (Pine Composer)	워싱턴 대학의 Aboil Kasar가 개발한 유닉스 기반의 텍스트 에디터로 pine 이메일 클라이언트 프로그램에 통합되어서 배포되었다. pico의 기본 인터페이스는 윈도우의 메모장(Notepad)와 유사하여 매우 단순한다.
nano	pico의 복제 버전이다. nano는 pico의 기능 이외에도 마우스 지원, 자동 들여쓰기, 정규 표현식 검색, 구문 강조 등의 기능을 추가로 제공한다.

[참고] GUI 편집기
- gedit, leafpad, 기타
- 오픈 소스 텍스트 편집기(GUI + CLI) 종류 및 추천
  https://jjeongil.tistory.com/292



1 VI(Visual Editor) 편집기 특징

vi('브이아이'로 부른다)는 Emacs와 함께 Unix 환경에서 가장 많이 쓰이는 문서 편집기이다. 1976년 빌 조이가 초기 BSD 릴리즈에 포함될 편집기로 만들었다. vi라는 이름은 한 줄씩 편집하는 줄단위 편집기가 아니라 한 화면을 편집하는 비주얼 에디터(visual editor)라는 뜻에서 유래했다. 간결하면서도, 강력한 기능으로 열광적인 사용자가 많다.

현재는 오리지널 vi를 사용하는 경우는 거의 없고, 일반적으로 기능을 확장하여 만들어진 vim 편집기를 주로 사용하고 있다. 리눅스의 기본 설계 원칙은 정보 및 구성 설정을 텍스트 기반 파일에 저장할 수 있도록 지원하는 것이다. 정보 및 구성 설정 파일은 (ㄱ) INI 파일 형식, (ㄴ) YAML 파일 형식, (ㄷ) XML 파일 형식등의 다양한 구조를 따른다.

RedHat 계열 서버에서 vi 편집기 사용하기 위해서는 vim-mininul 패키지가 필요하다. 이 패키지에는 vi 편집기의 핵심기능과 기본 vi 명령만 포함된다. 또한, vim-enhanced 패키지를 설치하면 훨씬 더 포괄적인 기능 세트, 온라인 도움말 시스템 및 튜토리얼 프로그램을 제공한다.

- VI(VIsual editor) -> /usr/bin/vi (vim-minimul 패키지)
- VIM(Visual editor IMproved) -> /usr/bin/vim (vim-enhanced 패키지)

다음과 같이 설치된 vim 관련 패키지를 확인할 수 있다.

```
# rpm -qa | grep -w vim
```

```
vim-common-8.0.1763-19.el8.4.x86_64
vim-enhanced-8.0.1763-19.el8.4.x86_64      # /usr/bin/vim
vim-filesystem-8.0.1763-19.el8.4.noarch
vim-minimal-8.0.1763-19.el8.4.x86_64       # /usr/bin/vi
```

* vim-minimal 패키지: 핵심 기능, 기본 vi 명령만 포함
* vim-enhanced 패키지: 포괄적인 기능 세트, 온라인 도움말 시스템 및 튜토리얼, vim 명령 포함

다음과 같이 설치된 vi/vim 명령을 확인할 수 있다.

```
# ls -lh /usr/bin/vi
# ls -lh /usr/bin/vim
```

다음과 같이 환경파일에 별칭을 설정하여 항상 vim 편집기가 실행될수 있도록 할 수 있다.

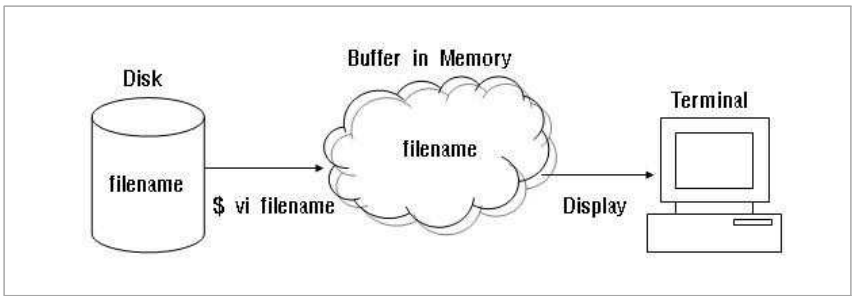
```
# gedit ~/.bashrc
```

```
alias vi='/usr/bin/vim'
```

```
# . ~/.bashrc
```

■ VI 편집기 동작원리

vi 편집기가 동작하는 원리를 보면 다음과 같이 버퍼에서 작업을 하게 된다. 그러므로 저장을 시키는 명령어를 입력하지 않는 이상 디스크상에 파일의 내용으로 저장되는 것은 아니다.



■ VI 편집기 실행 방법

명령어 형식	설명
$ vi	• 새 파일을 편집할 수 있는 화면이 나온다. • 작업을 한후에는 반드시 파일이름을 지정하여 저장하여 주어야 한다. 예) **:w filename** (최하위행 모드)
$ vi filename	• filename 이름을 가진 파일이 존재하는 경우, 해당 파일을 편집하고 filename 이름을 가진 파일이 존재하지 않는 경우, 새 파일을 편집할 수 있는 화면 상태가 된다.
$ vi -R filename $ view filename	• Readonly, 파일을 Readonly 상태로 열어준다. 중요한 파일, 여러 사람이 동시에 수정가능한 파일을 다룰때 편리하게 사용될수 있다. view 명령어와 동일한 기능을 수행할 수 있다.
$ vi -r filename	• Recovery, 이전 vi 편집 작업 중 비정상적으로 작업이 끝난 경우 편집하던 파일 복구시에 사용된다. 이 경우 사용자의 메일로 복구할 파일에 대한 정보가 오게 된다.
$ vi -L	• 이전 vi 편집 작업 중 비정상적으로 작업이 끝난 경우 복구할 파일들에 대한 전체적인 목록을 볼수 있다.
$ vi +38 filename	• 편집작업에 들어갈 때 특별한 명령어를 수행하면서 시작하는 경우 사용한다. 예) vi +38 filename(38번째 라인부터 시작)

■ VI 편집기 매뉴얼 확인

편집기에 대한 자세한 매뉴얼은 다음을 참고한다.

```
# cd /usr/share/vim/vim82/
# ls
```

```
autoload/      evim.vim           indent/     mswin.vim    spell/
bugreport.vim  filetype.vim       indent.vim  optwin.vim   synmenu.vim
colors/        ftoff.vim          indoff.vim  pack/        syntax/
compiler/      ftplugin/          keymap/     plugin/      tutor/
defaults.vim   ftplugin.vim       lang/       print/       vimrc_example.vim
delmenu.vim    ftplugof.vim       macros/     rgb.txt
doc/           gvimrc_example.vim menu.vim    scripts.vim
```

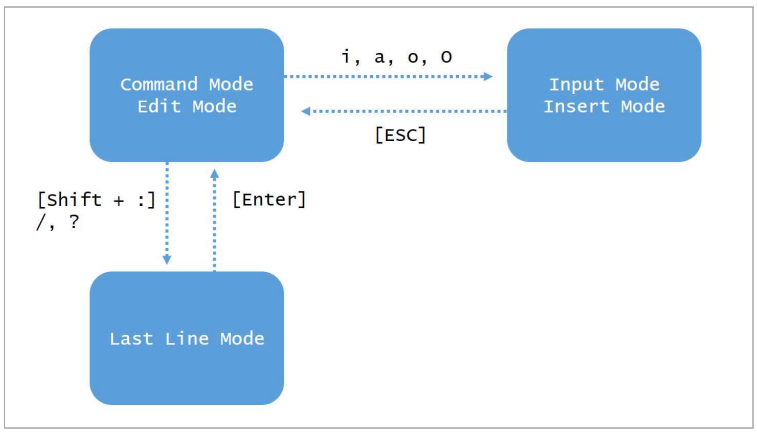
```
# cd tutor
# export LANG=ko_KR.UTF-8
# gedit tutor.ko.utf-8     (# vi tutor.ko.utf-8, # vimtutor)
```

```
===============================================================================
=    빔 길잡이 (VIM Tutor) 에 오신 것을 환영합니다    -      Version 1.7      =
===============================================================================

     빔(Vim)은 이 길잡이에서 다 설명할 수 없을 만큼 많은 명령을 가진
     매우 강력한 편집기입니다. 이 길잡이는 빔을 쉽게 전천후 편집기로 사용할
     수 있도록 충분한 명령에 대해 설명하고 있습니다.

     이 길잡이를 떼는 데에는 실습하는 데에 얼마나 시간을 쓰는 가에 따라서
     25-30 분 정도가 걸립니다.

     이 연습에 포함된 명령은 내용을 고칩니다. 이 파일의 복사본을 만들어서
     연습하세요. (vimtutor 를 통해 시작했다면, 이미 복사본을 사용하는
     중입니다.)

..... (중략) .....
```

편집기 [기본 기능] 익히기

- 편집기 작업시 [기본 기능]을 배워보자.
- 기본 기능: **이동, 삭제, 입력, 저장&종료**

■ 편집기 3가지 모드



■ 기본 작업 테스트

```
# cd /test
# vi filename
```

입력: k, j

입력: i

파일 내용

```
============================
hello
hello linux
Welcome To My Server!!!
hello linux
soldesk linux CentOS
Have a Good Time !!

============================
```

2번째 라인 : **linux -> liNux**

* 이동(move) -> 삭제(delete) -> 입력(insert) -> 저장/종료(save & quit)

(1). 이동(move): **방향키**
h, j, k, l

(2). 삭제(delete)
x(현재글자), dd(현재라인)

(3). 입력(input)
i(입력모드전환)

(4). 저장/종료(save & quit)
<shift + :>

```
: w!
: wq      /* w: write, q: quit */
: q!
: wq!
```

(5). 확인
[TERM1] 첫번째 터미널

```
# vi filename
```

[TERM2] 두번째 터미널

```
# cd /test
# cp /etc/services /test
# vi services
```

[참고] 편집기 참고 기능

■ [라인번호] 표시하기

: set number (: set nu)

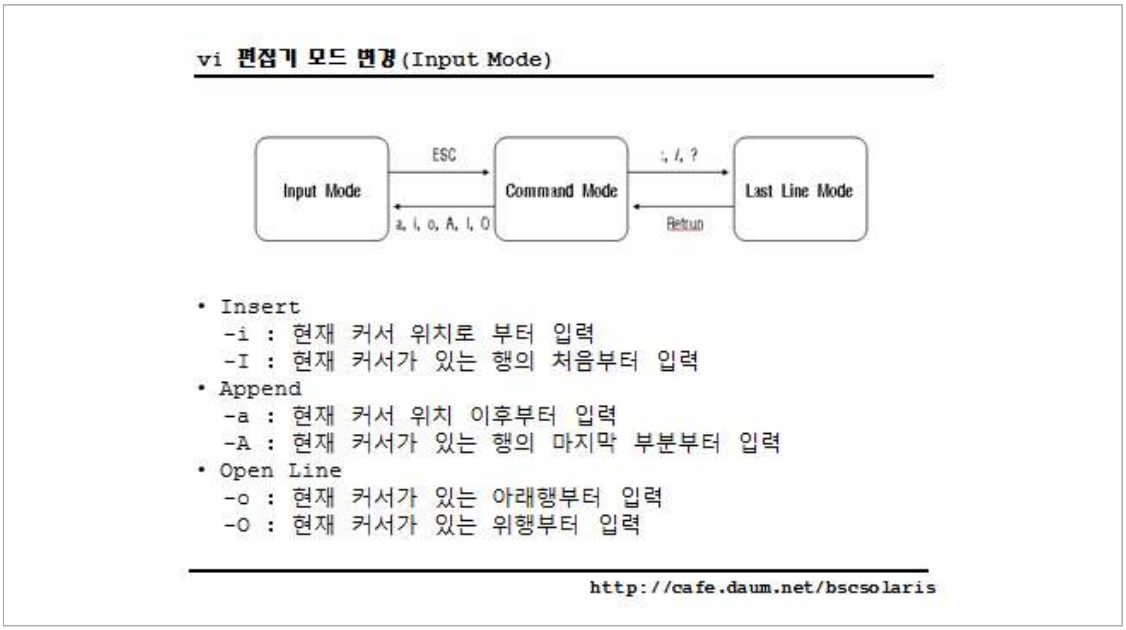
: set nonumber (: set nonu)

■ 특정 줄로 이동

: 30 30G nG

■ 특정 단어 검색

/ftp -> n -> N



vi 편집기에서는 3가지 모드가 있다. (ㄱ)명령행 모드(Comand Mode), (ㄴ)입력행 모드(Input Mode), (ㄷ)최하위행(라인) 모드(Last Line Mode)가 지원이 된다. 명령행 모드는 편집작업 및 기타 명령어를 입력할 수 있는 모드이고 입력행 모드는 입력만 할 수 있는 모드이다. 최하위행(라인) 모드는 저장 및 기타 검색 작업등을 할 수 있는 모드이다. (vim 편집기의 visual mode에 대해서는 이 교재에서 다루지 않는다.)

■ Command Mode(Edit Mode) 키 입력을 명령어로 해석
■ Input Mode(Insert Mode) 키 입력을 파일에 입력
■ Last Line Mode(Ex Mode) ex 명령어를 수행

(1) 입력행 모드(Input Mode)

vi 편집기 실행시 기본 모드는 명령행 모드이다. 명령행 모드에서 입력을 하기 위해서는 입력행 모드로 전환해야 한다. 이런 경우 다음과 같은 문자를 통해 입력 모드로 전환할 수 있다.
• a, A, i, I, o, O

.... liNux ....

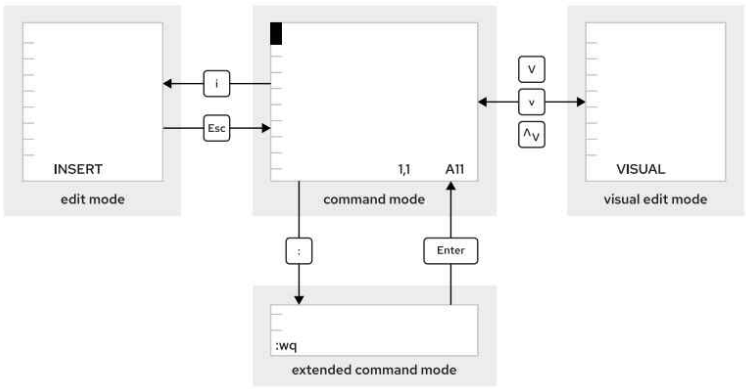
▶ Insert
- i : 현재 커서 위치로부터 입력한다.
- I : 현재 커서가 있는 행의 처음부터 입력한다.

▶ Append
- a : 현재 커서 위치 이후부터 입력한다.
- A : 현재 커서가 있는 행의 마지막부터 입력한다.

▶ Open Line
- o : 현재 커서가 있는 아래행부터 입력한다.
- O : 현재 커서가 있는 위행부터 입력한다.

[참고] 비쥬얼 모드(Visual Mode)

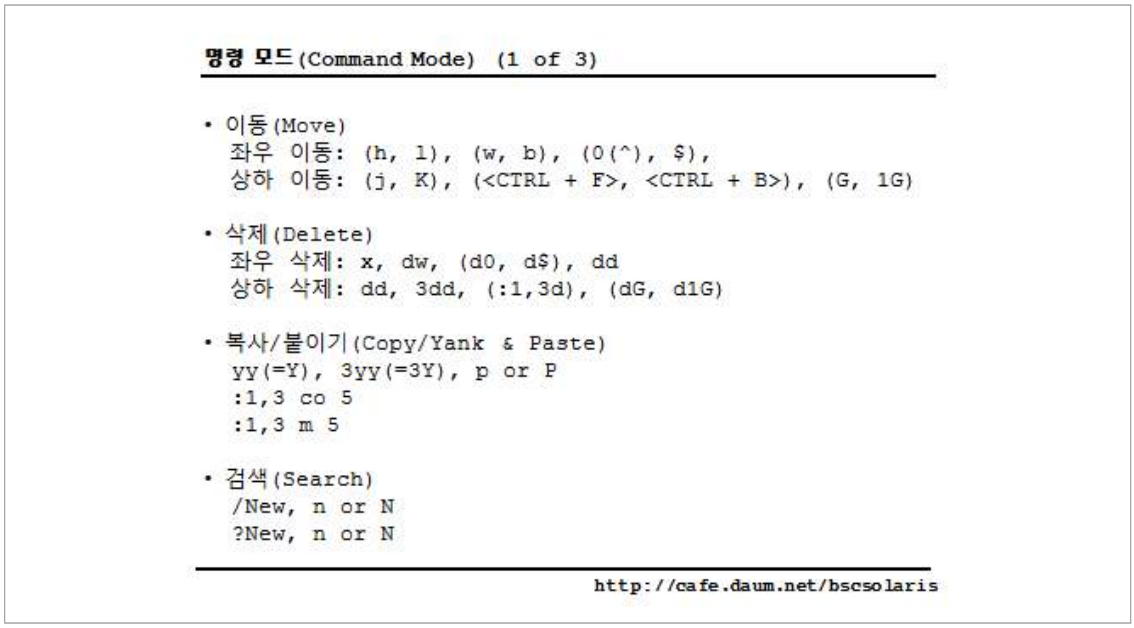
[참고] https://gracefulprograming.tistory.com/30



[그림] VIM 편집기 4가지 모드 종류

(redhat 공인교재 그림 참조)

■ 시각적 모드(Visual Mode)
* 문자 모드 : **v** => v
* 줄 모드 : **<Shift + v>** => V
* 블록 모드 : **<Ctrl + v>** => ^v



명령 모드(Command Mode) (1 of 3)

- 이동(Move)
  좌우 이동: (h, l), (w, b), (0(^), $),
  상하 이동: (j, K), (<CTRL + F>, <CTRL + B>), (G, 1G)
- 삭제(Delete)
  좌우 삭제: x, dw, (d0, d$), dd
  상하 삭제: dd, 3dd, (:1,3d), (dG, d1G)
- 복사/붙이기(Copy/Yank & Paste)
  yy(=Y), 3yy(=3Y), p or P
  :1,3 co 5
  :1,3 m 5
- 검색(Search)
  /New, n or N
  ?New, n or N

http://cafe.daum.net/bscsolaris

(2) 명령행 모드(Command Mode)

VI 편집기 기본기능: **이동**, **삭제**, **입력**, **저장&빠져나가기**

ⓐ 이동(Move)

■ **좌우 이동** : **(h, l)**, (w, b), **(0(^), $)**
■ **상하 이동** : **(j, k)**, **(<CTRL + F>, <CTRL + B>)**, (H, L), **(G, nG, 1G)**

[참고] :5 (=5G), :10 (=10G), 5, 10

- h : 한 문자 왼쪽으로 이동(←)
- l : 한 문자 오른쪽으로 이동(→)
- w : 한 단어 오른쪽으로 이동
- b : 한 단어 왼쪽으로 이동

- 0(^) : 라인의 처음 문자(라인의 처음)으로 이동
- $: 라인의 마지막으로 이동

- j : 한 문자 아래로 이동(↓), 다음 라인으로 이동
- k : 한 문자 위로 이동(↑), 이전 라인으로 이동

- <CTRL + F> : 다음 페이지로 이동
- <CTRL + B> : 이전 페이지로 이동

- H : 현재 페이지의 가장 첫번째 줄로 이동
- L : 현재 페이지의 가장 마지막 줄로 이동

- 1G : 문서의 첫 번째 라인으로 이동
- G : 문서의 마지막 라인으로 이동

ⓑ 삭제(Delete)

■ 좌우 삭제: x, dw, (d0, d$), dd
■ 상하 삭제: dd, 3dd, (:1,3d), (dG, d1G)

[참고] dd(=D), 3dd(=3D)

- x : 현재 커서 한 글자 삭제
- dw : 현재 커서 한 단어 삭제
- d0 : 현재 커서부터 라인의 처음까지 삭제
- d$: 현재 커서부터 라인의 마지막까지 삭제
- dd : 현재 라인 삭제

- 3dd : 현재 커서 라인을 포함해서 아래로 3개 라인 삭제
- :1,3d : 1번째 라인부터 3번째 라인까지 삭제

- dG : 현재 커서 라인부터 문서 마지막까지 삭제
- d1G : 현재 커서 라인부터 문서 처음까지 삭제

ⓒ 실행취소(Undo) & 다시실행(Redo)
■ undo: u
■ redo: <CTRL + r>

- u : 바로 이전에 상태로 되돌림
- U : 라인전체에 대해 이전 상태로 되돌림

* u(Undo) <--> <CTRL + r>(Redo)(WIN) <CTRL + Z>

ⓓ Join Line
■ J

- J : 현재라인에 아래 라인 붙이기

ⓔ Replace
■ r, R

- r : 현재 글자를 대치
- R : <ESC>키를 누르기 전까지 현재 글자 대치

[참고] vi 편집기 기본 기능
Linux -> LiNux
- 이동
- 삭제(x)
- 입력(i)
- 저장&빠져나가기

(3) 최하위행(라인) 모드(Last Line Mode)

ⓐ 복사/붙이기(Copy/Yank & Paste)

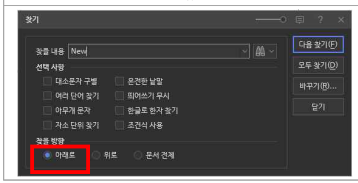
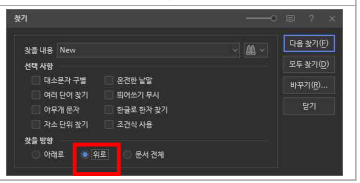
■ yy(=Y), 3yy(3Y), p or P
■ :1,3 co 5 (: 1,3 copy 5)
■ :1,3 m 5 (: 1,3 move 5) (: 3 m 5)

- yy : 현재 라인 복사(Yank)
- 3yy : 현재 커서 라인 포함해서 하위의 3개의 라인 복사
- :1,3 co 5 : 첫 번째 라인부터 3번째 라인까지 복사하여 5번째 라인 아래에 붙이기
- :1,3 m 5 : 첫 번째 라인부터 3번째 라인까지 5번째 라인 아래에 이동하기

- p : 현재 커서 아래에 붙이기
- P : 현재 커서 위에 붙이기

ⓑ 검색(Search)

■ /New n(정방향(↓)) or N(역방향(↑)) (WIN: <CTRL + F>)
■ ?New n(정방향(↑)) or N(역방향(↓))

아래로	위로
	

- /New : 현재 커서 라인부터 찾을려는 문자열(예: New) 검색
- ?New : 문서의 마지막 라인부터 찾을려는 문자열(예: New) 검색

- n : n(Next), 정방향다음번째 검색
- N : N(Next), 역방향으로 다음번째 검색

명령 모드(Command Mode) (2 of 3)

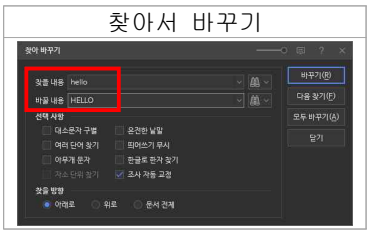
- 검색/바꾸기(Search & Replace)
  :%s/<찾을문자열>/<바꿀문자열>/g

```
:%s/hello/HELLO/g
:1,$s/hello/HELLO/g
:5,10s/HELLO/hello/g

:5,10s/^/#/
:5,10s/^#//

:5,10s/^/      /
:5,10s/^      //
```

http://cafe.daum.net/bscsolaris

ⓒ 검색/바꾸기(Search & Replace)

찾아서 바꾸기



■ (형식) :%s/<찾을문자열>/<바꿀문자열>/g
■ :5,10s/Old/New/g

[참고] "<찾을 문자열>"에는 "정규 표현식"을 사용할 수 있다.

예) :%s/hello/HELLO/g (:1,$s/hello/HELLO/g)
■ % => 1,$ => :5,10s/HELLO/hello/g
■ search(substitution, subtraction)
■ <찾을문자열>: "정규표현식" 사용 가능
■ <바꿀문자열>:
■ globally => %s/hello/HELLO/ , %s/hello/HELLO/g

```
예) :5,10s/^/#/                          /* 주석처리(comment) */
    :5,10s/^#//                          /* 주석해제(uncomment) */

    :5,10s/^/    / (4 blank character)   /* 들여쓰기 */
    :5,10s/^    //                       /* 내어쓰기 */
```

- :%s/hello/HELLO/g : 문서 전체에서 hello를 검색해서 HELLO로 변환
- :1,$s/hello/HELLO/g : 문서 전체에서 hello를 검색해서 HELLO로 변환
- :5,10s/^/#/ : 5번째 라인부터 10번째까지 라인의 처음부분에 '#' 처리

최하위행 모드(Last Line Mode)

• 저장 & 종료(Save & Quit)
  :w
  :w filename
  :w!
  :q
  :q!
  :wq
  :wq!
  :!CMD

http://cafe.daum.net/bscsolaris

ⓓ 저장(Save) & 빠져나가기(Quit)
■ :w /* w(write), 현재 파일에 저장 하기 */
■ :w filename /* 다른 이름으로 저장 하기 */
■ :w!(root use) /* 현재 파일에 강제적으로 저장 하기 */
■ :w! file /* 현재까지의 변경사항을 file로 저장 */
■ :3,10w file /* 3번째 라인부터 10번째 라인까지 file로 저장 */
■ :q /* q(quit), 편집기 종료 */
■ :q! /* 저장 안하고 편집기 종료 */
■ :wq /* 저장하고 편집기 종료 */
■ :wq!(root Use) /* 현재 파일에 강제적으로 저장하고 편집기 종료 */
■ :r file /* file의 내용을 현재 커서 위치에서 읽어 들임 */
■ :!CMD /* vi 편집기를 빠져나가지 않은 상태에서 쉘 명령어를 수행 */

[참고] 저장하고 빠져나가기(Save & Quit)의 여러가지 방법
■ :x
■ :wq
■ ZZ

```
[참고] vi 편집기 실행에 대해서
[TERM1]  # vi filename
         : wq!         <= 강제적(!)으로 저장(w)하고 빠져나가기(q)
         # cat filename
[TERM2]  # vi services
         : q!          <= 강제적(!)으로 빠져나가기(q), 저장 안하고 빠져나가기
         # cat services
```

>>
>>>> VIM 편집기는 개인적으로 익숙해 질때까지 연습을 하셔야 합니다. <<<<
>>

[참고] 저장하고 빠져 나가기
일반적으로 편집기 작업 완료 후 저장하고 빠져 나갈때 아래 내용을 이어서 수행한다.
* <ESC> => <Shift + :> => wq

[참고] VIM 편집 실행 중 비정상적으로 종료 되는 경우에 대한 처리

■ VIM 편집기로 파일 편집 중 비정상적으로 종료 되는 경우

```
# mkdir -p /test && cd /test && rm -rf /test/*
# vi filename
```

```
1111
:w!
```

=> 내용 입력 후 저장

```
# vi filename
```

```
1111
2222   <--- 새로 추가되는 내용(아직 추가된 내용이 저장되지 않은 상태)
```

-> !!! 편집기 강제 종료(저장하지 않은 상태) !!!!

■ 새로운 터미널 열고 다시 파일 열기

```
# cd /test
# vi filename
```

```
E325: ATTENTION
Found a swap file by the name ".filename.swp"
          owned by: root   dated: Thu May 15 17:44:39 2025
         file name: /test/filename
          modified: YES
         user name: root   host name: server1.example.com
        process ID: 8688
While opening file "filename"
             dated: Thu May 15 17:43:53 2025

(1) Another program may be editing the same file.  If this is the case,
    be careful not to end up with two different instances of the same
    file when making changes.  Quit, or continue with caution.
(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r filename"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file ".filename.swp"
    to avoid this message.

Swap file ".filename.swp" already exists!
[O]pen Read-Only, (E)dit anyway, (R)ecover, (D)elete it, (Q)uit, (A)bort: E   <-- 'E' 입력
```

```
1111

: q!
```

■ 파일 복구

```
# vi -L
```

```
Swap files found:
   In current directory:
1.    .filename.swp
          owned by: root   dated: Thu May 15 19:16:14 2025
         file name: /test/filename
          modified: YES
         user name: root   host name: server1.example.com
        process ID: 6631
   In directory ~/tmp:
      -- none --
   In directory /var/tmp:
      -- none --
   In directory /tmp:
      -- none --
```

```
# vi -r filename
```

```
Using swap file ".filename.swp"
Original file "/test/filename"
Recovery completed. You should check if everything is OK.
(You might want to write out this file under another name
and run diff with the original file to check for changes)
You may want to delete the .swp file now.

Press ENTER or type command to continue
```

```
1111
2222

:w filename.OLD
```

```
:q!
```

```
# rm -f /test/.filename.swp
# vi filename
-> 메세지가 나오지 않는다.
-> /test/.filename.swp 삭제하고 싶다면
    (ㄱ) # rm -f /test/.filename.swp
    (ㄴ) # vi /test/filename
        -> D
```

[참고] 이런 비슷한 상황이 열려 있는 파일을 다른 사용자가 같은 파일을 열때도 발생한다. 이런 경우에는 lsof CMD(ex: lsof /test/.filename.swp) 사용하여 사용중인 상태인지 확인한다.

VI 편집기 환경파일

vi 편집기의 동작하는 기능을 변경하기 위해서는 set 명령어를 사용한다. set 명령어 다음에 all을 사용하면 현재 편집기에 사용가능한 모든 기능변수들에 대한 현재 설정값을 표시한다.

vi 편집기의 기능을 현재 실행되는 편집 화면에서만 변경하기 위해서는 최하위행 모드에서 다음과 같은 방법을 사용한다.

```
# cd /test
# vi filename
```

```
: set all       (Last Line Mode)
: set number    (: set nu)
: set nonumber  (: set nonu)
: set tabstop=4 (: set ts=4)
--- Options ---
  ambiwidth=single  nohidden            nopreserveindent      termencoding=
noautoindent          history=50          prompt            noterse
noautoread          nohlsearch          noreadonly            textauto
noautowrite         noignorecase          remap             notextmode
noautowriteall        iminsert=0          report=2              textwidth=0
  background=light    imsearch=0          scroll=23         notildeop
nobackup            noincsearch           scrolljump=1          timeout
  backupcopy=auto   noinfercase           scrolloff=0           timeoutlen=1000
  backupext=~       noinsertmode        nosecure            nottimeout
  backupskip=/tmp/*   isprint=@,161-255    shell=/bin/bash       ttimeoutlen=-1
..... (중략) .....
  encoding=utf-8      maxmemtot=257666  nosmartcase             whichwrap=b,s
  endofline         nomodeline          nosmarttab            wildchar=<Tab>
  equalalways         modelines=5           softtabstop=0       wildcharm=0
  equalprg=           modifiable            startofline         wildignore=
noerrorbells          modified              swapfile            wildmode=full
  esckeys             more                  swapsync=fsync      window=47
noexpandtab           mouse=                switchbuf=          wrap
noexrc                mousemodel=extend     tabstop=8           wrapmargin=0
  fileencoding=       mousetime=500         tagbsearch          wrapscan
  fileformat=unix   nonumber                taglength=0         write
..... (중략) .....
```

```
: set tabstop=4       /* 탭간격 조정(처음 탭) */
: set shiftwidth=4    /* 탭간격 조정(중간 탭) */
: set number          /* 라인 번호 달기 */
: set autoindent      /* 자동 들여쓰기 */
```

■ VI/VIM 편집기 환경 설정 파일
* (관리자) /etc/vimrc
* (사용자) ~/.vimrc

■ 편집기에 따른 개인 환경 설정 파일
* (vi) -> **~/.exrc**
* (vim) -> **~/.vimrc** ->(~/.vimrc 없으면)-> **~/.exrc** ->(~/.exrc 없으면)-> **없음**

다음과 같이 $HOME/.vimrc 또는 $HOME/.exrc 파일에 정의할 수 있다.

```
# vi ~/.vimrc     (# vi ~/.exrc)
```

set tabstop=4	set ts=4
set shiftwidth=4	set sw=4
set number	set nu
set autoindent	set ai

위에서 정의된 내용을 테스트하기 위해서는 아래와 같이 수행해 본다.

```
# cd /test
# vi test.c
```

```
  1 int main()
  2 {
  3     printf("Hello");
  4     printf("WorldWn");
  5 }
```

[참고] VIM 편집기의 환경 설정 파일

■ 사용자 정의 $HOME/.vimrc 파일 생성
- (권장) 편집기 환경 설정 파일로는 ~/.vimrc 파일을 사용한다.
- 편집기 환경 설정 파일로 예전에 사용하던 ~/.exrc 사용해도 된다.

 # vi ~/.vimrc (# vi ~/.exrc)

```
set number
```

 # vi /etc/passwd

```
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
```

■ $HOME/.vimrc 파일의 샘플 파일 생성
 # cp /usr/share/vim/vim80/vimrc_example.vim ~/.vimrc

■ VIM 편집기 참고 파일(VIM 편집기 사용법 문서)
 # vi /usr/share/vim/vim80/tutor/tutor.ko.ufs-8

■ VIM 편집기의 Plugin 모음들
http://saelly.tistory.com/296
http://ethanschoonover.com/solarized

■ YAML & VIM 사용하는 $HOME/.vimrc 예
- 다음은 설정은 일반적으로 YAML 형식의 파일을 설정할 때 사용하는 .vimrc 파일의 예이다.

 # vi ~/.vimrc

```
syntax on
set ts=2
set sw=2
set ai
set nu
autocmd FileType yaml setlocal ts=2 sw=2 ai nu et
autocmd FileType python setlocal ts=2 sw=2 ai nu et
```

- syntax on : 구문 강조 사용
- FileType yaml : YAML 형식의 파일에 대해서만 적용하기
- setlocal : 현재 편집기에서만 설정을 변경할 때 사용
- ts(tapstop) : 파일 편집 중 <TAB> 칸 수 설정 하기(2칸으로 설정)
- sw(shiftwidth) : 자동들여쓰기를 할때의 <TAB>칸 수 설정 하기(2칸으로 설정)
- ai(autoindent) : 자동 들여 쓰기
- nu(number) : 편집기 라인 번호
- et(expandtab) : tab을 space로 변환

[참고] $HOME/.vimrc 파일에 대한 예제
[CentOS7] Vim 설치 & 옵션 설정
https://hanbyoru.tistory.com/198

[참고] Python 개발자를 위한 Vim 설정!
https://m.blog.naver.com/onevibe12/222003789290
https://ko.linux-console.net/?p=2091
https://ko.linux-console.net/?p=2091https://www.slideshare.net/JasonJungsuHE0/

[작업] VIM 편집기 사용법을 익힌다.

[작업1] (main/server1/server2) /etc/hosts 파일을 내용을 다음과 같이 편집한다.

① (main/server1/server2) /etc/hosts 파일 편집
- 작업 대상: **main, server1, server2**
- 작업 내용:
  * vi/vim 편집기를 사용하여 /etc/hosts 파일을 편집아래와 같이 내용을 추가한다.

```
# vi /etc/hosts
---------- /etc/hosts ------------
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

#
# Sfecific configuration
#
192.168.10.10  main.example.com     main
192.168.10.20  server1.example.com  server1
192.168.10.30  server2.example.com  server2
-----------------------------------
```

[작업2] (server1) WEB 서버 설정 예 - httpd.conf 파일 편집

- 작업 대상: **server1**
- 작업 내용:
  - 아파치 웹 서버의 httpd.conf 설정파일을 수정한다.

① (선수 작업) 필요한 패키지 설치 및 파일 복사

```
# (servre1)
# 작업 디렉토리 생성
mkdir -p /test && cd /test && rm -rf /test/*
# 웹 패키지 설치
yum -y install httpd mod_ssl
# 웹 설정 파일 복사
cp /etc/httpd/conf/httpd.conf /test
```

② (작업) httpd.conf 파일 설정

 # vi /test/httpd.conf

```
[수정전]
#ServerName www.example.com:80
[수정후]
ServerName 192.168.10.20:80
```

③ 작업 확인

(주의) 미리 웹서비스가 기동 중이 아니어야 한다.
systemctl disable --now httpd

```
# httpd -f /test/httpd.conf
# firefox http://192.168.10.20/
```

=> firefox 출력 내용 확인 후 종료

④ (정리작업)

```
# (server1)
pkill httpd
```

[작업3] (server1) FTP 서버 설정 예 - vsftpd.conf

- 작업 대상: **server1**
- 작업 내용:
  - vsftpd의 설정 파일인 vsftpd.conf, user_list 파일들을 설정한다.

① (선수 작업) 필요한 패키지 설치 및 파일 복사

```
# (servre1)
# 작업 디렉토리 생성
mkdir -p /test && cd /test && rm -rf /test/*
# 패키지 설치
yum -y install vsftpd ftp
# 실습 파일 복사
cp /etc/vsftpd/{vsftpd.conf,user_list} /test
```

② (작업) user_list, vsftpd.conf 파일 편집

```
# cd /test
# vi /test/user_list
```

```
[수정전]
root
[수정후]
~~root~~
```

```
# vi /test/vsftpd.conf
```

```
[수정전]
anonymous_enable=NO
[수정후]
anonymous_enable=YES
```

③ 작업 확인

```
# vsftpd /test/vsftpd.conf
# ftp localhost
fedora 사용자로 로그인
ftp> quit
```

④ (정리 작업)

```
# (server1)
pkill vsftpd
```

[참고] VI/VIM 단축키 모음