

# vim editor (1/4)

김선영

[sunzero@gmail\(dot\)com](mailto:sunzero@gmail(dot)com)


버 전: 2014-10

인사이트 출판사 <http://blog.insightbook.co.kr>

가메 출판사 <http://www.kame.co.kr>

저자홈페이지 <http://sunzero.tistory.com>

# Ch1.vim의 배경과 설치



항상 과거를 돌아보라. 그러면 뭔가 배울 수 있을 것이다. - 폴 새뮤얼슨

## ❖ vi (visual editor)

- ▶ UNIX / Linux 에서 가장 많이 사용하는 에디터
- ▶ 1976년 BSD의 Bill Joy가 개발
  - ▣ (Sun Microsystem 창업자)
  - ▣ ed (line editor)를 사용하던 시절에는 천재들만 프로그래밍을 할 수 있었다.  
왜냐하면 자신이 선언한 함수, 변수를 모조리 외우고 있어야만 프로그래밍이 가능했기 때문이다.



# vim - vim improved

- ▶ vi 에 추가적인 확장 기능 부여 (Amiga로부터...)
- ▶ 리눅스에서는 vi 대신에 **vim** 이 사용되어짐 (vim 은 vi 의 기능을 모두 포함)
  - ▣ 단 root 유저가 사용하는 vim은 최소화된 vim 에디터로서 static linking된 바이너리임  
(라이브러리 디렉터리가 데미지를 입어도 구동할 수 있도록...)
- ▶ 대부분의 UNIX에서도 complementary package 로 제공되어짐



Amiga 500 (1987)

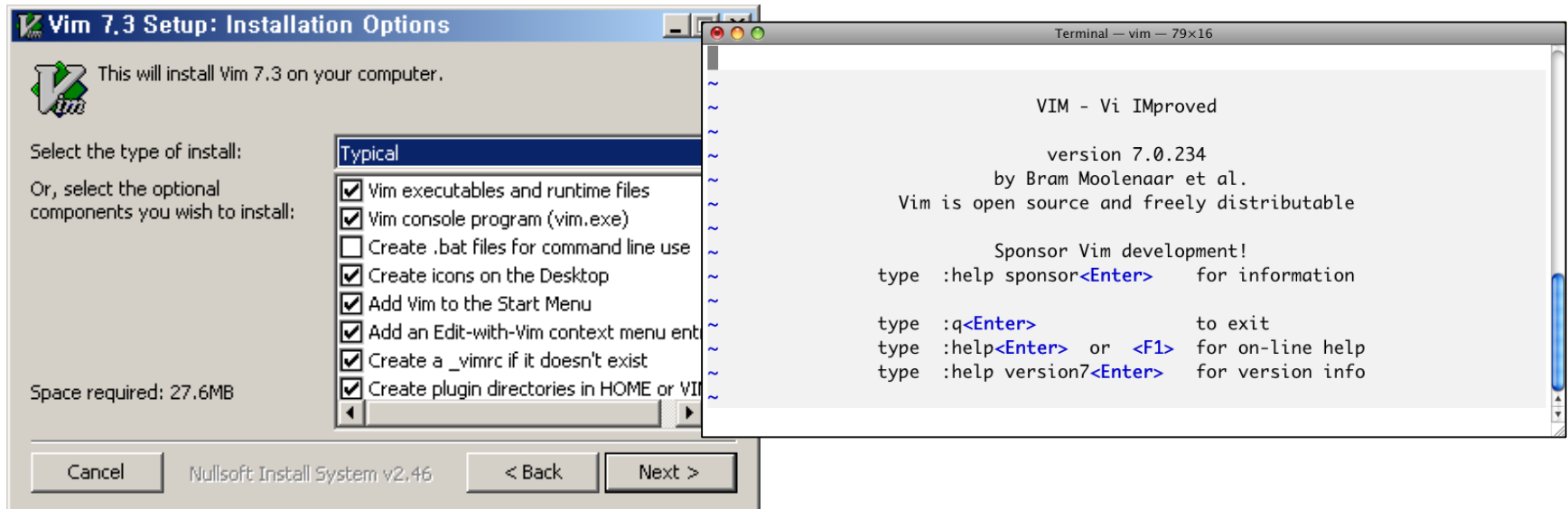
부잣집 도련님의 상징!

# vim : platform

❖ vim은 다양한 플랫폼을 지원한다.

▶ Linux, UNIX, Mac OSX, Windows ...

▶ 빨리 익숙해지려면 Windows gvim을 메모장 대신에 사용하는 것이 좋다.



# vi / vim alias

- ❖ alias를 통해 vi=vim으로 구동시킬 수 있다.

```
$ alias vi=vim
```

- ❖ login 시에 자동으로 설정되도록 하려면...

▶ ~/.bashrc 에 설정해둔다. (debian은 ~/.bash\_aliases)

```
# .bashrc (.bash_aliases on Debian)
```

```
...생략...
```

```
alias vi=vim
```

↘ alias를 적용하고 싶지 않을 경우는 prefix로 \ 를 사용하면 된다.

# Ch2.Vim 입문



시작이 반이다. - 아리스토텔레스

# vim의 시작

## ❖ vim [filename]

- ▶ 특정 파일명을 열면서 시작

```
$ vim mytext.txt
```

- ▶ 파일명이 "-" 일 경우에는 -stdin- 을 의미함

```
$ find . -name "*.txt" | vim -
```

## ❖ vi의 기본 작동 모드

- ▶ 일반모드, 입력모드, 명령행모드
  - ↳ vim은 여기에 비주얼 모드등이 추가된다.



# modes

## ❖ 전통적인 vi 는 3가지 모드를 가짐

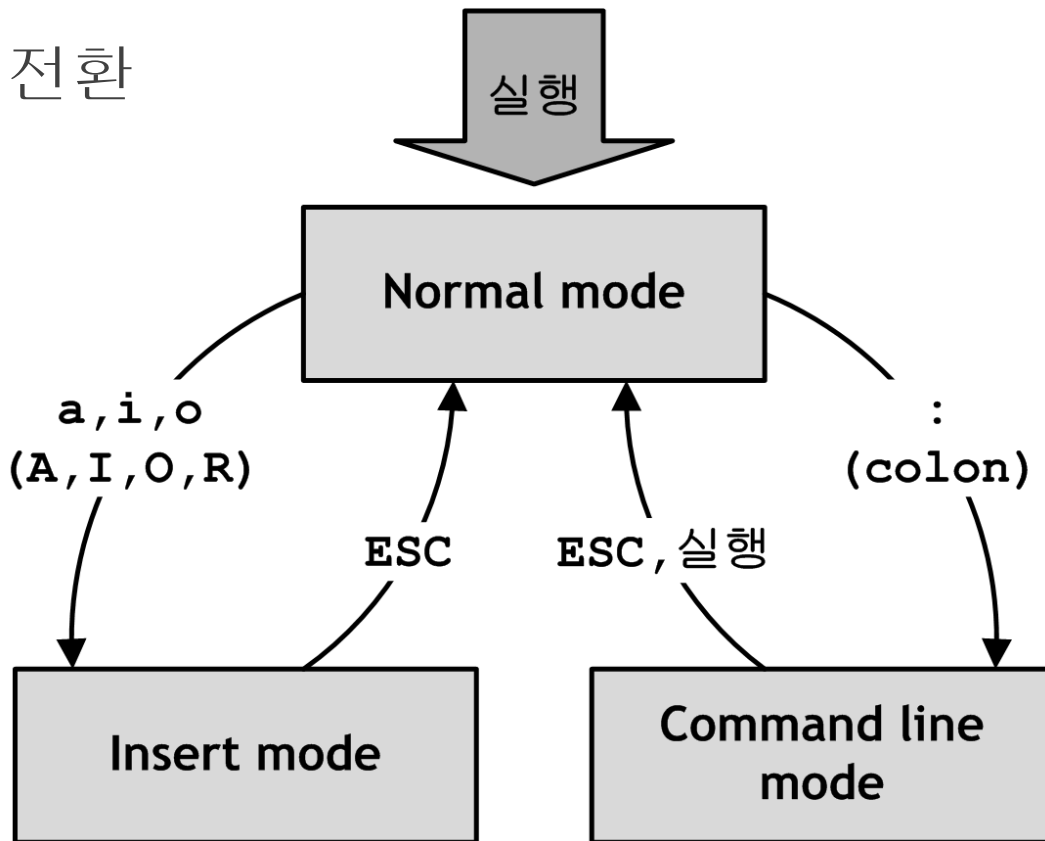
- ▶ 일반모드: **normal mode** (or command mode)
- ▶ 입력모드: **insert mode**
- ▶ 명령행모드: **command-line mode** (or colon mode)

## ❖ vim 은 추가 모드를 가짐

- ▶ 비주얼모드: **visual mode**
  - ↳ 마우스를 대신하는 드래그 모드 (아래아 한글의 F3키와 유사)

# modes (con't)

## ❖ 모드 전환



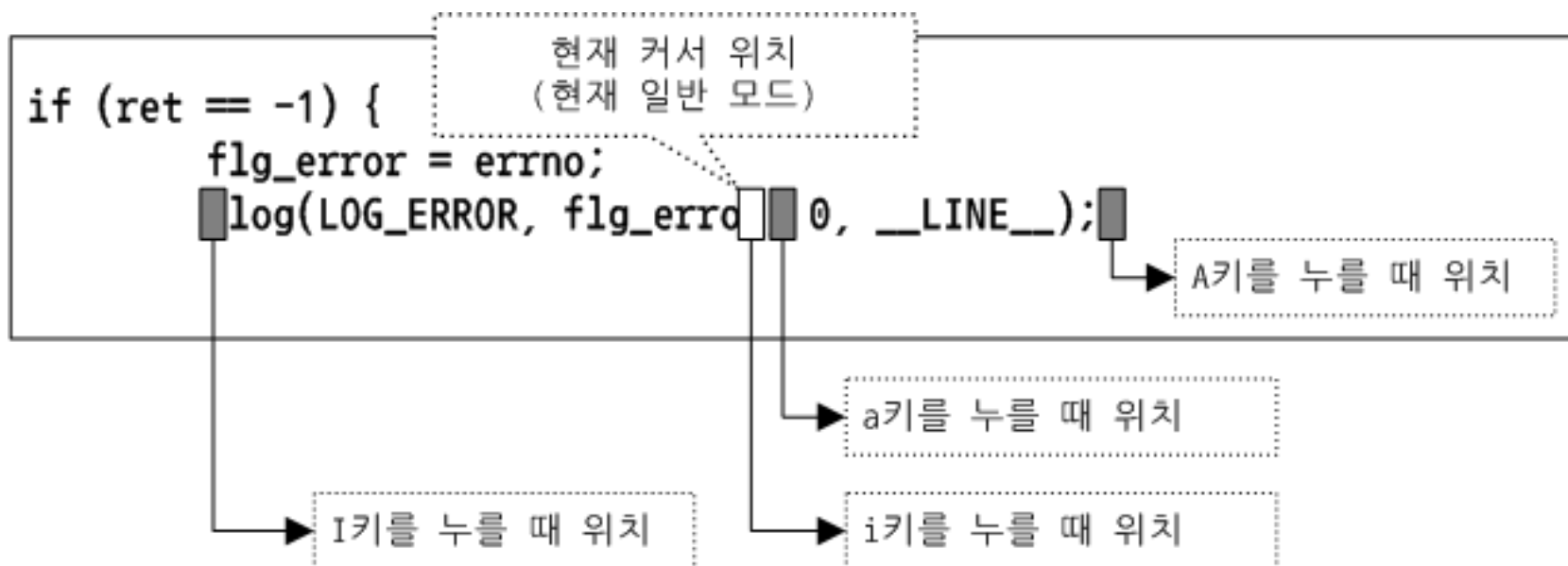
# modes : key

명령어	설명
a, A	a (append)는 현재 커서 위치에서 한 칸 뒤로 이동한 후 입력 모드로 전환
	A는 현재 행의 끝으로 이동한 후, 입력 모드로 전환됩니다.
i, I	i (insert)는 현재 커서 위치에서 입력 모드로 전환됩니다.
	I는 현재 행의 맨 앞으로 이동 후, 입력 모드로 전환됩니다.
o, O	o (open line)는 현재 행 아래에 새로운 행을 하나 만든 후 입력 모드로 전환
	O는 현재 행 위에 새로운 행을 하나 만든 후 입력 모드로 전환됩니다.
R	수정 (replace) 모드로 작동하므로 모든 글자는 덮쓰여집니다.

\* 처음엔 가장 많이 사용하는 i와 o 키를 주로 외워두자.

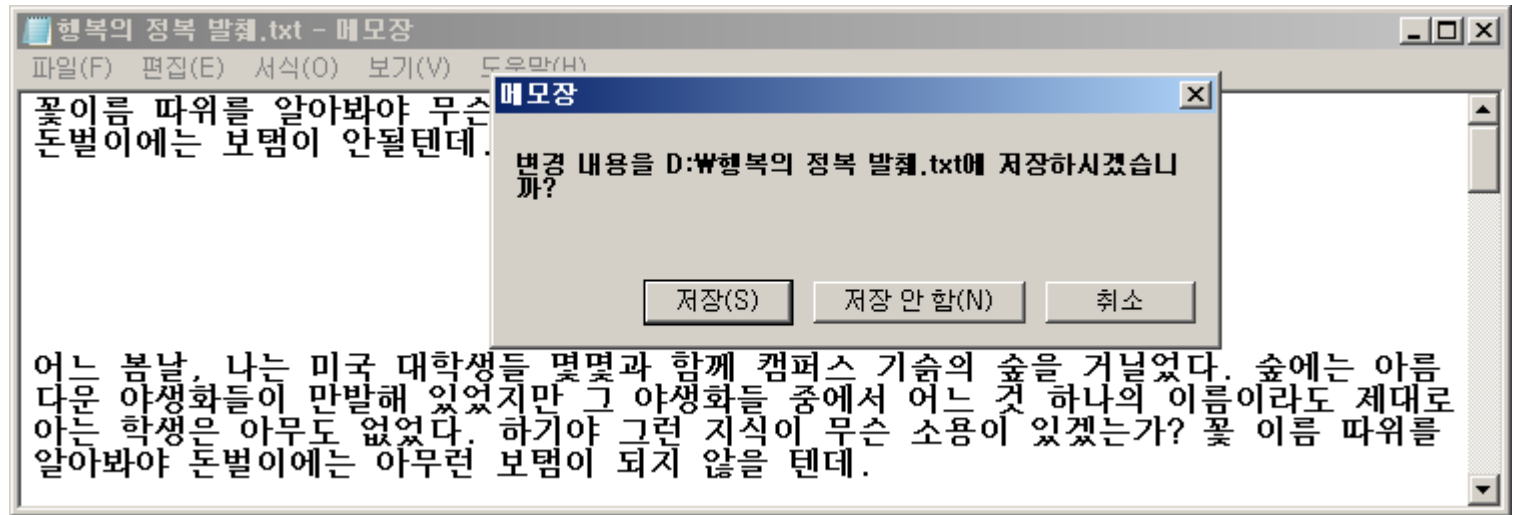
# modes : insert : a / A , i / I

- ❖ 대소문자에 따라 차이가 있다.



# modes : commandline : exit

- ❖ 강제 종료 명령 : ! 를 뒤에 더한다.



```
~  
~  
~  
~  
E37: No write since last change (add ! to override) 6,0-1 All
```

# modes (con't)

## ❖ normal mode가 필요한 이유

▶ = Text-based에서는 GUI menu가 없으므로 **short-cut**으로 구현해야..



# Practice

❖ 아래 박스 내용을 clientlist.txt 로 저장한다.

1304, Yona Yahav, M, 42, MP1

1294, Kebin Robinson, M, 41, CP1

1601, Steven Choi, M, 34, CP3

1314, TW Yoon, F, 46, CP1

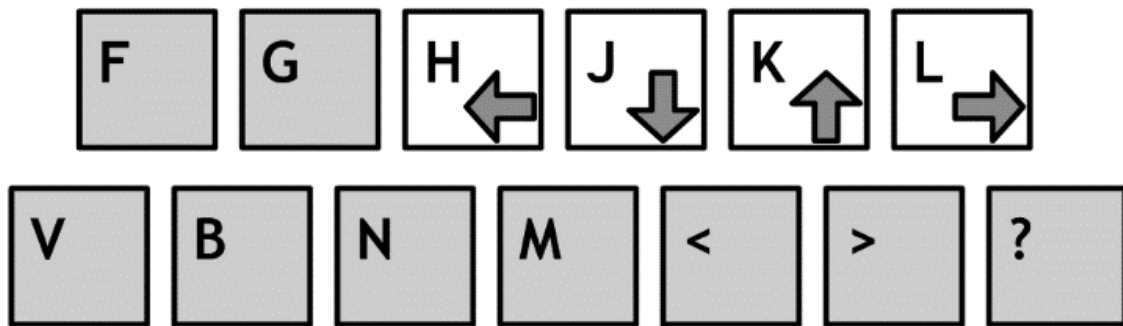
1315, Rina Suzuki, F, 36, MP1

1600, Robert Kim, M, 32, CP3

1297, Rarry Robinson, M, 38, CP2

1. vim clientlist.txt
2. **i** 를 눌러 입력 모드로 전환한다.
3. 위 코드의 내용을 입력한다.
4. **<ESC>**를 눌러 입력 모드를 종료하여, 일반 모드로 전환한다.
5. **:w** 명령으로 파일을 저장한다.
6. **:q** 로 vim을 종료한다.

- ❖ 화살표키 대신에 **h,j,k,l**을 사용해 본다.



- ▶ 지금 사용되는 **106**키 키보드는 과거에는 없던 물건이다.
  - ▣ 따라서 vi 계열은 **ten-key**가 없는 곳에서 이동명령을 사용할 수 있게 **hjkl**에 이동 기능을 맵핑해두었다.
  - ▣ 부가서비스로 손목을 최소한으로 이동하게 하는 효과가 있다.
  - ▣ 무릇 IT종사자는 좀 게을러야 발전이 생긴다. = 꿈수가 만들어진다.



# cursor : UNIX keyboard



❖ 과거의 유닉스 키보드의 모습을 간직한...

▶ Happy Hacking Keyboard와 103 Key 한글 키보드의 비교

# cursor : movement

명령어	설명
[#]h	좌로 #칸 이동, #의 생략 시는 1칸
[#]j	아래로 #칸 이동, #의 생략 시는 1칸
[#]k	위로 #칸 이동, #의 생략 시는 1칸
[#]l	우로 #칸 이동, #의 생략 시는 1칸
^	행의 맨 앞으로 이동
\$	행의 맨 끝으로 이동

# scroll

❖ PgUp, PgDown도 없다고 생각하자.

Ctrl B = Page Up = 위로 한 화면 스크롤

Ctrl F = Page Down = 아래로 한 화면 스크롤

Ctrl U = 위로 1/2화면 스크롤

Ctrl D = 아래로 1/2화면 스크롤

# goto # line

- ❖ 특정 라인으로 이동할 때 **scroll**을 사용하는 것은 비효율
- ▶ **line number**를 직접 입력하자.

명령어	설명
<b>[#]gg</b>	#행으로 이동합니다. #이 생략되면 1을 의미합니다.
<b>[#]G</b>	#행으로 이동합니다. #이 생략되면 마지막 행을 의미합니다.
<b>:#</b>	#행으로 이동합니다.

명령어	설명
<b>&lt;CTRL-G&gt;</b> <b>:file</b>	현재 문서 위치 정보를 하단 상태 바에 표시합니다.

# buffer : delete, cut

## ❖ 삭제

- ▶ vi 에서의 삭제 = 임시 버퍼에 잘라내기



명령어	설명
<b>x</b>	커서에 위치한 문자 삭제 (<Delete> 키와 같음)
<b>dd</b>	현재 행을 삭제
<b>:d</b>	
<b>D</b>	현재 컬럼 위치에서 현재 행의 끝부분까지 삭제 (d\$와 동일)
<b>J</b>	아래 행을 현재 행의 끝에 붙임 (아래 행의 앞부분 공백은 제거됨)

3dd라고 명령하면?

# buffer : del. newline character

```
if (ret == -1) {  
    flg_error = errno;  
    elog(LOG_ERROR, flg_error, __LINE__);  
    return -1;  
}
```

행 끝에서 x 를 누른다고  
밑에 행이 올라오지  
않는다.

```
if (ret == -1) {  
    flg_error = errno;  
    elog(LOG_ERROR, flg_error, , __LINE__);  
 return -1;  
}
```

현재 커서 위치  
(현재 일반 모드)

공백 부분

J

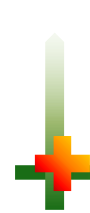
```
if (ret == -1) {  
    flg_error = errno;  
    elog(LOG_ERROR, flg_error, 0, __LINE__); return -1;  
}
```

공백은 무조건  
한칸으로 계산

# buffer : paste

명령어	설명
p	현재 행에 붙여 넣습니다 (put) .
:pu	개행 문자가 포함된 경우에는 현재 행의 아래에 붙여 넣습니다.
P	현재 행의 위쪽에 붙입니다. (대문자)

5p라고 명령하면?



# buffer : copy

명령어	설명
yy	현재 행을 레지스터에 복사 ( <b>yank</b> ) 합니다.
:y	
y	



# buffer : undo / redo

명령어	설명
<b>u</b>	<b>undo</b> 기능입니다. 바로 이전에 행한 명령 한 개를 취소합니다.
<b>CTRL-R</b>	<b>redo</b> 기능입니다. 바로 이전에 취소했던 명령을 다시 실행합니다.

➤ 이전 명령어 반복 : . (**dot**)

= normal mode에서 **IHello** (대문자 **i** + **Hello**) 라고 타이핑 후에 **<ESC>**를 누른다.

== **dot** 키를 누르면 어떤 일이 발생하는가? (문서 중간에서 명령하면?)

# cmd : range

- ❖ 명령행 모드에서 범위를 지정해서 명령
  - ▶ 특정 행 범위의 삭제에 편리

명령어	설명
<b>:20d</b>	20번 행을 삭제합니다.
<b>:10,25d</b>	10~25번 행을 삭제합니다.
<b>:10,\$d</b>	10~마지막 행까지 삭제합니다.
<b>:%y</b>	문서 전체를 복사합니다. %는 1,\$와 동일합니다.
<b>:.,+20y</b>	현재 행부터 아래로 스무 행을 복사합니다.
<b>:-10,+5d</b>	현재 행부터 위로 10행, 아래로 5행, 총 열여섯 행을 삭제합니다.
<b>:40pu</b>	40번 행에 레지스터의 내용을 붙여넣습니다.

# cmd : range (con't)

## ❖ 범위 연산 meta character

기호	의미
.	현재 행을 의미합니다.
\$	마지막 행을 의미합니다.
+#	현재 위치에서 #만큼 아래 행을 의미합니다.
-#	현재 위치에서 #만큼 위 행을 의미합니다.
%	문서(파일) 전체를 의미합니다.

# visual mode

❖ mouse의 **drag** 기능을 대신

▶ 매우 직관적이고 편리하다. (vim 확장 기능)

명령어	설명
<b>v</b>	(소문자) 일반 비주얼 모드로 현재 커서 위치에서 블록을 지정합니다.
<b>V</b>	(대문자) <b>visual line mode</b> 로, 현재 커서가 위치한 행에서 행 단위로 블록을 지정
<b>CTRL-V</b>	<b>visual block mode</b> 로, 열( <b>column</b> ) 단위로 블록을 지정합니다. (<CTRL-V>가 예약되어 사용할 수 없는 경우는 <CTRL-Q>로 대신할 수 있다.)*

\* 윈도 환경에서는 **CTRL-V**가 붙이기로 사용될때 **CTRL-Q**로 대체할 수 있다.

# visual mode

```
1304, Yona Yahav, M, 42, MP1
1294, Kebin Robinson, M, 41, CP1
1601, Steven Choi, M, 34, CP3
1314, TW Yoon, F, 46, CP1
1280, Wendy Johnson, 36, CP1, F
1361, Marc Herold, M, 45, MP2
1315, Rina Suzuki, F, 36, MP1
1600, Robert Kim, M, 32, CP3
1297, Rarry Robinson, M, 38, CP2
~
-- VISUAL --
```

visual mode에서 cursor key는 그대로 사용할 수 있다.  
예를 들어 GG를 누르면 행 끝으로 이동한다.

A11

# visual line mode

```
1304, Yona Yahav, M, 42, MP1
1294, Kebin Robinson, M, 41, CP1
1601, Steven Choi, M, 34, CP3
1314, TW Yoon, F, 46, CP1
1280, Wendy Johnson, 36, CP1, F
1361, Marc Herold, M, 45, MP2
1315, Rina Suzuki, F, 36, MP1
1600, Robert Kim, M, 32, CP3
1297, Rarry Robinson, M, 38, CP2
```

~

-- VISUAL LINE --

5,17

All

# visual block mode

```
1304, Yona Yahav, M, 42, MP1
1294, Kebin Robinson, M, 41, CP1
1601, Steven Choi, M, 34, CP3
1314, TW Yoon, F, 46, CP1
1280, Wendy Johnson, 36, CP1, F
1361, Marc Herold, M, 45, MP2
1315, Rina Suzuki, F, 36, MP1
1600, Robert Kim, M, 32, CP3
1297, Rarry Robinson, M, 38, CP2
```

~

-- VISUAL BLOCK --

8,9

All

특정 컬럼 전체를 삭제할 때 편리하다. (단 컬럼의 길이가 같다면...)  
= 각 컬럼의 길이가 다르다면 recording 기능을 사용하는 것이 좋다.

# visual mode : range

```
1294, Kebin Robinson, M, 41, CP1
1601, Steven Choi, M, 34, CP3
1314, TW Yoon, F, 46, CP1
1280, Wendy Johnson, 36, CP1, F
1361, Marc Herold, M, 45, MP2
1315, Rina Suzuki, F, 36, MP1
1600, Robert Kim, M, 32, CP3
1297, Rarry Robinson, M, 38, CP2
~
~
:'<,>|
```

visual mode에서 colon을 누르면 자동으로 **range**가 설정된다.



# visual block : column editing

ESC를

두번 눌러야만 한다.

❖ CTRL-V , 에디팅(I, A, R, c ...) , ESC\*2

▶ ESC를 2번 타이핑해야만 column editing이 완료된다.

↳ I : insert , A : append

↳ c : change

↳ ~ : switch case

❖ E.g.

▶ 특정 열에 문자열을 삽입하는 경우

▶ 특정 열에 문자열을 교체하는 경우

# Practice - column editing

❖ 아래와 같은 형태로 편집하려면?

▶ 이태리 장인처럼 한줄 한줄 편집하지는 말고...

```
000013-04, Yona Yahav, M, 42, MP1
000012-94, Kebin Robinson, M, 41, CP1
000016-01, Steven Choi, M, 34, CP3
000013-14, TW Yoon, F, 46, CP1
000012-80, Wendy Johnson, 36, CP1, F
000013-61, Marc Herold, M, 45, MP2
000013-15, Rina Suzuki, F, 36, MP1
000016-00, Robert Kim, M, 32, CP3
000012-97, Rarry Robinson, M, 38, CP2
```

# visual mode

❖ gv

▶ previous highlighted text 영역 불러오기

❖ o

▶ highlighted text 블록의 시작, 끝 이동