VI 에디터 사용법

이호석(e-mail: lhs0@kuls.korea.ac.kr, lhs0@sbr.net)

2000년 8월 12일

목 차

1 vi란?

리눅스나 유닉스를 처음 접하는 사람에게 vi는 매우 골치아픈 존재이다. 도스나 윈도우에서 문서 편집기를 써 본 사람에게는 vi는 매우 불편한 편집기로 생각될 것이다. 그러나 내가 아는 한 vi만큼 가볍고 편하게 쓸 수 있는 편집기는 지구상에는 없다! '가볍고'라는 단어를 붙인 이유는 이맥스(emacs)라는 편집기를 쓰는 사람이 항의할 것이 걱정되기 때문이다. 이맥스는 호랑이 담배피던 시절부터[1] 가장 똑똑한 에디터로 군림해 왔다. 그러나 프로그램 크기만도 수십메가에 달하고 설치되어 있지 않은 경우도 많다. 하지만 대부분의 유닉스 계열 시스템이라면 vi는 십중팔구 설치되어 있을 것이다. 따라서 한번 익혀두면 여러모로 편리하게 쓸 수 있다.

vi라는 이름은 '**VI**sual display editor'를 의미한다. vi는 버클리의 어느 천재가 만들었다고 한다. vi를 만들던 시절에는 ed와 같은 라인 에디터가 일반적이었다. 그 옛날 옛적 도스의 'edlin' 이라는 라인 에디터를 써 본 사람이라면 라인 에디터가 얼마나 불편한지 알 것이다. 유닉스처럼 텍스트 에디터와 포매터가 분리된 환경에서는 텍스트 에디터의 비중이 크기 때문에 기능 면에서도 많은 요구가 있게 마련이다. 때문에 텍스트 에디터가 워드프로세서의 기능의 상당 부분을 가지게 되었다. 유닉스에 여러가지 종류가 있듯이 vi도 여러가지 클론이 만들어졌다. 요즘 대부분의 배포판에는 vim이라는 vi의 클론이 포함되어 있다. vim은 완벽하게 한글을 지원하고 원래의 vi의 기능을 충실하게 갖고 있을 뿐만 아니라 여러가지 좀더 편리한 툴들을 제공한다.

2 vi 입문

2.1 실행과 종료, 파일 저장

우선 vi를 실행해 보자. 쉘 프롬프트에서 vi<CR>[2] 를 입력하면 다음과 같은 화면이 보일 것이다. 만약 이미 존재하는 어떤 파일을 편집하고 싶다면 'vi [file]<CR>'을 입력한다.

'~'는 빈 줄, 즉 아무것도 없는 줄이라는 뜻이다. 친절하게도 vim은 실행하자마자 가장 중요한 명령을 알려준다. 바로 ':q<CR>'이다. 한번 실행해 보자. 쉘 프롬프트로 돌아갈 것이다. 이미 'quit'의 약자라는걸 짐작하고 있을 것이다. vi를 종료하는 명령 못지 않게 중요한 명령이 있다. 바로 파일을 저장하는 명령이다. 파일을 저장하기 위해서는 명령 모드에서 ':w [file]<CR>'을 타이프하면 된다. 만약 이미 파일 이름이 지정되어 있다면 그냥 간단히 ':w'만 타이프해도 된다.[3] 또, 파일을 저장한 후 vi를 종료하려면 명령 모드에서 ':wq<CR>' 혹은 'ZZ'를 차례로 타이프하면 된다.

[1] 좀 과장하면 그렇다는 이야기이다.

[2] <CR>은 엔터를 치라는 뜻이다. <, C, R, >을 차례대로 타이프하라는 뜻이 절대로 아니다!

[3] 'write'의 첫 글자이다.

```
~
~
~
~                            VIM = Vi IMproved
~
~                              version 5.6.70
~                          by Bram Moolenaar et al.
~
~                        Vim is freely distributable
~             type  :help uganda<Enter>     if you like vim
~
~             type  :q<Enter>               to exit
~             type  :help<Enter>  or  <F1>  for on-line help
~             type  :help version5<Enter>   for version info
~
~
~

[영어][완성][세벌식]
```

2.2 글 입력

이제 뭔가를 써 넣을 차례다. 그 전에 우선 한 가지를 알아둘 것이 있다. vi가 다른 '평범한' 편집기와 비교할 때 가장 특이한 점은 프로그램의 상태가 '명령 모드'와 '입력 모드'로 구분된다는 점이다. 보통의 편집기는 키보드에서 입력되는 문자가 그대로 문서에 포함될 것이다. 즉, 기본적으로 입력 모드에 있다. 이에 반해 vi는 기본적으로 명령 모드에 있다. 이 상태에서 키보드에서 입력하는 문자 하나하나는 vi에 대한 명령어가 된다. 입력 모드로 들어가는 문자를 입력하면 비로소 그때부터 입력 모드가 되는 것이다. 입력 모드에서 명령 모드로 복귀하려면 '`<ESC>`'키를 누른다. 입력 모드로 들어가는 명령을 요약하면 다음 표와 같다.

a	커서 위치의 다음 칸부터부터 끼워넣기(append)
A	커서가 있는 줄의 끝에서부터 끼워넣기
i	커서 위치부터 끼워넣기(insert)
I	커서가 있는 줄의 맨 앞에서부터 끼워넣기
o	커서 바로 아래에 줄을 만들고 끼워넣기(open line)
O	커서 바로 위에 줄을 만들고 끼워넣기

다음과 같이 적어 보자.

```
iI am a member of KULS!<CR>Are you a member of KULS?<CR>나는 018이다.
```

화면에 다음과 같이 입력될 것이다.

```
I am a member of KULS!
Are you a member of KULS?
나는 018이다.
~
~
```

입력을 마쳤으면 '`<ESC>`'를 눌러 다시 명령 모드로 돌아오자.[4]

[4]만약 현재 상태가 입력 모드인지 명령 모드인지 잘 모른다면 무조건 '`<ESC>`'키를 눌러 일단 명령모드로 돌아온 후 다시 시작하라.

2.3 커서 이동

이제 커서를 이동시켜 보자. 커서 이동에 사용하는 키는 다음과 같다.

h	한칸 왼쪽으로 이동	l	한칸 오른쪽으로 이동
j	한줄 아래로 이동	k	한줄 위로 이동
w	다음 단어의 첫 글자로 이동	W	다음 단어의 첫 글자로 이동
b	이전 단어의 첫 글자로 이동	B	이전 단어의 첫 글자로 이동
e	단어의 마지막 글자로 이동	E	단어의 마지막 글자로 이동
ˆ	그 줄의 첫 글자로 이동	$	그 줄의 마지막 글자로 이동
0	그 줄의 처음으로 이동	<CR>	다음 줄의 첫 글자로 이동
+	다음 줄의 첫 글자로 이동	−	윗줄의 첫 글자로 이동
(	이전 문장의 첫 글자로 이동	)	다음 문장의 첫 글자로 이동
{	이전 문단으로 이동	}	다음 문단으로 이동

단어 단위 이동 명령에서 대문자 명령은 소문자 명령과 약간 차이가 있다. 대문자 명령은 무조건 띄어쓰기 전까지를 한 단어로 취급한다. 다시 말해 `I'm a member of "KULS"`에서 `w, b, e`가 `I, ', m, a, member, of, ", KULS, "` 를 각각 한 단어로 취급하는 데 반해 `W, B, E`는 `I'm, a, member, of, "KULS"` 를 한 단어로 취급한다. 또, vi는 빈 줄이 나오기 전까지를 한 문단으로 취급한다.[5]

2.4 글 수정

작성되어 있는 글을 수정하는 명령은 보통 `c`와 커서 이동 명령이 결합된 형태이다.[6] 정리하면 다음과 같다.

r	커서 위치의 한 글자 교체
R	커서 위치부터 <ESC>를 누를때까지 다른 글자로 교체
s, cl	커서 위치의 글자를 한개를 여러 글자로 교체
ch	커서 바로 앞의 글자를 한개를 여러 글자로 교체
cw	커서 위치의 한 단어를 교체
c0	커서 위치부터 줄의 처음까지 교체
C, c$	커서 위치부터 줄의 끝까지 교체
cc	커서가 있는 줄을 교체
cj	커서가 있는 줄과 그 다음 줄을 교체
ck	커서가 있는 줄과 그 앞줄을 교체

이제 실습을 해보자. 다음과 같이 타이프 해 보자.

```
jjjr1kwwwwcwKorea University Linux Study<ESC>
```

아래와같이 바뀌었을 것이다. 세 글자 왼쪽으로 이동한 후(`jjj`) 커서 위치의 글자를 1로 교체하라(`r1`)는 뜻이다. `r` 명령의 경우 한 글자만 교체하는 명령이므로 교체 후 자동으로 명령모드로 전환된다. 그 다음은 한줄 위로 이동한 후(`k`) 네 단어 오른쪽으로 이동하

[5] 즉, {를 누르면 이전 빈 줄로, }를 누르면 다음 빈 줄로 이동한다.

[6] `c`는 change를 뜻한다.

고(wwww) 커서 위치의 단어를 Korea University Linux Study로 교체하고(cw) 명령모드로 전환하라(<ESC>)는 뜻이다.

```
I am a member of KULS!
Are you a member of Korea University Linux Study?
나는 011이다.
~
~
```

다시 다음과 같이 타이프해 보자.

```
O고려대학교 리눅스 동아리 KULS! Since 1999.<ESC>jkkksthe<ESC>
```

이제 아래와 같이 바뀌었을 것이다. 먼저 커서가 있는 줄의 위쪽에 한 줄을 삽입하고(O) (고려대학교 리눅스 동아리 KULS! Since 1999.)를 적은 다음 아래줄로 가서 오른쪽으로 세 글자 이동한 후(jkkk) 커서 위치의 한 글자를(s) the로 교체하라고 명령한 것이다.

```
I am a member of KULS!
고려대학교 리눅스 동아리 KULS! Since 1999.
Are you the member of Korea University Linux Study?
나는 011이다.
~
~
```

2.5 글 삭제

글을 삭제하는 명령도 수정하는 명령과 거의 비슷하다. 글을 수정하는 명령이 c와 커서 이동 명령이 결합된 형태인것처럼 글을 삭제하는 명령은 d와 커서 이동 명령이 결합된 형태이다.[7] 정리하면 다음과 같다.

x, dl	커서 위치의 글자 삭제
X, dh	커서 바로 앞의 글자 삭제
dw	한 단어를 삭제
d0	커서 위치부터 줄의 처음까지 삭제
D, d$	커서 위치부터 줄의 끝까지 삭제
dd	커서가 있는 줄을 삭제
dj	커서가 있는 줄과 그 다음 줄을 삭제
dk	커서가 있는 줄과 그 앞줄을 삭제

이제 글을 삭제해보자. 다음과 같이 타이프한다.

```
j0wd0wxxxxwwXXXjwwDjdd
```

다음 줄의 처음으로 이동해(j0) 한 단어 오른쪽으로 이동한 후(w) 커서 위치에서부터 줄의 처음까지 지우고(d0) 다시 한 단어 이동한 뒤(w) 커서 위치의 네 글자를 지우고(xxxx) 다시 두 단어 이동한 뒤(ww) 커서 바로 앞의 글자 세개를 지우고(XXX) 다음 줄로 내려가 두 단어 오른쪽으로 이동한 후 거기에서부터 줄의 끝까지 지우고(jwwD) 다음 줄을 모두 지우라(jdd)는 뜻이다. 이제 아래와 같이 바뀌었을 것이다.

```
I am a member of KULS!
고려대학교 리눅스 동아리 KULS! Since 1999.
```

[7] d는 이미 짐작하고 있겠지만 delete의 첫 글자이다.

```
you member Korea University
~
~
```

2.6 복사 & 붙이기

글을 복사하는 명령을 이미 두 가지를 배웠다. 위에서 글 교체와 글 삭제 명령으로 지워진 글은 버퍼에 저장된다. 버퍼에 저장된 글을 끼워넣는 명령은 두 가지가 있다. `p` 를 누르면 현재 커서 위치의 바로 다음에 끼워넣고, `P`를 누르면 현재 커서 위치의 바로 앞에 끼워넣는다.

현재 작성되어 있는 글에 영향을 미치지 않고 글을 복사하는 방법은 `c`나 `d` 대신 `y`를 사용한다는 점을 제외하면 위에서 살펴본 교체나 삭제 방법과 동일하다.[8] 정리하면 다음과 같다.

yw	커서 위치부터 단어의 끝까지 복사
y0	커서 위치부터 줄의 처음까지 복사
y$	커서 위치부터 줄의 끝까지 복사
yy	커서가 있는 줄을 복사
yj	커서가 있는 줄과 그 다음 줄을 복사
yk	커서가 있는 줄과 그 앞줄을 복사
p	커서의 다음 위치에 붙여넣기
P	커서가 있는 위치에 붙여넣기

다음과 같이 입력해보자.

```
kkPjyyjpjdw$pdw$p0dw$p
```

먼저 두 줄 위로 올라가 커서가 앞쪽(위쪽)에 조금 전에 삭제한 내용을 끼워넣고(`kkP`) 다음 줄을 복사해서(`jyy`) 한 줄 아래로 이동한 후 커서 뒤쪽(아래쪽)에 끼워넣은 후(`jp`) 한 줄 아래로 내려가 한 단어를 삭제한 후 그 단어를 그 줄의 맨 끝에 붙여넣고(`jdw$p`) 다시 맨 앞의 한 단어를 삭제한 후 맨 뒤에 붙여넣으라(`0dw$p`)는 뜻이다. 이제 다음과 같이 바뀌었을 것이다.

```
나는 011이다.
I am a member of KULS!
고려대학교 리눅스 동아리 KULS! Since 1999.
I am a member of KULS!
Korea University you member
~
```

2.7 기타

다음은 위에는 해당되지 않지만 많이 사용되는 명령들이다.

---

[8] y는 잡아당기다 라는 뜻의 영어 단어 yank의 첫 글자이다.

u	작업 취소(undo)
U	그 줄에 행해진 작업 모두 취소
Ctrl+r	작업 재실행(redo)
.	조금 전에 했던 명령을 반복
J	현재 줄과 아래 줄을 연결
~	대문자를 소문자로, 소문자를 대문자로 바꿈
%	괄호의 반대쪽 짝으로 이동
Ctrl+l	현재 화면을 지우고 다시 그림
Ctrl+g	파일에 관한 정보를 표시

다음과 같이 입력해 보자.

```
kkk0bbbxxxuuujdw.UkkJ
```

먼저 세 줄 위로 이동해 그 줄의 맨 앞으로 이동한 후 세 단어 오른쪽으로 이동하고(kkk0bbb) 글자 세개를 지웠다가 이를 취소한 후(xxxuuu) 한줄 아래로 내려가 한 단어를 삭제하고(jdw) 다시 한번 한 단어를 삭제하고(.) 그 줄에서 행해진 작업을 모두 취소한 뒤(U) 두 줄 올라가 그 줄과 그 다음 줄을 연결하라(kkJ)는 명령이다. 이제 다음과 같이 바뀌었다.

```
나는 011이다. I am a member of KULS!
고려대리눅스 동아리 KULS! Since 1999.
I am a member of KULS!
Korea University you member
~
```

Ctrl+l 명령은 화면을 다시 나타내는 명령이다. 문서 작성중에 talk 요청이 오거나 혹은 다른 메시지에 의해 화면이 지저분해질 경우 사용하면 편리하다.

지금까지 배운 대부분의 명령 앞에는 숫자 인수를 줄 수 있다. 다음과 같이 타이프해 보자.

```
3j0w2dw2k2yy3P
```

세 줄 아래로 내려간 후 첫 번째 칸으로 이동했다가 한 단어 오른쪽으로 이동하고(3j0w) 두 단어를 지운 후(2dw) 두줄 위로 올라가서(2k) 두 줄을 복사해서(2yy) 세번 붙여넣기 하라(3P)는 뜻이다. 이제 다음과 같이 바뀌었을 것이다.

```
나는 011이다. I am a member of KULS!
고려대리눅스 동아리 KULS! Since 1999.
I am a member of KULS!
고려대리눅스 동아리 KULS! Since 1999.
I am a member of KULS!
고려대리눅스 동아리 KULS! Since 1999.
I am a member of KULS!
고려대리눅스 동아리 KULS! Since 1999.
I am a member of KULS!
Korea member
~
```

이제 기본적인 내용은 다 익혔다. 지금까지 배운 내용만으로도 간단한 문서를 작성하고 편집하는 데에는 문제가 없을 것이다. 이제 vi를 빠져나가자.

```
:q<CR>
```

엇, 그런데 vi가 말을 듣지 않는다. 화면 맨 아래를 보면 다음과 같이 나와 있을 것이다.

```
~
~
No write since last change (use ! to override)
```

음. 파일을 저장하지 않았기 때문에 그냥은 나갈 수 없다는 뜻이다. 강제로 빠져나가려면 !를 사용해야 한다.

`:q!<CR>`

이제 별 탈 없이 vi를 빠져나올 수 있을 것이다.

3 vi 중급 사용자로

일단 vi를 실행한 후 /usr/doc/HOWTO/Installation-HOWTO 라는 파일을 한번 열어보자. 애써 실행한 vi를 빠져나온 다음 다음과 같이

```
lhs0@libretto$ vi /usr/doc/HOWTO/Installation-HOWTO
```

vi를 다시 실행해줘도 되겠지만, 장차 vi의 고수가 될 사람이 이런 구차한 방법을 쓸 수는 없다. 명령 모드에서 다음과 같이 입력하자.

`:e /usr/doc/HOWTO/Installation-HOWTO<CR>`

bash에서처럼 파일 이름을 전부 다 타이프하지 않고 일부만 타이프한 후 <TAB> 키를 누르면 자동으로 완성된다. 다음과 같은 화면이 나올 것이다.[9]

```
The Linux Installation HOWTO
by Eric S. Raymond
v4.15, 20 November 1998

This document describes how to obtain and install Linux software. It
is the first document which a new Linux user should read to get
started.
______________________________________________________________________

Table of Contents

...

"/usr/doc/HOWTO/Installation-HOWTO" [readonly] 1518 lines, 59961 characters
[영어][완성][세벌식]
```

전체 1518줄이다. j키를 눌러 한 줄씩 내려가며 한번 세보자. 농담이다. 설마 정말 이렇게 하는 사람이 있다면 필자는 아주 슬프다.(--;) 화면 맨 아래에 보면 1518 lines 라고 나와 있다. 만약 나오지 않는다면 Ctrl+g를 눌러보자.

3.1 화면 이동

앞 장에서 이미 커서 이동 명령을 배웠다. 그러나 커다란 문서에서 j, k 키만 가지고 돌아다니는 것은 너무 시간이 오래 걸린다. 커다란 문서를 편집할 때 화면 이동에 쓰이는 키는 다음과 같다.

[9]꼭 이 파일이 아니라도 상관없다. 만약 이 파일이 없다면 다른 아무 파일이나 큼지막한 파일을 열어보자.

H	커서를 화면의 맨 위로	z<CR>	현재 줄을 화면의 맨 위로
M	커서를 화면의 중앙으로	z.	현재 줄을 화면의 중앙으로
L	커서를 화면의 맨 아래로	z-	현재 줄을 화면의 맨 아래로
[n]H	커서를 위에서 [n]번째 줄로	[n]L	커서를 아래에서 [n]번째 줄로
Ctrl+u	반 화면 위로 스크롤	Ctrl+b	한 화면 위로 스크롤
Ctrl+d	반 화면 아래로 스크롤	Ctrl+f	한 화면 아래로 스크롤
gg, 1G	문서의 맨 처음 줄로	G	문서의 맨 마지막 줄로
[n]G	[n]번째 줄로 이동		

이제 또 실습을 해 볼 차례다. 먼저 G를 눌러보자. 화면의 맨 마지막 줄로 커서가 이동할 것이다. 이번에는 gg를 입력해 보자. 다시 맨 처음 화면으로 돌아올 것이다. 극과 극으로만 움직이니 별로 재미가 없다. 100G를 한번 입력해 보자. 100번째 줄로 커서가 이동할 것이다. 정말 100번째 줄인지 궁금하면 k키를 한번씩 누르며 몇 번 눌러야 첫 번째 줄 까지 가는지 한번 세보자. 정말 따라하는 사람이 있다면 정말 이 글을 더 이상 쓰기가 싫어질 것 같다.

```
:set number<CR>
```

[10] 를 입력해 보자. 화면 왼쪽에 줄 번호가 표시될 것이다.

```
:set nonumber<CR>
```

[11] 를 입력하면 다시 사라질 것이다.

이제 Ctrl+f를 눌러 한 페이지씩 아래로 내려가 보자. 다시 Ctrl+b를 눌러 원래 위치로 돌아와 보자. 이번에는 H를 눌러 보자. 커서가 화면의 맨 첫 번째 줄로 이동할 것이다. M, L도 한번씩 눌러보자. 이번에는 z<CR>을 눌러보자. H를 눌렀을 때와는 약간 다를 것이다. H를 누르면 화면은 정지된 채 커서만 이동하는 데 반해 z<CR>을 누르면 커서의 위치가 문서에 고정된 채 문서가 스크롤되어 커서가 있는 줄이 첫 번째 줄에 위치하게 된다. z., z-도 한번씩 눌러보자.

3.2 마킹

이제 커다란 문서에서도 마음대로 커서를 움직일 수 있게 되었다. 그런데 만약 어떤 위치에서 작업하다가 잠깐 다른 위치로 가서 뭔가 복사해서 원래의 위치에 붙여넣을 경우를 생각해 보자. 줄 번호를 기억해 두었다가 [n]G 명령으로 그 줄로 가도 되겠지만 똑똑한 컴퓨터를 놔두고 머리를 혹사시킬(?) 필요는 없다. vi는 26개의 마킹을 제공한다. 즉, 어떤 위치에 마크를 해 둔 후 다른 위치로 이동했다가 다시 돌아갈 때 그 마크를 이용할 수 있다. 마킹에 관계된 명령들을 정리하면 다음과 같다.

ma	현재 위치를 ‘a’로 마크		
‘a	마크된 ‘a’로 이동	’a	마크된 ‘a’가 있는 줄의 처음으로 이동
‘‘	직전의 커서 위치로 이동	’’	직전에 커서가 위치하던 줄의 처음으로

[10] 그냥 ‘:set nu<CR>’만 입력해도 된다

[11] 역시 그냥 ‘:set nonu<CR>’만 입력해도 된다.

3.3 여러개의 버퍼 사용

앞에서 버퍼를 사용하는 방법에 대해서 이미 배웠다. 그런데 앞에서 배운 내용으로는 마지막으로 버퍼에 저장된 내용밖에는 사용할 수 없다. vi는 이전 9개까지 버퍼에 저장되었던 내용을 기억한다. 먼저 삭제, 복사, 교체한 글은 1번 버퍼에 저장된다. 그 상태에서 또 다른 글을 삭제, 복사, 교체하면 그 내용이 1번 버퍼에 저장되고 1번 버퍼에 내용은 2번 버퍼로 옮겨진다. 그 상태에서 다시 글을 삭제하거나 복사하거나 교체하면 그 내용이 다시 1번 버퍼에 옮겨지고 1번 버퍼에 있던 글은 2번 버퍼에, 2번 버퍼에 있던 글은 3번 버퍼로 옮겨진다. 이런식으로 총 9개의 버퍼에 내용이 저장된다. `[n]`번째 버퍼에 있는 내용을 붙여넣기 위해서는 다음과 같이 입력하면 된다.

```
"[n]p 혹은 "[n]P
```

즉, 붙여넣기 전에 `"[n]`을 붙여주면 된다. 만약 어떤 버퍼에 저장된 내용을 붙여넣어야 할지 잘 모르겠으면 다음과 같이 해보면 자동적으로 버퍼의 번호를 증가시킬 수 있다.[12]

```
"1pu.u.u.u.
```

위에서 이야기한 9개의 버퍼 외에도 vi에서는 a부터 z까지 이름이 붙은 26개의 버퍼를 더 사용할 수 있다. 만약 현재 줄부터 3줄을 버퍼 'a'에 복사 싶다면 `"a3yy`를 입력한다. 현재 커서의 위치에서 줄의 끝까지 지우면서 그 내용을 버퍼 'b'에 저장하고 싶다면 `"bD`를 입력하면 된다. 버퍼에 있는 내용을 붙여넣는 방법은 앞에서 설명한 9개의 버퍼와 같다. 만약 버퍼 'a'에 있는 내용을 붙여넣고 싶다면 `"ap`를 입력하면 된다.

3.4 패턴 검색 및 교체

패턴을 검색하는 방법은 크게 두 가지가 있다. 먼저 그 줄에서 일치하는 글자를 찾는 방법이다. 만약 현재 줄에서 'a'라는 문자를 찾고 싶다면 `fa`를 입력한다. 다음 'a'를 검색하려면 `;`를 입력하고 다시 이전의 'a'를 검색하려면 `,`를 입력한다. 사실 위에서 배운 한 줄 안에서의 검색은 별로 쓸 일이 없을 것 같다. 대부분의 경우 그냥 눈으로 보고 `h, l` 키나 `w, b` 키로 찾아가는 것이 속편하다는 사람이 많을 것 같다.

보다 많이 쓰이는 검색 방법은 문서 전체에서 특정 패턴을 찾는 방법이다. 문서 전체에서 특정 패턴을 찾는 방법을 정리하면 다음과 같다.

`/[pattern]<CR>`	현재 위치에서부터 아래 방향으로 패턴 검색
`?[pattern]<CR>`	현재 위치에서부터 위쪽 방향으로 패턴 검색
`n`	검색하던 방향으로 계속 패턴 검색
`N`	검색하던 반대 방향으로 계속 패턴 검색

만약 'linux'라는 단어를 검색하고 싶다면 다음과 같이 입력한다.

```
/linux<CR>
```

계속하여 일치하는 패턴을 검색하고 싶다면 `n`을 입력한다. 문서의 맨 마지막 패턴에서 다시 `n`을 누르면 다시 문서의 처음부터 검색을 시작할 것이다. 이 기능을 끄고 싶다면 다음과 같이 입력한다.

```
:set nowrapscan<CR>
```

[12] 직접 꼭 해보자! emacs에서 `Alt+y`를 눌렀을 때와 비슷하게 바뀐다.

[13] 이제 문서의 끝에 도달하면 다시 처음으로 돌아가지 않을 것이다. 다시 켜고 싶다면 다음과 같이 입력한다.

```
:set wrapscan<CR>
```

[14]

검색만 하니 별로 재미가 없다. 이제 특정 패턴을 다른 패턴으로 바꾸어 보자. 어떤 패턴을 다른 패턴으로 바꾸는 방법을 정리하면 다음과 같다.

:s/old/new<CR>	현재 줄의 처음 old를 new로 교체
:s/old/new/g<CR>	현재 줄의 모든 old를 new로 교체
:1,20s/old/new/g<CR>	1부터 20번째 줄까지 모든 old를 new로 교체
:-2,+4s/old/new/g<CR>	커서 2줄 위부터 4줄 아래까지 old를 new로 교체
:%s/old/new/g<CR>	문서 전체에서 old를 new로 교체
:%s/old/new/gc<CR>	문서 전체에서 old를 new로 확인하며 교체
:g/pattern/s/old/new/g<CR>	pattern이 있는 모든 줄의 old를 new로 교체
:g/pattern/s//new/g<CR>	:%s/old/new/g<CR>과 동일

3.5 vi에서 UNIX 명령 실행

다음과 같이 입력해보자.

```
:!bash<CR>
```

앗, 이건... 친숙한 쉘 프롬프트이다. 만약 vi를 사용하다가 잠깐 쉘 프롬프트로 빠져나와야 할 경우 사용한다. 다시 vi로 돌아가고 싶으면 exit<CR>를 입력한다. :q<CR>로 vi를 빠져나온것과는 달리 현재 커서의 위치나 버퍼의 내용이 그대로 보존되므로 잠깐 동안 쉘 프롬프트로 빠져나왔다가 금방 다시 vi로 돌아가야 할 경우 편리하게 사용할 수 있다.[15]

만약 한 개의 명령어만 실행할 경우 좀더 편리한 방법이 있다.

```
:![command]<CR>
```

필자는 현재 latex과 vi로 이 문서를 작성하고 있는데 문서를 미리보기 할 때마다 vi를 빠져나가서 봐야 한다면 무척 불편할 것이다. 다행히 다음 명령어로 vi를 빠져나가지 않고 미리보기를 할 수 있다.

```
:!latex %; xdvi vi-seminar.dvi<CR>
```

'latex' 이라는 명령으로 현재 편집중인 파일인 vi-seminar.tex 파일을 컴파일한 다음 생성된 vi-seminar.dvi 파일을 'xdvi'라는 프로그램으로 열어보라는 명령이다. 매번 이렇게 타이프 할 필요는 없다. :을 누른 후 위쪽 화살표 키 혹은 Ctrl+p 키를 누르면 이전에 사용했던 명령이 나올 것이다. 이전에 사용했던 명령을 찾아 단순히 엔터 키만 쳐 주면 된다.

[13]:set nows<CR>만 입력해도 된다.

[14]:set ws<CR>만 입력해도 된다.

[15]그런데 주의할 점이 하나 있다. 이렇게 vi를 빠져나온 후 다시 vi를 실행해서는 안된다. 별 문제야 없겠지만 같은 파일을 다시 연다면 문제가 생길 수 있다.

3.6 끝줄 모드의 사용

이미 끝줄 모드에서 사용하는 명령어를 몇 가지 배웠다. 명령 모드에서 ':'를 입력하면 아래와 같이 화면 맨 아래에서 입력하는 글자들이 보일 것이다.

```
~
~
~
:wq
[영어][완성][세벌식]
```

이와 같은 모드를 '끝줄 모드' 혹은 'ex 모드'로 따로 구분해서 부르기도 한다. 끝줄 모드에서 많이 사용하는 명령어를 정리하면 다음과 같다. 특정 패턴을 다른 패턴으로 교체하는 방법은 이미 위에서 정리하였으므로 다시 하지 않겠다.

:q<CR>	vi를 종료함
:w<CR>	편집중인 문서를 저장
:w [file]<CR>	편집중인 문서를 [file]로 저장
:w >> [file]<CR>	편집중인 문서를 [file]에 덧붙여서 저장
:e [file]<CR>	[file]을 불러옴
:e#<CR>	이전에 편집하던 파일을 불러옴
:e%<CR>	현재 파일을 다시 불러옴, 즉 저장하지 않은 작업 취소
:r [file]<CR>	[file]을 커서위치에 끼워넣기
:set [option]<CR>	[option]을 켜기
:set [nooption]<CR>	[option]을 끄기
:![command]<CR>	[command] 실행
:r ![command]<CR>	[command] 실행 결과를 끼워넣기

예를 들어 현재 위치에 날짜와 시간을 삽입하고 싶으면

`:r !date<CR>`

를 입력한다. 만약 phone 이라는 파일을 정렬하여 삽입하고 싶으면

`:r !sort phone<CR>`

를 입력하면 된다. 앞에서 이미 'set'명령에 사용할 수 있는 옵션을 몇 가지 배웠다. 'number'와 'wrapscan'이 생각나지 않는다면 앞장으로 되돌아가 확인하기 바란다. 다른 유용한 옵션들은 다음 장에서 설명하겠다.

4 vi의 고수로

지금까지 배운 내용만 사용하더라도 이제 vi를 다른 범상한 편집기와는 비교도 할 수 없을 만큼 편리하게 사용할 수 있다. 이 정도만으로도 친구에게 자랑할 수 있을 것이다. 그러나 이 정도만으로는 아직 '나는 vi의 고수다!' 라고 말하기는 좀 부끄러운 감이 있다. 모든걸 정리하는 마음으로 앞에서 열어 두었던 큼지막한 파일을 닫고 새 파일을 열자. 만약 vi를 종료했다가 다시 실행하는 사람이 있다면 정말 필자는 삶이 허무해진다! 다음과 같이 타이프한다.

`:e newfile<CR>`

4.1 상용구 사용

아래아 한글에 '상용구'라는 기능이 있다. 이 기능을 사용하는 사람이 얼마나 될지는 잘 모르겠지만, 아무튼 vi에도 이런 기능이 있다. 다음과 같이 타이프해 보자.

```
:ab kuls Korea University Linux Study<CR>
```

이제 입력 모드로 전환해 'I'm a member of kuls.'를 입력하자. 'kuls'를 타잎하는 순간 'Korea University Linux Study'로 바뀔 것이다. 등록된 상용구를 해제하는 방법은 다음과 같다.

```
:unab kuls<CR>
```

등록된 상용구를 보여주는 명령은 다음과 같다.

```
:ab<CR>
```

만약 여러 줄을 상용구로 지정하고 싶다면 다음과 같이 한다.[16]

```
:ab kuls2 Korea University^MLinux Study<CR>
```

4.2 매크로 사용

vi의 특징중 한 가지는 있을법한 기능은 다 있다는 것이다. 아래아 한글에서 '매크로'를 이용하는 사람이 있을지 모르겠다. vi에서도 매크로를 이용할 수 있다. 다음과 같이 타이프 해 보자.[17]

```
:map #2 dwelp<CR>
```

이제 다음 문장의 the에서 <F2>키를 눌러보자.

```
You can the type letter.
```

아래와 같이 바뀔 것이다.

```
You can type the letter.
```

매크로에서는 <CR>, <ESC>등 모든 키 입력을 포함시킬 수 있다. 매크로에서 <CR>키를 입력하려면 Ctrl+v와 Ctrl+m을 연속으로 타이프한다. <ESC>키를 입력하려면 Ctrl+v를 누른 후 Ctrl+[키를 누른다.

vi에서는 명령 모드에서 사용하는 매크로와 입력 모드에서 사용하는 매크로를 따로 지정할 수 있다. 입력 모드에서 사용하는 매크로는 다음과 같이 지정한다.

```
:map!  x sequence<CR>
```

위의 예에서는 x를 sequence로 정의한다. 매크로를 해제할때는 다음 명령을 이용한다.

```
:unmap!  x<CR>
```

매번 이렇게 매크로를 지정해 주는 것은 귀찮은 일이다. vi를 실행할 때마다 매크로가 실행되어 있게 하려면 홈 디렉토리의 .exrc 파일에 넣어두면 된다. 다음은 LaTeX을 자주 사용하는 필자의 .exrc파일의 일부이다.

[16] ^M을 타이프할때는 먼저 Ctrl+v를 누른 후 이어서 Ctrl+m을 누른다.
[17] '#2'는 <F2>키를 의미한다. 그냥 '<F2>' 라고 쓰면 되는 경우도 있고 안 되는 경우도 있다.

```
" <F9>를 누르면 선택된 부분만 컴파일하여 xdvi로 보여줌
map <F9> ^[:1,?^\\begin{document}$?w! _region_.tex^M;;R1
map ;;R1 :'<,'>w >> _region_.tex^M;;R2
map ;;R2 :/^\\end{document}$/w >> _region_.tex^M;;R3
map ;;R3 :!latex _region_.tex; xdvi -expert _region_.dvi; rm _region_.???^M^M

" <F10>을 누르면 현재 편집중인 문서를 컴파일하여 xdvi로 보여줌
map <F10> :w^M:!latex %; xdvi -expert %<.dvi^M^M

" <F11>을 누르면 컴파일된 문서를 ps로 출력하여 gv로 미리보기
map <F11> :!dvips -o %<.ps %<.dvi; gv %<.ps^M^M

" <F12>를 누르면 변환된 포스트스크립트 문서를 pdf로 변환
map <F12> :!ps2pdf %<.ps %<.pdf&^M^M

" 레이텍 문서 전처리부를 자동으로 만들어주는 매크로
map ;tex i\documentclass[a4paper]{article}^[;;T1
map ;;T1 o\usepackage{hangul,a4,indentfirst}^M\setlength{\parindent}{1em}^[;;T2
map ;;T2 o\addtolength{\hoffset}{-0.5cm}^M\addtolength{\textwidth}{1cm}^[;;T3
map ;;T3 o\kscntformat{section}{}{}^M\linespread{1.3}^M^[;;T4
map ;;T4 o\begin{document}^M^M\end{document}^[ki

" 글꼴에 관계된 매크로
map ;tt i\texttt{^[Ea}^[B
map ;sf i\textsf{^[Ea}^[B
map ;bf i\textbf{^[Ea}^[B
map! ;tt \texttt{
map! ;sf \textsf{
map! ;bf \textbf{

" 환경(Environment)에 관계된 매크로
map! ;itemize \begin{itemize}^M\end{itemize}^[O \item
map! ;enumerate \begin{enumerate}^M\end{enumerate}^[O \item
map! ;description \begin{description}^M\end{description}^[O \item[
map! ;equation \begin{equation}^M\end{equation}^[O \label{eq:}^[i
map! ;figure \begin{figure}[!hbp]^M\end{figure}^[O

" 특수문자 입력을 간편하게 하기 위한 상용구
ab ;\ $\backslash$
ab ;~ \textasciitilde
```

4.3 vi 설정

vi의 설정 파일은 /etc/vimrc 파일과 홈 디렉토리의 .vimrc 파일이다. 만약 GUI 버전의 vi를 사용한다면 /etc/gvimrc 파일과 .gvimrc파일로 설정할 수 있다. 앞에서 이미 vi를 설정하는 명령을 배웠고[18], 몇 가지 옵션들도 이미 살펴보았다. 아래에서 다른 몇 가지 옵션들을 필자의 .vimrc 파일을 예로 들어 설명하겠다.

```
set autoindent      " 자동으로 들여쓰기를 한다.
set cindent         " C 프로그래밍을 할때 자동으로 들여쓰기를 한다.
set smartindent     " 좀더 똑똑한 들여쓰기를 위한 옵션이다.
set textwidth=79    " 만약 79번째 글자를 넘어가면 \
set wrap            " 자동으로 <CR>를 삽입하여 다음 줄로 넘어간다.
set nowrapscan      " 검색할 때 문서의 끝에서 다시 처음으로 돌아가지 않는다.
```

[18] 바로 :set [option] 명령이다.

```
set nobackup         " 백업 파일을 만들지 않는다.
set visualbell       " 키를 잘못눌렀을 때 삑 소리를 내는 대신 번쩍이게 한다.
set ruler            " 화면 우측 하단에 현재 커서의 위치(줄,칸)를 보여준다.
set tabstop=4        " Tab을 눌렀을 때 8칸 대신 4칸 이동하도록 한다.
set shiftwidth=4     " 자동 들여쓰기를 할때 4칸 들여쓰도록 한다.

" 아래는 한글을 제대로 처리하기 위해 필요하다.
if $LANG[0] == 'k' && $LANG[1] == 'o'
  set fileencoding=korea
endif

"터미널이 xterm-debian 혹은 xterm-xfree86일 경우 컬러를 사용한다.
if &term =~ "xterm-debian" || &term =~ "xterm-xfree86"
  set t_Co=16
  set t_Sf=^[[3%dm
  set t_Sb=^[[4%dm
  set t_kb=^H
  fixdel
endif

"문법 강조 기능을 사용한다.
if has("syntax")
  syntax on            " Default to no syntax highlightning
endif
```

4.4 vim의 특별한 기능

vi의 클론인 vim에는 원래의 vi에는 없지만 편리하게 사용할 수 있는 많은 기능들이 있다. 문법 강조 기능[19]도 원래의 vi에는 없는 기능이다. vim의 기능 중 'Visual Selection' 이라는 기능이 있다. 명령 모드에서 v를 누른 후 커서를 움직여 보자. 아래아 한글의 '블럭' 처럼 역상으로 움직인 범위에 있는 글들이 선택될 것이다. 이 상태에서 y, c, d 키를 눌러 선택된 범위의 글들을 복사, 교체 또는 삭제할 수 있다.

프로그래밍에 편리하게 사용할 수 있는 기능으로 탭을 끼워넣거나 취소하는 기능이 있다. 만약 커서가 있는 줄에서부터 5줄에 탭을 끼워넣으려면 다음과 같이 타이프한다.

`5>>`

이를 취소하려면 다시 다음을 타이프한다.[20]

`5<<`

Visual Selection 기능과 결합해서 사용하면 좀더 편리하게 사용할 수 있다. v를 눌러 들여쓰기 할 범위를 선택한 다음 >를 타이프하면 선택된 범위가 들여쓰기 될 것이다.

4.5 정규식(Regular Expression)

vi의 검색 및 교체 기능에서 '정규식(Regular Expression)'이라는 특별한 표현을 사용할 수 있는데 이를 사용하면 보다 효과적인 검색/교체를 할 수 있다. 정규식은 vi 뿐만 아니라 다른 명령어에서도 많이 쓰이므로 알아두면 편리하다. 그러나 정규식에 대한 내용만

[19]vim으로 C 소스를 열어보면 예쁘게 색깔이 칠해져 나올 것이다. 이것을 문법 강조(syntax highlighting) 기능이라고 한다.

[20]물론 그냥 u를 입력해도 된다. 그러나 여기서는 어떻게 동작하는지 보자는 것이므로 '5<<'를 입력한다.

으로도 1시간 강의는 충분히 할 수 있다. 여기서는 간단한 사용법만을 이야기하고 넘어가겠다.

우선 정규식에서 많이 쓰이는 기호는 다음과 같다.

^	줄의 처음을 나타낸다.
$	줄의 끝을 나타낸다.
.	아무 글자나 한 글자를 나타낸다.
[...]	[과] 사이에 있는 글자 중 하나
*	* 앞의 내용이 0번 이상 반복됨
\{min,max\}	min 이상 max 이하의 횟수만큼 반복됨

예를 들어 줄의 맨 처음에 나오는 'linux'만 검색하고 싶다면 다음과 같이 입력한다.

```
/^linux<CR>
```

만약 3~5자리로 이루어진 모든 숫자를 검색하고 싶다면 다음과 같이 입력한다.

```
/[0123456789]\{3,5\}<CR> 혹은 /[0-9]\{3,5\}<CR>
```

만약 'ab'로 시작하는 소문자로 되어 있는 모든 단어를 삭제하려면 다음과 같이 입력한다.

```
:%s/ab[a-z]*//g<CR>
```

5 맺음말

vi의 고수가 되기 위해서 위의 내용을 무조건 외우려고 하지 말라. j 키가 무슨 역할을 하느냐고 필자에게 물어본다 해도 곧바로 대답할 수 없을 것이다. 머리로 외우려고 하지 말고 감으로 익혀야 한다. 한꺼번에 모든것을 익히려 하지 말고 일단 어떤 어떤 기능이 있다는 것 정도만 알아두라. 그리고 직접 문서를 편집하면서 그때그때 사용 방법을 찾아보자. 처음에는 좀 귀찮겠지만, 그러는 사이 실력이 늘 것이다. 열심히 연습하면 얼마 지나지 않아 윈도우용 vi를 찾아다니게 될 것이다. 그러면 성공이다! vi의 고수가 되자!