

DEVELOPMENT ENVIRONMENT 1

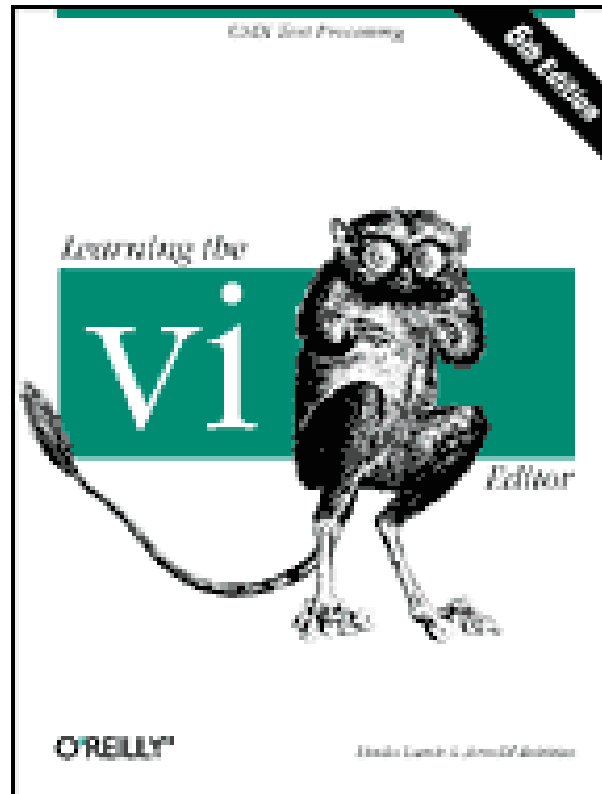
VI EDITOR

Jo, Heeseung

Suggested Reading for vi

If your really need to know about vi,

- Try "Learning the vi Editor" from Oreilly.Com



What is vi?

The `visual editor` on the Unix

Before vi the primary editor used on Unix was the line editor

- User was able to see/edit only one line of the text at a time

The vi editor is not a text formatter (like MS Word, Word Perfect, etc.)

- You cannot set margins
- Center headings
- Etc. ...

vi History

Although other stories exist,
the true one tells that vi was originally
written by [Bill Joy](#) in 1976

Who is Bill Joy you ask?

- He co-founded Sun Microsystems in 1982 and served as chief scientist until 2003

Joy's prowess as a computer programmer is legendary, with an anecdote that he wrote the vi editor [in a weekend](#)

- Joy denies this assertion



Some vi Ports

All Unix OS's

Macintosh

MS-Dos

Atari

Windows 3.x

Amiga

Windows 9x/2k/NT/XP

OpenVMS/Alpha

OS/2

OpenVMS/VAX

Blah blah blah ...

Characteristics of vi

The vi editor is:

- A very powerful
- But at the same time it is cryptic
- It is hard to learn, specially for windows users

The best way to learn vi commands is to use them

So practice... practice... practice...

vim equals vi

The current iteration of vi for Linux is called vim

- vi improved
- <http://www.vim.org>



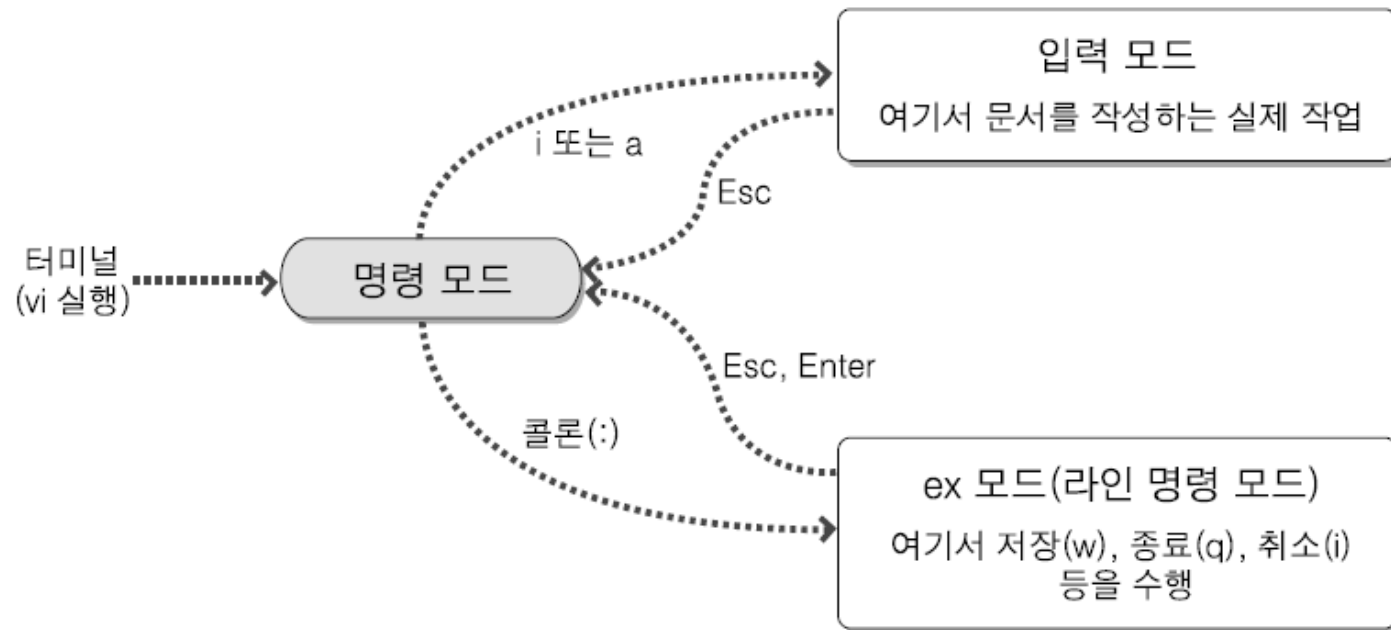
vi edit의 모드

Command mode

- Text의 위치나 편집 가능하게 함

Input mode

- 사용자가 text를 작성할 수 있는 모드



How to exit from vi (command mode)

`:q <enter>`

- To exit, if you have not made any changes to the file

`:q! <enter>`

- The forced quit, it will discard the changes and quit

`:wq <enter>`

- For save and Exit

`:x <enter>`

- Same as above command

`ZZ`

- For save and Exit (Note this command is uppercase)

The `!` Character forces over writes, etc. `:wq!`

Moving Around

You can move around only when you are in the command mode

Arrow keys usually works (but may not)

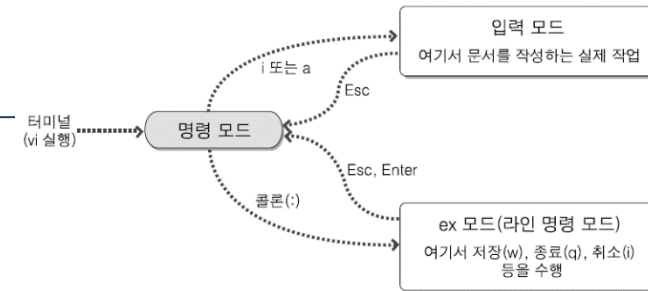
The standard keys for moving cursor are:

- h - for left
- l - for right
- j - for down
- k - for up

h k
 l
 j

Entering text

To enter the text in vi you should first switch to input mode



- To switch to input mode there are several different commands
- **a** - Append mode places the insertion point after the current character
- **i** - Insert mode places the insertion point before the current character
- **I** - places the insertion point at the beginning of current line
- **o** - is for open mode and places the insertion point after the current line
- **O** - places the insertion point before the current line
- **R** - starts the replace(overwrite) mode

Editing text

x - deletes the current character

d - is the delete command but pressing only d will not delete anything you need to press a second key

- dd - deletes the current line
- dw - deletes to end of word
- d0 - deletes to beginning of line

There are many more keys to be used with delete command

Undo and repeat command

- u - undo the changes made by editing commands
- . - repeats the last edit command

Copy, cut and paste in vi

`yy / dd` - (yank) copy / cut current line to buffer

`nyy / ndd` - where n is number of lines

`p` - paste the yanked lines from buffer to the line below

`v` - visual mode

`y` - yank

`P` - paste the yanked lines from buffer to the line above

(the paste commands will also work after yank)

Jump

gg - jump the first line of file

G - jump the end of file

nG - jump to the line (where n is line number)

Search / Replace

`/asdf` - search forward for 'asdf'

- To move forward (down), type `n`
- To move backward (up), type `N`

`:%s/asdf/xyz/g` - replace 'asdf' by 'xyz'

Everything in a Slide

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc
normal mode

~ toggle case	! external filter	@. play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence	) end sentence	"soft" bol down	+ next line
\. goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto ³ format
Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank ^{1,3}	u undo	i insert mode	o open below	p paste after ¹	[misc	]	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	. ex cmd line	" reg. ¹ spec	bol/ goto col	
a append	s subst char	d delete ^{1,3}	f find char	g extra ⁶ cmds	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	' goto mk. bol	\. not used!	
Z quit ⁴	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- ³ indent	> indent ³	? find (rev.)			
Z extra ⁵ cmds	X delete char	c change ^{1,3}	V visual mode	b prev word	n next (find)	m set mark	, reverse t/T/f/F	. repeat cmd	/ find			

motion	moves the cursor, or defines the range for an operator
command	direct action command, if red , it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input
q.	commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: quux(foo, bar, baz);
WORDS: quux(foo, bar, baz);

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important comands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

유용한 vi environment -> .vimrc

au FileType * setl fo-=cro

- 자동 주석

syntax on / off

set number / nonumber

set wrap / nowrap

set hlsearch

set ignorecase

set ts=4

au BufReadPost * if line("'\"")|execute("normal `\"")|endif

- 파일 오픈시 편집하던 위치로 이동

** 파일 작성시 vi를 사용

- copy/paste 기능은 사용금지
- 반드시 타이핑과 편집 기능을 이용하여 작성

1. 아래의 내용을 가지는 7years.txt 파일을 작성

```
I hope my children come and visit, once or twice a month
Soon, I'll be 60 years old, will I think the world is cold?
Or will I have a lot of children who can warm me?
Soon, I'll be 60 years old
Soon, I'll be 60 years old, will I think the world is cold?
Or will I have a lot of children who can hold me?
Soon, I'll be 60 years old
Once, I was seven years old, my mama told me
"Go make yourself some friends, or you'll be lonely"
Once, I was seven years old
Once, I was seven years old
```

2. 아래의 내용을 가지는 599.c 파일을 작성

```
#include <stdio.h>
int main()
{
    printf("5*1 = 5\n");
    printf("5*2 = 10\n");
    printf("5*3 = 15\n");
    printf("5*4 = 20\n");
    printf("5*5 = 25\n");
    printf("5*6 = 30\n");
    printf("5*7 = 35\n");
    printf("5*8 = 40\n");
    printf("5*9 = 45\n");
}
```

GCC 기본 사용법

프로그램 실행법

GCC Option(1)

-o

- 출력(output) 화일명을 정하는 옵션
- `gcc -o hello hello.c`
hello.c라는 파일을 hello라는 실행파일로 만듦

-l

- 필요 라이브러리를 지정

-g

- 컴파일된 오브젝트파일에 디버깅 코드를 추가
- 추후 gdb를 사용할 때 꼭 필요
- `gcc -g -o hello hello.c`

GCC Option(2)

-O

- 코드를 최적화(optimization)
- -O2는 가장 많이 최적화
- -O0라고 하면 최적화를 하지 않음
- 기본값: -O1 (시스템 마다 상이함)

-I

- `#include` 문장에서 지정한 헤더 화일이 들어있는 곳을 정하는 옵션
- `gcc -c source.c -Iinclude`
-I를 붙여씀

-L

- 그 라이브러리가 어느 디렉토리에 있는지 알려 줌

Ex: vi로 c code를 작성

실습목표

- vi를 이용하여, 99단을 출력하는 프로그램을 작성
- main.c 에서는 printgugu 함수를 call
- gugu.c 에서는 printgugu를 구현하여 99단을 출력
- Compile: `gcc -o gugu.out main.c gugu.c`
- `./gugu.out` 으로 실행

main.c

```
main() {  
  
    printgugu()  
  
}
```

gugu.c

```
printgugu() {  
  
    do... print gugu  
  
}
```

Ex: vi로 c code를 작성

실습목표

- vi를 이용하여, $1^2 + 2^2 + 3^2 + \dots + n^2$ 을 출력하는 프로그램을 작성
- main.c 에서는 n을 입력받고 calculate 함수를 call
- calc.c 에서는 calculate를 구현하여 결과를 리턴
- Compile: `gcc -o nn.out main.c calc.c`
- `./nn.out` 으로 실행

main.c

```
main() {  
  
    calculate(n)  
  
}
```

calc.c

```
calculate(n) {  
  
    do... something  
  
}
```

Ex: vi로 c code를 작성

실습목표

- 삼각형 각변의 길이 3개를 입력 받아 삼각형이 구성될 수 있는지 아닌지 판별하는 프로그램을 작성하시오.
- Input : 4 4 6
- >> 삼각형을 만들 수 있습니다.
- Input : 9 3 4
- >> 삼각형을 만들 수 없습니다.