[vim 으로 diff 가 실패할 때]

연습 중

명령모드에서 Ctrl+D 하면 autocomplete
Ctrl+Q : Vertical Selection.
R : overwrite mode
:reg, “xyy, “xp : “x 레지스터에 저장해두고 꺼내어 붙여넣기

환경 설정 파일 **vimrc** 파일을 두는 위치

`:help vimrc` 라고 하면 각 OS별 vimrc 의 위치를 알 수 있다.

`:echo $HOME` 라고 하면 현재의 $HOME 위치를 알 수 있다.

개인 설정 디렉토리의 위치

`:echo &rtp` 를 입력하면 알 수 있다.

나의 **VIM** 설정

https://github.com/zelon/dotfiles

위의 옵션 중 **hls** 는 **highlighted-search** 이다. 끄려면 **nohls**

Plugin - 플러그인

플러그인은 공식 홈페이지인 http://www.vim.org 에서 ***.vim** 파일을 받아서 **vim** 밑의 **plugin** 폴더에 넣으면 작동되는 것이 기본.

OS	플러그인 폴더 위치
linux/unix/mac	~/.vim
windows	~/vimfiles

```
~/vimfiles
   +  doc
   +  plugin
   +  syntax
```

위와 같은 형태로 구성된다. NERDTree.zip 같은 경우에는 압축을 푼 후에 해당 위치에 덮어쓰면 된다. :NERDTree 라고 하면 해당 플러그인이 동작하며, :NERD 까지 친 후에 <CTRL+D> 를 누르면 나머지 명령들을 볼 수 있다.

snipMate

for 만 입력하면 for 문 구조를 빠르게 만들어주거나, html 을 누르고 tab 을 누르면 전체 구조를 잡아준다

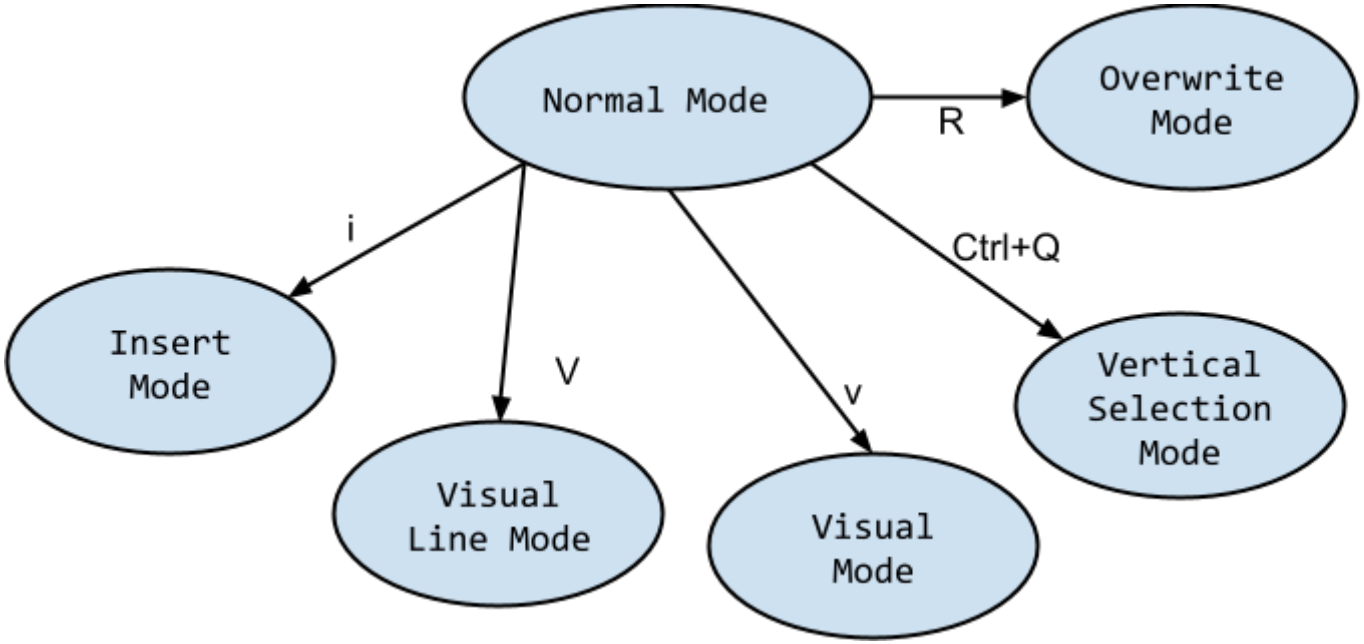
matchIt

html, xml 에서 해당 태그의 짝으로 이동한다. % 기능의 확장판

vundle - 플러그인 매니저

https://github.com/gmarik/vundle

Mode 바꾸기

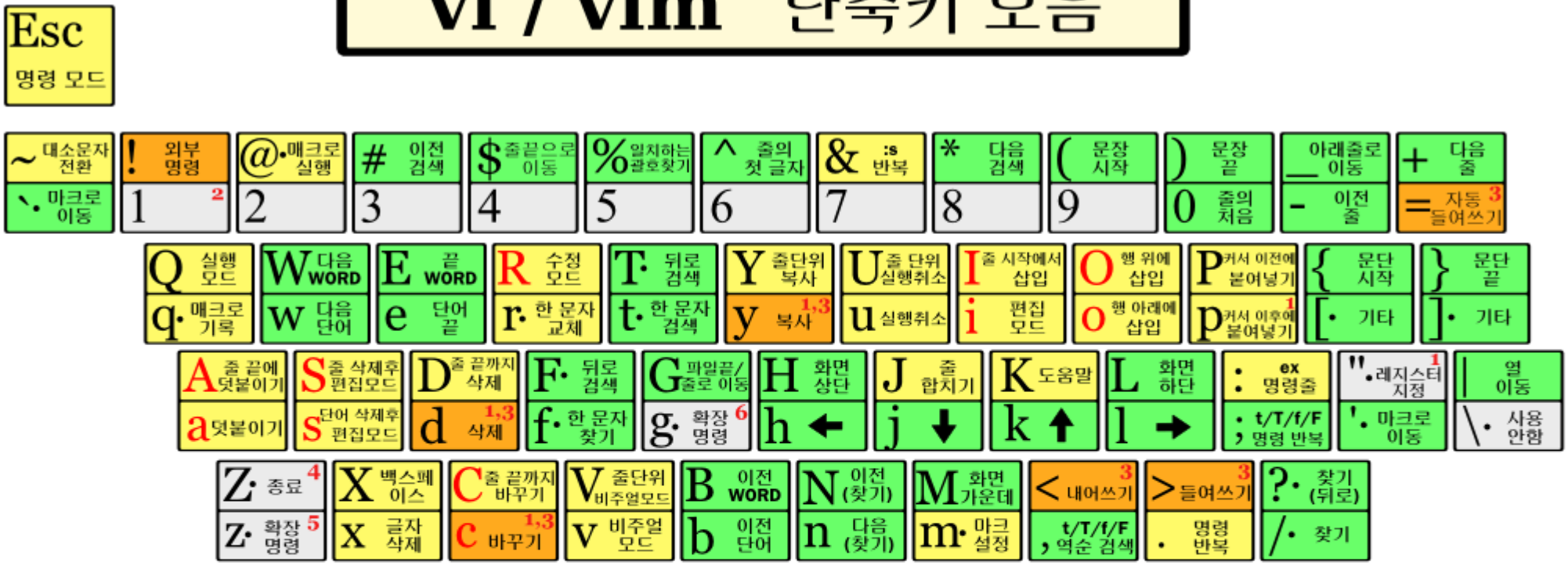


단축키

출처 : http://kldp.org/node/102947

version 1.1
April 1st, 06

vi / vim 단축키 모음



동작 커서를 이동하거나, 연산자가 동작할 범위를 지정합니다.

명령 바로 동작하는 명령, 빨간색은 편집 모드로 변경됩니다.

연산자 이동 관련 문자(숫자나 커서 이동)와 함께 사용하여야 하며, 커서의 위치부터 목적지까지 연산합니다.

확장 특별한 키 함수로, 추가적인 키 입력이 필요합니다.

q. 입력후 (숫자를 제외한 .으로 끝날수 있는) 글자를 입력하여야 합니다.

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)

단축키 맵핑하기

단축키 설정	내용	단축키 해제
nmap key command	일반 모드(normal)에서의 단축키	nunmap
imap key command	입력(insert) 모드에서의 단축키	iunmap
vmap key command	visual mode	vunmap
cmap key command	명령행 모드	cunmap

이벤트 명령

:autocmd BufRead *.cpp colo slate	새로운 파일을 열 때 *.cpp 파일이면 colo slate 명령 실행

녹화하기

녹화 시작	q{reg}
녹화 종료	(녹화중에)q
등록된 녹화 실행	@{reg} 예를 들어, @a 를 하면 a register 에 등록된 것 실행
범위에 녹화 실행	:%normal! @a % 는 전체 범위 normal 은 normal mode !는 문제가 생겨도 계속 진행을 뜻함
마지막 실행을 다시 실행	@@
녹화 편집	“{reg}p 한 후에 편집해서 다시 “{reg}y$ 로 register 를 다시 설정하면 됨

ctag 사용 **-** 소스 탐색하기

윈도우에서 **ctag** 설치

http://ctags.sourceforge.net/ 에서 다운로드하여, 압축을 푼 후 ctags.exe 파일만 C:\Windows 에 넣는다.

쉘에서 **ctags -R**하여 모든 소스에 대해 인덱스 생성

vi를 실행하고 검색할 단어를 :ta tag로 지정하든지, 단어가 있는 곳에 커서를 둔다.

해당함수(오브젝트)가 정의된 곳으로 가기: Ctrl +]

되돌아가기: Ctrl + t

창 왼쪽에 현재 파일의 분석 보기 **- taglist**

http://www.vim.org/scripts/script.php?script_id=273 에서 **taglist plugin** 을 다운받아서, 윈도우는 **C:\Program Files (x86)\Vim\vimfiles** 이런 곳에 넣고, 리눅스는 **.vim** 아래에 넣는다.
그리고 파일을 열고, **:Tlist** 를 치면 왼쪽에 나온다.

숫자 증가/감소

<Ctrl+A> 를 누르면 증가, **<Ctrl+X>** 를 누르면 감소이지만, 윈도우 단축키와 충돌이므로 아래의 명령을 이용하자
:nunmap <C-A>

특수 문자 입력

^M 을 입력하거나 치환하려면, **<Ctrl+Q>**, **<Ctrl+M>** 을 입력하자

endline, 윈도우 **<->** 리눅스 파일 형식 전환

:set fileformat=dos|unix|mac

python 으로 **vim plugin** 만들기

http://www.terminally-incoherent.com/blog/2013/05/06/vriting-vim-plugins-in-python/

~/vimfiles/plugin 에 **helloworld.py helloworld.vim** 을 만들고, 아래의 파일을 입력한 후 **:ZELON** 해보자

```
# helloworld.py
print(“hello zelon”)
```

```
“ helloworld.vim

if !has(‘python’)
  finish
endif

function! HelloWorld()
  pyfile helloworld.py
endfunc

command! ZELON call HelloWorld()
```

Multihighlight - 여러 단어 하이라이트 하기

```
:hi 해서 어떤 그룹들이 있는지 확인 한 후
:let m = matchadd(“GroupName”, “MyWord”) 로 추가
:echo getmatches() 에서 지울 id 를 찾아보고
:call matchdelete(3) 로 해당 id 만 지우거나
:call clearmatches() 로 전체를 지운다
```

Custom Syntax highlighting 개인 문법 강조하기

만약 prop 라는 확장자를 가지는 파일에 대해서 특정 문법 강조를 하고 싶다면,

먼저 기존의 문법 파일을 구하자. 윈도우의 경우 C:\Program Files\Vim\vim73\syntax 에서, 리눅스의 경우 /usr/share/vim/vim73/syntax 에서 기존의 문법 파일들(*.vim)을 구할 수 있다. 익숙한 cpp.vim 을 선택해보자.

이 파일들을 참조해서 ~/vimfiles/syntax/Prop.vim 파일들 만들자.

Prop.vim 파일을 vimfiles/syntax 에 넣고, vimrc 파일에, au BufNewFile,BufRead *.prop setf prop 하거나, setf prop 대신에 so xxx/xxx/a.vim 으로 하면 사용자 다렉토리에 Prop.vim 파일을 위치시킬 수 있다.

혹시나 적용이 잘 안될 경우, set ft 하면 현재 파일 타입을 알 수 있다.

아래를 참고해서 *.vim 을 만들자

```
syn keyword cppStatement      new delete this friend using
syn keyword cppAccess         public protected private
syn keyword cppType           inline virtual explicit export bool wchar_t
syn keyword cppExceptions     throw try catch
syn keyword cppOperator       operator typeid
syn keyword cppOperator       and bitor or xor compl bitand and_eq or_eq xor_eq not not_eq
syn match cppCast       "\<\(const\|static\|dynamic\|reinterpret\)_cast\s*<"me=e-1
syn match cppCast       "\<\(const\|static\|dynamic\|reinterpret\)_cast\s*$"
syn keyword cppStorageClass   mutable
syn keyword cppStructure      class typename template namespace
```

```
syn keyword cppBoolean        true false
syn keyword zelonStatement zelon zelonion

" Default highlighting
if version >= 508 || !exists("did_cpp_syntax_inits")
  if version < 508
    let did_cpp_syntax_inits = 1
    command -nargs=+ HiLink hi link <args>
  else
    command -nargs=+ HiLink hi def link <args>
  endif
  HiLink cppExceptions        Exception
  HiLink cppOperator          Operator
  HiLink cppStatement         Statement
  HiLink cppType        Type
  HiLink cppStorageClass      StorageClass
  HiLink cppStructure         Structure
  HiLink cppBoolean           Boolean
  HiLink cppConstant          Constant
  delcommand HiLink
endif

highlight zelonStatement guibg=Red guifg=#00ff00
```

리눅스별 **vim** 기능 설치

```
rpm : yum install vim-enhanced
```

apt : apt-get install vim
mac : vim.org 에서 cocoa 버젼 설치. cocoa 버젼이 맥UI 이니깐 다른 거 받고 싶으면 다른 것도 해보셈~

파일 열기 및 저장

:e filename	해당 파일 열기
:e .	현재 폴더 목록 열기 여기서 엔터로 파일을 열 수 있다
:files	열려있는 파일(버퍼) 목록 보기 % : 현재 편집 중인 버퍼 # : 바로 이전의 버퍼. CTRL+^ 을 하면 가는 버퍼 a : 현재 화면에 보이는 버퍼. 창 분할 중이면 여러개가 a 이다. + : 변경된 부분이 있는 버퍼
<CTRL+^>	이전 파일 열기. ctrl+6 을 CTRL+^ 로 표기한다. 계속 누르면 최근 파일을 왔다갔다 할 수 있다. :e 로 열고 계속 왔다갔다~
vi a.txt b.txt c.txt 후에 :n 이나 :N	여러 파일 왔다갔다
:n 파일패턴	버퍼에 파일들 추가. :n *.txt 하면 현재 디렉토리의 모든 *.txt 파일들을 버퍼목록에 추가
:update, :up	수정된 내용이 있을 때만 저장
:w filename	filename 으로 현재 내용 저장. 여전히 현재 파일을 편집함
:save filename	filename 으로 현재 내용 저장. 새로운 파일을 편집하게 됨

:x	:update and :q
gf	커서가 위치한 파일을 열어준다. 찾을 수 없으면 path 를 뒤져서 열어준다

tab 기능

vim7.0 버젼부터 지원되는 기능

:tabnew, :tabne	새 탭 생성
:tabedit filename, :tabe filename	새 탭으로 편집
:tabnext n, :tabn	n 번째 탭으로 이동. n 없으면 다음 탭으로 이동
:tabprevious	이전 탭으로 이동
vim -p a.txt b.txt	시작 할 때 탭으로 여러 파일 열기

창 분할/나누기

:sp	가로 모드로 나누기

:vs	세로 모드로 나누기
:qa	모든 창 닫으면서 vi 종료
<CTRL+W> <i,j,k,l>	<i,j,k,l> 방향의 창으로 이동
<CTRL+W> <+,->	현재 창 크기 늘이기/줄이기

이동

다음 단어	w	
이전 단어	b	
현재 단어의 끝으로	e	
그 줄에서 빠른 찾기	f(문자)	fi 를 누르면 그 줄에서 가장 가까운 i 로 이동
파일의 끝	G	
파일의 처음	gg	
특정 라인으로	:(숫자) 혹은 (숫자)G	:10 혹은 10G 를 누르면 10번째 줄로 이동
화면의 처음 줄로	H	high
화면의 가운데 줄로	M	middle
화면의 마지막 줄로	L	low

마지막으로 수정한 줄로	‘.	

스크롤

기능	단축키	내용
다음 페이지	Ctrl + f	
이전 페이지	Ctrl + b	
현재 커서 위치를 화면의 가운데로 스크롤	zz	
한줄 위로 스크롤	Ctrl + y	윈도우에서는 redo 라서... 음...
한줄 아래로 스크롤	Ctrl + e	

블럭 잡기 - 범위 선택하기

v	블럭 시작	
V	줄단위 블럭 시작	
Ctrl+v 혹은 Ctrl+q	Vertical 선택 시작	Ctrl+v 후에 Shift_i 를 누르고 // 를 누르고 esc 2번 누르면 선택부분 주석처리가 되는 셈
#G	블럭 잡기 중에 특정 행까지 이동	

특정 위치 기억**(mark)** 하기

mark는 vim 에서 특정한 위치를 지정할 때 쓴다. 이 지정된 위치는 다른 명령어들과의 조합을 이용해서 다양하게 쓰일 수 있다.
마크는 a~z 까지 26개를 쓸 수 있다

```
a마크하기                              ma
b마크하기                              mb
...                                    ...

a마크로이동                            `a
b마크로이동                            `b
...                                    ...
```

undo & redo

Undo	u
Redo	Ctrl+r

줄 나누기 **- Split by line**

```
:%s/kim/\r/g
```

kim 을 endline 으로 치환하면서 줄바꿈을 넣어준다

대소문자 바꾸기

범위 지정 후 u	범위를 소문자로 바꾸기	
범위 지정 후 U	범위를 대문자로 바꾸기	
범위 지정 혹은 글자 위에서 ~	대->소문자, 소->대문자로 바꾸기	

명령 모드에서의 메타 문자

명령어	설명	예
.	현재 행	현재 행을 지워라 :.d
$	마지막 행	현재 행부터 마지막 행까지 지워라 :.,$d
+#	현재 위치 포함 #째줄 아래까지	현재 위치부터 3줄아래까지 삭제 :+3d
-#	현재 위치 포함 #째줄 위까지	현재 위치부터 3줄위까지 삭제 :-3d
%	1,$ 문서 전체	문서 전체를 복사 :%y

복사

한줄복사	yy
현재위치에 복사	P
다음위치에 복사	p
a마크까지 복사	y'a

붙여넣기 할 때 “2p 라고 하면 방금말고 이전에 복사해두었던게 붙여넣어짐

잘라내기 및 삭제

vim 에서는 삭제가 모두 잘라내기로 동작한다. 레지스터는 윈도우에서의 클립보드라고 생각하면 되고, 이 내용은 _viminfo 파일에 저장되기 때문에 vim 을 다시 시작해도 사용할 수 있다.

현재 위치 이후의 모든 내용 삭제(같은 행 내에서)	d$ 혹은 D
현재 위치의 단어 삭제	daw
현재 줄 삭제	dd
잘라낸 목록보기	:reg ^J 는 endline ^I 는 탭
잘라낸 목록에서 골라서 붙여넣기	“ap 하면 “a 레지스터가 붙여넣기 됨
레지스터에 넣기	“ayy 하면 “a 레지스터에 현재줄을 보관
레지스터에 덧붙이기	“Ayy 하면 “a 레지스터에 현재줄을 덧붙임

정규 표현식

- http://kldp.org/KoreanDoc/html/Vim-KLDP/edit-11.html#EDIT-12

아래에 나오는 것을 / 를 입력하고 입력해보자.
^Load.*$\n : 앞에 Load 로 시작하고 아무 문자나 계속 오다가 마지막에 end-of-line 이 오는 것을 찾는다.

[[:digit:]] : 숫자를 찾는다.

찾기

setting find hilight	:set hlsearach , :set nohlsearch, current off : nohlsearch
setting incremental search	:set incsearch
대소문자 구분끄기	:set ic
대소문자 구분켜기	:set noic

현재 단어를 아래로 찾기 (단어위에서)*
현재 단어를 위로 찾기 (단어위에서)#

fc	현재 문장에서 c 를 검색	
;	현재 문장에서 검색을 다시 실행	
,	현재 문장에서 검색을 반대방향으로 실행	
:nohl	/ 로 현재 검색된 단어의 하이라이트 해제. 다시 / 하면 하이라이트 됨 영원히 끄려면 :set nohl	
특수문자 입력	<CTRL+Q> 누른 후 입력. 예를 들면, <CTRL+Q><CTRL+M> 을 하면 ^M 을 입력할 수 있음	

바꾸기

:%s/old/new/g

- % 는 1,$를 뜻하는 메타문자
- s 뒤의 / 는 구분자. / 대신에 , 를 쓰면 :%s,old,new,g 가 된다.

- 끝에 /g 가 붙은 것은 한 라인에 old 라는 단어가 2개 이상있어도 모두 바꾸겠다는 뜻이다.
- 끝에 /cg 를 붙이면(c를 더 붙이면) 바꿀 때 물어본다. confirm 의 c 이다.

```
:%s/\(abc\)/--\1--/g
```

- abc 를 찾아서 --abc-- 로 바꾼다

중복행 바꾸기

```
:%s/abc\ndef/abc\rkkk
```

abc

def

를

abc

kkk

로 바꿈.

찾아서 출력하기 **- grep**

:%g/pattern

찾은 후 명령 실행

grep 한 것만 남기기	:v/searchString/d 1
grep 된 줄만 지우기	:%g/searchString/d 1
grep 된 것 근처 5줄 보여주기	:g/<pattern>/z#.5 " Same, but with some beautification. :g/<pattern>/z#.5|echo "=========="

Diff

커맨드 라인에서 diff 형태로 vim 시작 : vim -d filename1 filename2

[c	이전 diff 위치
]c	다음 diff 위치
:diffg	다른 창에서 가져오기(get)
:diffp	다른 창으로 복사(put)
vert diffsplit newfilename	간단하게 비교창 띄우기
zR	diff 시에 접힌 부분을 모두 펼치기
Ctrl+w, =	양쪽 창의 넓이를 같게 만들기

Merge (3way with 4pane)

http://www.toofishes.net/blog/three-way-merging-git-using-vim/

```
gvim -f -d -c "wincmd J" "$MERGED" "$LOCAL" "$BASE" "$REMOTE"
```

를 하면 4way window 를 열 수 있다. perforce 의 경우 -f -d -c “wincmd J” %r %2 %b %1 을 하면 되는데, 모두 가로로 열린다. 실행인자를 살펴보면 \”wincmd J\” 처럼되어 있다. 그래서 별도의 .bat 파일을 만들어서 사용하면 제대로 된다 -_-

:diffg RE	Put merged from REMOTE
:diffg BA	Put merged from BASE
:diffg LO	Put merged from LOCAL

파일 인코딩 **- file encoding**

EUC-KR 은 유닉스의 한글 완성형. cp949는 euc-kr 을 확장하여 더 많은 글자를 포함한 윈도우 계열의 형식

현재 열린 인코딩 보기	:set fileencoding 혹은 :set fenc
기본 인코딩 순서 정하기	:set fileencodings=ucs-bom,utf-8,korea
파일 인코딩 바꾸어 저장하려면	:set fenc=utf-8 후 :w 하면 cp949 로 파일을 열었더라도 utf-8 로 저장함

folding

zo	Folding Open
zO	Folding all open

```
zc                                      Folding close
[z                                      start of folding
]z                                      end of folding
zj                                      next start of folding
```

Key	Descrionton	
zf	visual block 으로 선택된 부분을 접기	

Tips for programmer

들여쓰기 정리 **- indentation**

Visual Mode 로 범위를 정한 다음에 = 키를 누름

탭을 공백으로 변환

Visual Mode 로 범위를 정한 후, :retab (number) 하면 하나의 탭을 number 개의 공백으로 변환. number 생략하면 tabstop 설정을 따름

c++ 프로그래밍할 때 파일 쉽게 이동하기

- http://ctags.sf.net 에서 ctag 를 설치한 후 path 를 건다. 그리고 http://www.vim.org 에서 taglist.vim, a.vim 을 설치한다.
- taglist 는 ctags 로 만들어진 클래스, 파일로 빠르게 이동하는 플러그인이고, a 는 .h, .cpp 파일을 빠르게 이동하는 플러그인이다.
- Ctrl + ^ 를 누르면 방금 전에 열었던 파일을 왔다갔다 하면서 편집할 수 있다.

기타

for programmer :

현재 단어의 manpage 보기 : Shift + K

move to matched brace : %, use the command --> d%

현재부터 시작된 {} 안의 내용을 오른쪽 탭으로 밀기 : >i{

현재부터 시작된 {} 까지도 오른쪽 탭으로 밀기 : >%

변수가 선언된 곳으로 가기 : gd, gD

Indenting a block using visual mode

1. Position the cursor on the left or right curly brace.

2. Start visual mode with the v command.

3. Select the inner {} block with the command i}.

4. Indent the text with >.

Complete Words : Ctrl + P

선택구역 한꺼번에 탭 넣기 : v 로 선택후, > 로 탭 넣기

전체 파일 포맷팅 : gg=G

Vim 기능중에 유용한것:

현재 커서 위치에 있는 헤더파일 열기 gf

-> 되돌아 올땐 Ctrl+o

화면 두개로 나누기 **:split**

지금 소스와 **include** 된 헤더에서 현재 커서가 있는 토큰이 사용된 모든 라인 보기

Ctrl+[I

-> 되돌아 올땐 **Ctrl+o**

윈도우에서 붙여넣기 중 계단 현상 방지

:set paste

줄 합치기

대문자 **J**

UTF-8 파일 열어보기

```
set tenc=korea

set enc=utf-8
```

euc-kr 로 되어 있는 파일을 **utf-8** 환경에서 편집하기#

```
:e ++enc=euc-kr
```

^M 을 안보이게 하기#

```
:set ff=unix
```

위의 명령을 하면 바꾼다. 그 후 :wq 로 저장하면 된다.

나중에 편집할 것

```
c : action like dmotion, but it changes mode to insert mode
```

change upper case and lower case : ~

in find action, next character is special meaning : *[]^%?~$

regular expression : the$ --> 줄의 끝에 있는 the 를 찾아라.

include^ --> 줄의 끝에 있는 include 를 찾아라.

^$ --> find the empty line

. --> match any single character

marks :

set mark : ma, mb, mc ......

move to mark : `a, `b, `c, `"(Move to first line in the file) ...

delete to mark : d`a

show all marks : :marks

exit this file and open new file : :vi filename.ext

when editing multiple file :

    show now editing file list : :args

        next file : :next

 previous file : :prev

     first file : :rewind

      last file : :last

Window

   split : :split

  splie line up : count[Ctrl+W]-
    splie line down : count[Ctrl+W]+
  splie line same : count[Ctrl+W]=
  next window : Ctrl+WW

vi 20| 하면 20컬럼으로감
0 하면 첫칸
라인 이동 후 - 하면 이전 라인으로
3H 하면 현재 화면에서 위에서 3번째줄로 이동
북마킹 mX 북마크로이동 `X

:s 끝에 c 를 하면 컨펌 받음
대문자 Y 현재줄 복사
:g/london/y london 포함줄을 복사

도움말

:help 를 치면 일반 모드, 비주얼 모드 등에서의 명령어를 구분해서 도움말을 볼 수 있는 help 가 나온다. 예를 들면, 일반 모드의 x 명령에 관한 명령은 :help x 이고, 명령행 모드에서의 grep 은 :help :grep 이다. 여러 키 입력은 _ 로 연결해서, :help CTRL-W_CTRL-W 이다.
강조된 단어에서 Ctrl+] 을 누르면 go into, 돌아오는 건 Ctrl+T

링크

http://vim.wikia.com/wiki/Vim_Tips_Wiki : vim best tip

- vim scripting : http://www.ibm.com/developerworks/linux/library/l-vim-script-1/index.html

http://wiki.kldp.org/wiki.php/VimTutorKo

문제 해결

vim 으로 **diff** 가 실패할 때

vim 으로 diff 할 때(vim -d a.txt b.txt) 다음과 같은 에러를 내면서 diff 가 실패할 때가 있다.

E97: Cannot create diffs

diff 를 제대로 하지 못했다는 말인데, 다음을 체크해보자.

- 실행 경로의 diff 가 다른 프로그램이지 않은지
    - Subversion client 가 설치되어서 path 에 subversion 에 포함된 diff 가 먼저 존재해서 vim 의 diff 가 아니라 subversion 의 diff 가 실행되면 제대로 동작하지 않는다
- vimrc 에 set directory=.,$TMP,$TEMP 로 temp directory 를 설정해준다. 임시 diff 파일을 쓰지 못하면 이런 현상이 발생한다.