[OSX Development Environment]

1. ctag & cscope

```
$ brew install ctag
$ brew install cscope
$ vim gen_tags.sh
++
#!/bin/sh

echo "Start to create tags file..."
rm cscope.files
rm cscope.out
rm tags

find . \( -name '*.c' -o -name '*.cpp' -o -name '*.cc' -o -name '*.h' -o -name
'*.hpp' -o -name '*.in' \) -print > cscope.files
cscope -i cscope.files

ctags -R -L cscope.files
echo "Completed to create tags file..."
—

$ vim ~/.zshrc
++
# cscope
export CSCOPE_DB=~/Workspace/rufus_next/src/cscope.out
—

$ vim ~/.vim/init.vim
++
" ctags
" set tags=./tags,tags;$HOME
set tags=~/Workspace/rufus_next/src/tags,tags;$HOME

" cscope
if has("cscope")
    set csprg=/usr/local/bin/cscope
    set csto=0
    set cst
    set nocsverb
    if filereadable("~/Workspace/rufus_next/src/cscope.out")
      cs add ~/Workspace/rufus_next/src/cscope.out
    else
      cs add ~/Workspace/rufus/src/cscope.out
    endif

    if $CSCOPE_DB != ""
        cs add $CSCOPE_DB
```

```
endif

set csverb
set cscopeverbose
" cscope/vim key mappings
" 's' symbol: find all references to the token under cursor
" 'g' global: find global definition(s) of the token under cursor
" 'c' calls:  find all calls to the function name under cursor
" 't' text:   find all instances of the text under cursor
" 'e' egrep:  egrep search for the word under cursor
" 'f' file:   open the filename under cursor
" 'i' includes: find files that include the filename under cursor
" 'd' called: find functions that function under cursor calls
nmap <C-\>s :cs find s <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>g :cs find g <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>c :cs find c <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>t :cs find t <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>e :cs find e <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>f :cs find f <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>i :cs find i <C-R>=expand("<cword>")<CR><CR>
nmap <C-\>d :cs find d <C-R>=expand("<cword>")<CR><CR>

" CTRL-space <C-@> search and split horizonal window
nmap <C-@>s :scs find s <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>g :scs find g <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>c :scs find c <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>t :scs find t <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>e :scs find e <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>f :scs find f <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>i :scs find i <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>d :scs find d <C-R>=expand("<cword>")<CR><CR>

" CTRL-space CTRL-space vertical split
nmap <C-@><C-@>s :vert scs find s <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>g :vert scs find g <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>c :vert scs find c <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>t :vert scs find t <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>e :vert scs find e <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>f :vert scs find f <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>i :vert scs find i <C-R>=expand("<cword>")<CR><CR>
nmap <C-@><C-@>d :vert scs find d <C-R>=expand("<cword>")<CR><CR>

" key map timeout
"set notimeout
" Or
"set timetouteln=4000
"set ttimeout
"set ttimeoutlen=100
```

```
endif
--
```

1.1. Error
connection을 못찾을 경우 cscope.files가 생성된 곳에서 cscope를 터미널에서 한번 실행을 한다.
이후 vi에서 :cs show로 connection을 찾을수 있다.

2. clang-formant
2.1. install
```
$ brew install clang-format
$ sudo apt-get install clang-format
```

2.2. vim .SpaceVim/init.vim or /etc/vim/vimrc
```
++
" python for osx SpaveVim (osx only)
let g:python_host_prog = '/usr/bin/python'
let g:python2_host_prog = '/usr/local/bin/python'
let g:python3_host_prog = '/usr/local/bin/python3'
--

++
" clang-format for osx SpaveVim
map <C-K> :py3f /usr/local/share/clang/clang-format.py<cr>
imap <C-K> <c-o>:py3f /usr/local/share/clang/clang-format.py<cr>
--

++
" clang-format for ubuntu vim
map <C-K> :py3f /usr/share/clang/clang-format-6.0/clang-format.py<cr>
imap <C-K> <c-o>:py3f /usr/share/clang/clang-format-6.0/clang-format.py<cr>
--
```

2.3. usage
```
$ clang-format -style=google -dump-config > .clang-format
```

2.4. neovim에서 python, python3 에러날 경우 처리
```
$ sudo easy_install pip (pip 설치안된 경우)
$ pip install pynvim
$ pip3 install pynvim
```

3. SpaceVim (NeoVim) 에서 QuickFix window 없애는 법
3.1. ale 플러그인 설치
설치는 3.2. 라인 추가후 vim 실행시 ale 에러가 발생하면서 자동으로 플러그인이 설치 된다.
이후 vim 실행시 ale가 설치되었으므로 에러 메세지는 없어짐.

3.2. .SpaveVim의 init.vim 추가
++
```
" disable QuickFix window
let g:ale_set_quickfix = 0
let g:ale_keep_list_window_open = 0
```
--

3.2. .SpaceVim.d의 init.toml 추가
++
```
[options]
    # ale enable to disable QuickFix window
    enable_neomake = false
    enable_ale     = true
```
--

3.3. neomake로 팝업이 발생되면 https://spacevim.org/layers/checkers/ 에서 하기 두가지

SPC t s	Normal	toggle syntax

SPC e c	Normal	clear errors

방법을 사용해 본다.

E. ERROR FIXES
E.1. SSH same IP conflict error
```
$ rm ~/home/jwjang/.ssh/known_hosts
```

E.2. Beyond compare for osx
E.2.1. Delete expire date file for 30 days
```
$ rm /Users/colubrismxkorea/Library/Application\ Support/Beyond\ Compare/registry.dat
```

E.2.2. Automatic delete date file every Monday AM 08:20
```
$ crontab -e
20 8 * * 1 rm /Users/colubrismxkorea/Library/Application\ Support/Beyond\
Compare/registry.dat
:wq
```

E.3. ??