colorscheme elflord " 着色模式：黄色和粉红色为主

set guifont=Monaco:h11 " 字体 && 字号

set history=200

set backspace=2 " 设置退格键可用

set autoindent " 自动对齐

set ai! " 设置自动缩进

set smartindent " 智能自动缩进

set shiftwidth=2 " 换行时行间交错使用2空格

set cindent shiftwidth=2 " 自动缩进2空格

set nu " 显示行号

""set mouse=a " 启用鼠标

set ruler " 右下角显示光标位置的状态行

set incsearch " 开启实时搜索功能

set hlsearch " 开启高亮显示结果

set nowrapscan " 搜索到文件两端时不重新搜索

set nocompatible " 关闭兼容模式

set vb t\_vb= " 关闭提示音

set hidden " 允许在有未保存的修改时切换缓冲区

set autochdir " 设定文件浏览器目录为当前目录

set foldmethod=syntax " 选择代码折叠类型

set foldlevel=100 " 禁止自动折叠

set laststatus=2 " 开启状态栏信息

set cmdheight=2 " 命令行的高度，默认为1，这里设为2

""set showmatch " 显示括号配对情况

set cursorline " 突出显示当前行

"set nowrap " 设置不自动换行

set writebackup " 设置无备份文件

set nobackup

""set list " 显示Tab符，使用一高亮竖线代替

set listchars=tab:\|\ ,

set tabstop=2 " 设置Tab键的宽度 [等同的空格个数]

set expandtab " 将Tab自动转化成空格 [需要输入真正的Tab键时，使用 Ctrl+V + Tab]

syntax enable " 打开语法高亮

syntax on " 开启文件类型侦测

filetype indent on " 针对不同的文件类型采用不同的缩进格式

filetype plugin on " 针对不同的文件类型加载对应的插件

filetype plugin indent on " 启用自动补全

" 每行超过80个的字符用下划线标示

"au BufRead,BufNewFile \*.s,\*.c,\*.cpp,\*.h,\*.cl,\*.rb,\*.sql,\*.sh,\*.vim,\*.js,\*.css,\*.html 2match Underlined /.\%81v/

" 设置编码

set fenc=utf-8

set encoding=utf-8

set fileencodings=utf-8,gbk,cp936,latin-1

" ======= 引号 && 括号自动匹配 ======= "

:inoremap ( ()<ESC>i

:inoremap ) <c-r>=ClosePair(')')<CR>

:inoremap { {}<ESC>i

:inoremap } <c-r>=ClosePair('}')<CR>

:inoremap [ []<ESC>i

:inoremap ] <c-r>=ClosePair(']')<CR>

:inoremap " ""<ESC>i

:inoremap ' ''<ESC>i

:inoremap ` ``<ESC>i

function ClosePair(char)

if getline('.')[col('.') - 1] == a:char

return "\<Right>"

else

return a:char

endif

endf

" ESC + H Normal模式下开启Devhelp查询功能

function! DevHelpCurrentWord()

let word = expand("<cword>")

exe "!devhelp -s " . word . " &"

endfunction

nmap <esc>h :call DevHelpCurrentWord()<CR>

" MiniBufExplorer 多个文件切换 可使用鼠标双击相应文件名进行切换

let g:miniBufExplMapWindowNavVim=1

let g:miniBufExplMapWindowNavArrows=1

let g:miniBufExplMapCTabSwitchBufs=1

let g:miniBufExplModSelTarget=1

" :Tlist 调用TagList

let Tlist\_Show\_One\_File=1 " 只显示当前文件的tags

let Tlist\_Exit\_OnlyWindow=1 " 如果Taglist窗口是最后一个窗口则退出Vim

let Tlist\_Use\_Right\_Window=1 " 在右侧窗口中显示

let Tlist\_File\_Fold\_Auto\_Close=1 " 自动折叠

" :LoadTemplate 根据文件类型自动加载模板

let g:template\_path='~/.vim/template/'

" snipMate Tab智能补全

let g:snips\_author='Ruchee'

""" Ctrl + H 将光标移到当前行的行首

""imap <c-h> <ESC>I

""

""" Ctrl + J 将光标移到下一行的行首

""imap <c-j> <ESC>jI

""

""" Ctrl + K 将光标移到上一行的末尾

""imap <c-k> <ESC>kA

""

""" Ctrl + L 将光标移到当前行的行尾

""imap <c-l> <ESC>A

""

""" jj 保存当前文件并留在插入模式 [插入模式]

""imap jj <ESC>:w<CR>li

""

""" kk 返回Normal模式，不执行保存 [插入模式]

""imap kk <ESC>l

""

""" nt 打开NERDTree窗口，在左侧栏显示 [非插入模式]

""map nt :NERDTree<CR>

""

""" tl 打开Taglist窗口，在右侧栏显示 [非插入模式]

""map tl :Tlist<CR><c-l>

""

""

" ======= 编译 && 运行 ======= "

" 编译源文件

func! CompileCode()

exec "w"

if &filetype == "c"

exec "!clang -std=c99 %<.c -o %<"

elseif &filetype == "cpp"

exec "!clang++ -std=c++98 %<.cpp -o %<"

elseif &filetype == "ruby"

exec "!ruby %<.rb"

elseif &filetype == "sh"

exec "!bash %<.sh"

endif

endfunc

" 运行可执行文件

func! RunCode()

exec "w"

if &filetype == "c" || &filetype == "cpp"

exec "! ./%<"

elseif &filetype == "ruby"

exec "!ruby %<.rb"

elseif &filetype == "sh"

exec "!bash %<.sh"

endif

endfunc

""" Ctrl + C 一键保存、编译

""map <c-c> :call CompileCode()<CR>

""imap <c-c> <ESC>:call CompileCode()<CR>

""vmap <c-c> <ESC>:call CompileCode()<CR>

""

""" Ctrl + R 一键保存、运行

""map <c-r> :call RunCode()<CR>

""imap <c-r> <ESC>:call RunCode()<CR>

""vmap <c-r> <ESC>:call RunCode()<CR>

""