1 安装 Homebrew 2022.07.13

Homebrew的官方网站 https://brew.sh/

1-1 1. 设置USTC的镜像

```
HOMEBREW_CORE_GIT_REMOTE=https://mirrors.ustc.edu.cn/homebrew-core.git
```

1-2 2. 安装Homebrew

执行以下命令

```
/bin/bash -c "$(curl -fsSL https://cdn.jsdelivr.net/gh/ineo6/homebrew-install/
install.sh)"
```

1-3 3. 添加Homebrew到环境变量

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> /Users/$your username$/.zprofile
```

```
eval "$(/opt/homebrew/bin/brew shellenv)"
```

1-4 有关报错

I Brew Update的错误

fatal: Could not resolve HEAD to a revision

```
rm -rf $(brew --repo homebrew/core)
brew tap homebrew/core
```

2 安装Yarn

```
brew install yarn
```

3 安装Neovim

```
brew install neovim
```

4 安装Vim-plug

```
sh -c 'curl -fLo "${XDG_DATA_HOME:-$HOME/.local/share}"/nvim/site/autoload/plug.vim
--create-dirs \
       https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim'
```

5 插件列表

5-1 Vim-surround

```
Plug 'git@github.com:tpope/vim-surround' " Surrounding ysw)
```

使用ysw进行surrounding

6 我的

```
 " Basic configurations
set encoding=UTF-8
set number
set autoindent
set tabstop=4
set shiftwidth=4
set smarttab
set softtabstop=4
set mouse=a
set relativenumber
set smarttab
set cindent
" always uses spaces instead of tab characters
set expandtab

" Specify a directory for plugins
call plug#begin('~/.vim/plugged')

Plug 'git@github.com:tpope/vim-surround' " Surrounding ysw)
Plug 'git@github.com:tpope/vim-commentary' " For Commenting gcc & gc
Plug 'git@github.com:vim-airline/vim-airline' " Status bar
Plug 'git@github.com:ap/vim-css-color' " CSS Color Preview
Plug 'git@github.com:tc50cal/vim-terminal' " Vim Terminal
Plug 'git@github.com:preservim/tagbar' " Tagbar for code navigation
Plug 'git@github.com:terryma/vim-multiple-cursors' " CTRL + N for multiple cursors
" Plug 'git@github.com:neoclide/coc.nvim', {'branch': 'release'}
Plug 'git@github.com:preservim/nerdtree' " NerdTree
" Plug 'git@github.com:tiagofumo/vim-nerdtree-syntax-highlight'
Plug 'git@github.com:ryanoasis/vim-devicons'
Plug 'git@github.com:airblade/vim-gitgutter'
Plug 'git@github.com:ctrlpvim/ctrlp.vim' " fuzzy find files
Plug 'git@github.com:scrooloose/nerdcommenter'
" Plug 'git@github.com:prettier/vim-prettier', { 'do': 'yarn install' }
Plug 'git@github.com:christoomey/vim-tmux-navigator'
Plug 'git@github.com:morhetz/gruvbox'
Plug 'git@github.com:HerringtonDarkholme/yats.vim' " TS Syntax
" fzf plug
Plug 'git@github.com:junegunn/fzf', { 'do': { -> fzf#install() } }
Plug 'git@github.com:junegunn/fzf.vim'
Plug 'git@github.com:vim-autoformat/vim-autoformat'

" Theme tokyonight
Plug 'git@github.com:folke/tokyonight.nvim', { 'branch': 'main' }

Plug 'git@github.com:iamcco/markdown-preview.nvim', { 'do': 'cd app && yarn install' }

" Syntax testing.
Plug 'git@github.com:dense-analysis/ale'

" UltiSnips
Plug 'git@github.com:SirVer/ultisnips'
Plug 'git@github.com:honza/vim-snippets'

Plug 'git@github.com:jiangmiao/auto-pairs'

Plug 'git@github.com:neovim/nvim-lspconfig'

" Initialize plugin system
call plug#end()
```

```
" Theme tokyonight
colorscheme tokyonight
" colorscheme gruvbox
" There are also colorschemes for the different styles
" colorscheme tokyonight-night
" colorscheme tokyonight-storm
" colorscheme tokyonight-day
" colorscheme tokyonight-moon

" NERDTree

" open NERDTree automatically
" autocmd StdinReadPre * let s:std_in=1
" autocmd VimEnter * NERDTree

vmap ++ <plug>NERDCommenterToggle
nmap ++ <plug>NERDCommenterToggle

let g:NERDTreeDirArrowExpandable="+"
let g:NERDTreeDirArrowCollapsible="~"


" air-line plug
let g:airline_powerline_fonts = 1

if !exists('g:airline_symbols')
    let g:airline_symbols = {}
endif

" airline symbols
let g:airline_left_sep = '<-'
let g:airline_left_alt_sep = '<<-'
let g:airline_right_sep = '->'
let g:airline_right_alt_sep = '->>'
let g:airline_symbols.branch = 'Branch'
let g:airline_symbols.readonly = 'ReadOnly'
let g:airline_symbols.linenr = ' Line'

" ctrlp
let g:ctrlp_user_command = ['.git/', 'git --git-dir=%s/.git ls-files -oc --exclude-standard']

" j/k will move virtual lines (lines that wrap)
noremap <silent> <expr> j (v:count == 0 ? 'gj' : 'j')
noremap <silent> <expr> k (v:count == 0 ? 'gk' : 'k')

" " Coc-configurations


" " May need for vim (not neovim) since coc.nvim calculate byte offset by count
" " utf-8 byte sequence.
" set encoding=utf-8
" " Some servers have issues with backup files, see #649.
" set nobackup
" set nowritebackup

" " Having longer updatetime (default is 4000 ms = 4 s) leads to noticeable
" " delays and poor user experience.
" set updatetime=300

" " Always show the signcolumn, otherwise it would shift the text each time
" " diagnostics appear/become resolved.
" set signcolumn=yes

" " Use tab for trigger completion with characters ahead and navigate.
" " NOTE: There's always complete item selected by default, you may want to enable
" " no select by `"suggest.noselect": true` in your configuration file.
" " NOTE: Use command ':verbose imap <tab>' to make sure tab is not mapped by
" " other plugin before putting this into your config.
" inoremap <silent><expr> <TAB>
"       \ coc#pum#visible() ? coc#pum#next(1) :
"       \ CheckBackspace() ? "\<Tab>" :
"       \ coc#refresh()
" inoremap <expr><S-TAB> coc#pum#visible() ? coc#pum#prev(1) : "\<C-h>"

" " Make <CR> to accept selected completion item or notify coc.nvim to format
" " <C-g>u breaks current undo, please make your own choice.
" inoremap <silent><expr> <CR> coc#pum#visible() ? coc#pum#confirm()
"                               \: "\<C-g>u\<CR>\<c-r>=coc#on_enter()\<CR>"

" function! CheckBackspace() abort
"   let col = col('.') - 1
"   return !col || getline('.')[col - 1]  =~# '\s'
" endfunction

" " Use <c-space> to trigger completion.
" if has('nvim')
"   inoremap <silent><expr> <c-space> coc#refresh()
" else
"   inoremap <silent><expr> <c-@> coc#refresh()
" endif

" " Use `[g` and `]g` to navigate diagnostics
" " Use `:CocDiagnostics` to get all diagnostics of current buffer in location list.
" nmap <silent> [g <Plug>(coc-diagnostic-prev)
" nmap <silent> ]g <Plug>(coc-diagnostic-next)

" " GoTo code navigation.
" nmap <silent> gd <Plug>(coc-definition)
" nmap <silent> gy <Plug>(coc-type-definition)
" nmap <silent> gi <Plug>(coc-implementation)
" nmap <silent> gr <Plug>(coc-references)

" " Use K to show documentation in preview window.
" nnoremap <silent> K :call ShowDocumentation()<CR>

" function! ShowDocumentation()
"   if CocAction('hasProvider', 'hover')
"     call CocActionAsync('doHover')
"   else
"     call feedkeys('K', 'in')
"   endif
" endfunction

" " Highlight the symbol and its references when holding the cursor.
" autocmd CursorHold * silent call CocActionAsync('highlight')

" " Symbol renaming.
" nmap <leader>rn <Plug>(coc-rename)

" " Formatting selected code.
```

```vim
" xmap <leader>f  <Plug>(coc-format-selected)
" nmap <leader>f  <Plug>(coc-format-selected)

" augroup mygroup
"   autocmd!
"   " Setup formatexpr specified filetype(s).
"   autocmd FileType typescript,json setl formatexpr=CocAction('formatSelected')
"   " Update signature help on jump placeholder.
"   autocmd User CocJumpPlaceholder call CocActionAsync('showSignatureHelp')
" augroup end

" " Applying codeAction to the selected region.
" " Example: `<leader>aap` for current paragraph
" xmap <leader>a  <Plug>(coc-codeaction-selected)
" nmap <leader>a  <Plug>(coc-codeaction-selected)

" " Remap keys for applying codeAction to the current buffer.
" nmap <leader>ac  <Plug>(coc-codeaction)
" " Apply AutoFix to problem on the current line.
" nmap <leader>qf  <Plug>(coc-fix-current)

" " Run the Code Lens action on the current line.
" nmap <leader>cl  <Plug>(coc-codelens-action)

" " Map function and class text objects
" " NOTE: Requires 'textDocument.documentSymbol' support from the language server.
" xmap if <Plug>(coc-funcobj-i)
" omap if <Plug>(coc-funcobj-i)
" xmap af <Plug>(coc-funcobj-a)
" omap af <Plug>(coc-funcobj-a)
" xmap ic <Plug>(coc-classobj-i)
" omap ic <Plug>(coc-classobj-i)
" xmap ac <Plug>(coc-classobj-a)
" omap ac <Plug>(coc-classobj-a)

" " Remap <C-f> and <C-b> for scroll float windows/popups.
" if has('nvim-0.4.0') || has('patch-8.2.0750')
"   nnoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ? coc#float#scroll(1) : "\<C-f>"
"   nnoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ? coc#float#scroll(0) : "\<C-b>"
"   inoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ?
"\<c-r>=coc#float#scroll(1)\<cr>" : "\<Right>"
"   inoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ?
"\<c-r>=coc#float#scroll(0)\<cr>" : "\<Left>"
"   vnoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ? coc#float#scroll(1) : "\<C-f>"
"   vnoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ? coc#float#scroll(0) : "\<C-b>"
" endif

" " Use CTRL-S for selections ranges.
" " Requires 'textDocument/selectionRange' support of language server.
" nmap <silent> <C-s> <Plug>(coc-range-select)
" xmap <silent> <C-s> <Plug>(coc-range-select)

" " Add `:Format` command to format current buffer.
" command! -nargs=0 Format :call CocActionAsync('format')

" " Add `:Fold` command to fold current buffer.
" command! -nargs=? Fold :call     CocAction('fold', <f-args>)

" " Add `:OR` command for organize imports of the current buffer.
" command! -nargs=0 OR   :call     CocActionAsync('runCommand', 'editor.action.organizeImport')
" " Add (Neo)Vim's native statusline support.
" " NOTE: Please see `:h coc-status` for integrations with external plugins that
" " provide custom statusline: lightline.vim, vim-airline.
" set statusline^=%{coc#status()}%{get(b:,'coc_current_function','')}

" " Mappings for CoCList
" " Show all diagnostics.
" nnoremap <silent><nowait> <space>a  :<C-u>CocList diagnostics<cr>
" " Manage extensions.
" nnoremap <silent><nowait> <space>e  :<C-u>CocList extensions<cr>
" " Show commands.
" nnoremap <silent><nowait> <space>c  :<C-u>CocList commands<cr>
" " Find symbol of current document.
" nnoremap <silent><nowait> <space>o  :<C-u>CocList outline<cr>
" " Search workspace symbols.
" nnoremap <silent><nowait> <space>s  :<C-u>CocList -I symbols<cr>
" " Do default action for next item.
" nnoremap <silent><nowait> <space>j  :<C-u>CocNext<CR>
" " Do default action for previous item.
" nnoremap <silent><nowait> <space>k  :<C-u>CocPrev<CR>
" " Resume latest coc list.
" nnoremap <silent><nowait> <space>p  :<C-u>CocListResume<CR>

" NERDTree
cnoreabbrev tree NERDTree

" Tagbar
nmap <F8> :TagbarToggle<CR>
cnoreabbrev tagbar TagbarToggle

" Autoformat
nmap <F3> :Autoformat<CR>
cnoreabbrev fmt Autoformat

" Terminal
cnoreabbrev tzsh TerminalSplit zsh

" fzf
cnoreabbrev find Ag
```

7 后续

```
pip3 install jedi
```

```
pip3 install pynvim
```

8 案件冲突

在命令模式下输入：`verbose map <key>`就可以查看按键`<key>`的映射

9 Github Update

```
cp -r ./ ~/Desktop/nvim-update

cd ~/Desktop/nvim-update

# 删除./文件夹下所有名字为.git的文件夹

find ./ -name .git |xargs rm -rf

find ./ -name .github |xargs rm -rf

# Echo time information to Changelog file.
time=$(date "+%Y-%m-%d %H:%M:%S")

# echo username and updated date to README.md

echo -e "\nLast updated by **`whoami`** at: "${time} >> "README.md"

# github deploy

git init
git add .
git commit -m "$(date "+%Y-%m-%d %H:%M:%S")"
git branch -M main
git remote add origin git@github.com:desonglll/neovim.git
git push -u origin main -f

cd ~/Desktop

rm -rf nvim-update
```