```
"[tony@tony-pc .vim]$ figlet YBX vimrc
"
"__   ________  __          _
"\ \ / /  __  ) \/ /  __   _(_)_ __ ___   _ __ ___
" \ V /|  _ \\  /  \ \ / / | | '_ ` _ \| '__/ __|
"  | | | |_) /  \   \ V /| | | | | | | | | (__
"  |_| |____/_/\_\   \_/ |_|_| |_| |_|_|  \___|
"
"================================================================
"========== 自定义快捷键 ====================================
"<C-q> 多行的Command-Line模式。
noremap <C-q> Q
nnoremap Q :q<CR>:q
noremap S :w<CR>
noremap s <nop>
"小写s是删除当前字符，将s变成 no operation
noremap R :source $MYVIMRC<CR>
" <C-s> 保存
inoremap <C-s> <ESC>:w<CR>i

" --当前数字-1
" ==当前数字+1(因为+还需要按Shift，就改成==)
noremap -- <C-x>
noremap == <C-a>
noremap <C-a> <ESC>ggVG




"================================================================
"============ 显示设置 ======================================
set wildmenu
set scrolloff=5                    " 永远与最上面，最下面保持5行
set cursorline                     " 显示一条线
set showcmd                        " 输入的命令显示出来，看的清楚些
set backspace=indent,eol,start     " backspace在行首去到上一行行尾
set autochdir                      " 自动切换当前目录为当前文件所在的目录
set wrap                           " 自动换行

"----------------------------------------
" 配置vim，按tab键缩进2个空格，如果是python文件，缩进4个空格
"----------------------------------------
set autoindent    " 自动缩进
set expandtab
set ts=2  " 2个空格
set et    " 将tab转成空格
autocmd FileType python setlocal sw=4 ts=4 et ai

"----------------------------------------
"进入写入模式光标变成竖线
"https://blog.csdn.net/xiaxzhou/article/details/78515044
"----------------------------------------
if has("autocmd")
  au VimEnter,InsertLeave * silent execute '!echo -ne "\e[2 q"' | redraw!
  au InsertEnter,InsertChange *
    \ if v:insertmode == 'i' |
    \   silent execute '!echo -ne "\e[6 q"' | redraw! |
    \ elseif v:insertmode == 'r' |
    \   silent execute '!echo -ne "\e[4 q"' | redraw! |
    \ endif
  au VimLeave * silent execute '!echo -ne "\e[ q"' | redraw!
endif

"------- 高亮 -------------
syntax on "代码高亮
```

```
"------- 搜索 ----------
set hlsearch "设置搜索高亮
"每次打开新的文件去除上次搜索高亮
exec "nohlsearch"
set incsearch   "一边输入一边高亮
set ignorecase  "忽略大小写搜索
set smartcase   "智能搜索
noremap = n
noremap - N

"------------ 分屏-------------
"右侧分屏光标在右边
map s<right> :set splitright<CR>:vsplit<CR>
"左侧
map s<left> :set nosplitright<CR>:vsplit<CR>
"上侧
map s<up> :set nosplitbelow<CR>:split<CR>
"下侧
map s<down> :set splitbelow<CR>:split<CR>

"重新调整左右大小
map <S-left> :vertical resize-5<CR>
map <S-right> :vertical resize+5<CR>
map <S-up> :res +5<CR>
map <S-down> :res -5<CR>

"分屏移动光标
noremap <S-h> <C-w>h
noremap <S-j> <C-w>j
noremap <S-k> <C-w>k
noremap <S-l> <C-w>l

"横变竖，竖变横
map sv <C-w>t<C-w>H
map sh <C-w>t<C-w>K

"----------------------------------------
"vim 打开文件位置为上次关闭光标的位置
"----------------------------------------
if has("autocmd")
au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") | exe "normal! g'\"" |
endif
endif



"=====================================================
"============= 文件格式 ==============================
set encoding=utf-8

"想使用 <SPACE><SPACE> 来进行跳转下一个占位符
noremap <SPACE><SPACE> <Esc>/<++><CR>:nohlsearch<CR>4xi
"noremap <SPACE><SPACE> <Esc>/<++><CR>:nohlsearch<CR>c4i


":%TOhtml 想打印 html 文件
```

```
"================================================
"=============== 插件管理 ========================
"================================================
call plug#begin('~/.vim/plugged')
  Plug 'vim-airline/vim-airline'
  Plug 'majutsushi/tagbar'

  "nertree
  Plug 'scrooloose/nerdtree'
  Plug 'Xuyuanp/nerdtree-git-plugin'

  Plug 'iamcco/mathjax-support-for-mkdp'
  Plug 'iamcco/markdown-preview.nvim', { 'do': { -> mkdp#util#install() }, 'for':
['markdown', 'vim-plug']}

  Plug 'dense-analysis/ale'
  "Plug 'ycm-core/YouCompleteMe'

  "vim snippets
  Plug 'honza/vim-snippets'
  Plug 'SirVer/ultisnips'

  " A Vim Plugin for Lively Previewing LaTeX PDF Output
  Plug 'jcf/vim-latex'  "vim-latex，编译latex文档，使生成pdf文档
  Plug 'xuhdev/vim-latex-live-preview', { 'for': 'tex' }  "用pdf软件实时预览latex文档的编写

  " vim-table-mode
  Plug 'dhruvasagar/vim-table-mode'
call plug#end()

"------------------ NERDTree插件配置 -------------------------
"将F2设置为开关NERDTree的快捷键
map <f2> :NERDTreeToggle<cr>
"修改树的显示图标
  let g:NERDTreeDirArrowExpandable = '+'
  let g:NERDTreeDirArrowCollapsible = '-'
"窗口位置
  let g:NERDTreeWinPos='left'
"窗口尺寸
  let g:NERDTreeSize=30
"不显示隐藏文件
  let g:NERDTreeHidden=0


" nerdtree-git 配置
" https://github.com/Xuyuanp/nerdtree-git-plugin
  let g:NERDTreeIndicatorMapCustom = {
    \ "Modified"  : "✹",
    \ "Staged"    : "✚",
    \ "Untracked" : "✭",
    \ "Renamed"   : "➜",
    \ "Unmerged"  : "═",
    \ "Deleted"   : "✖",
    \ "Dirty"     : "✗",
    \ "Clean"     : "✔",
    \ 'Ignored'   : '☒',
    \ "Unknown"   : "?"
    \ }

"================== Markdown Preview for (Neo)vim=======
noremap m :MarkdownPreview<CR>
let g:mkdp_auto_start = 0
let g:mkdp_auto_close = 1
let g:mkdp_refresh_slow = 0
let g:mkdp_command_for_global = 0
```

```
let g:mkdp_open_to_the_world = 0
let g:mkdp_open_ip = ''
let g:mkdp_browser = '/bin/google-chrome-stable'
let g:mkdp_echo_preview_url = 0
let g:mkdp_browserfunc = ''
let g:mkdp_preview_options = {
    \ 'mkit': {},
    \ 'katex': {},
    \ 'uml': {},
    \ 'maid': {},
    \ 'disable_sync_scroll': 0,
    \ 'sync_scroll_type': 'middle',
    \ 'hide_yaml_meta': 1,
    \ 'sequence_diagrams': {},
    \ 'flowchart_diagrams': {},
    \ 'content_editable': v:false
    \ }
let g:mkdp_markdown_css = ''
let g:mkdp_highlight_css = ''
let g:mkdp_page_title = '「${name}」'

"=================vim-table-mode----------
" Use this option to define the table corner character
let g:table_mode_corner = '|'
" Use this option to define the delimiter which used by
let g:table_mode_delimiter = ' '

function! s:isAtStartOfLine(mapping)
  let text_before_cursor = getline('.')[0 : col('.')-1]
  let mapping_pattern = '\V' . escape(a:mapping, '\')
  let comment_pattern = '\V' . escape(substitute(&l:commentstring, '%s.*$', '', ''), '\')
  return (text_before_cursor =~? '^' . ('\v(' . comment_pattern . '\v)?') . '\s*\v' .
mapping_pattern . '\v$')
endfunction

inoreabbrev <expr> <bar><bar>
          \ <SID>isAtStartOfLine('\|\|') ?
          \ '<c-o>:TableModeEnable<cr><bar><space><bar><left><left>' : '<bar><bar>'
inoreabbrev <expr> __
          \ <SID>isAtStartOfLine('__') ?
          \ '<c-o>:silent! TableModeDisable<cr>' : '__'


function! s:isAtStartOfLine(mapping)
          let text_before_cursor = getline('.')[0 : col('.')-1]
            let mapping_pattern = '\V' . escape(a:mapping, '\')
              let comment_pattern = '\V' . escape(substitute(&l:commentstring, '%s.*$',
'', ''), '\')
                return (text_before_cursor =~? '^' . ('\v(' . comment_pattern . '\v)?') .
'\s*\v' . mapping_pattern . '\v$')
        endfunction

        inoreabbrev <expr> <bar><bar>
                  \ <SID>isAtStartOfLine('\|\|') ?
                  \ '<c-o>:TableModeEnable<cr><bar><space><bar><left><left>' : '<bar>
<bar>'
        inoreabbrev <expr> __
                  \ <SID>isAtStartOfLine('__') ?
                  \ '<c-o>:silent! TableModeDisable<cr>' : '__'

        function! s:isAtStartOfLine(mapping)
                  let text_before_cursor = getline('.')[0 : col('.')-1]
                    let mapping_pattern = '\V' . escape(a:mapping, '\')
                      let comment_pattern = '\V' . escape(substitute(&l:commentstring,
'%s.*$', '', ''), '\')
                        return (text_before_cursor =~? '^' . ('\v(' . comment_pattern .
'\v)?') . '\s*\v' . mapping_pattern . '\v$')
                endfunction
```

```
                inoreabbrev <expr> <bar><bar>
                          \ <SID>isAtStartOfLine('\|\|') ?
                          \ '<c-o>:TableModeEnable<cr><bar><space><bar><left><left>' :
'<bar><bar>'
                inoreabbrev <expr> __
                          \ <SID>isAtStartOfLine('__') ?
                          \ '<c-o>:silent! TableModeDisable<cr>' : '__'

                function! s:isAtStartOfLine(mapping)
                          let text_before_cursor = getline('.')[0 : col('.')-1]
                            let mapping_pattern = '\V' . escape(a:mapping, '\')
                              let comment_pattern = '\V' .
escape(substitute(&l:commentstring, '%s.*$', '', ''), '\')
                                    return (text_before_cursor =~? '^' . ('\v(' .
comment_pattern . '\v)?') . '\s*\v' . mapping_pattern . '\v$')
                          endfunction

                          inoreabbrev <expr> <bar><bar>
                                    \ <SID>isAtStartOfLine('\|\|') ?
                                    \ '<c-o>:TableModeEnable<cr><bar><space><bar><left>
<left>' : '<bar><bar>'
                          inoreabbrev <expr> __
                                    \ <SID>isAtStartOfLine('__') ?
                                    \ '<c-o>:silent! TableModeDisable<cr>' : '__'

"==================Snippet------------------------------------
" 下面是相关的配置信息
let g:UltiSnipsExpandTrigger = "<tab>"
"let g:UltiSnipsListSnippets = "<c-tab>"
let g:UltiSnipsJumpForwardTrigger = "<tab>"   "<tab>切换下一个
let g:UltiSnipsJumpBackwardTrigger = "<s-tab>"  "<Shift-tab>切换上一个

let g:UltiSnipsSnippetDirectories=[$HOME.'/.vim/UltiSnips']
"let g:UltiSnipsSnippetDirectories = [$HOME."/.vim/plugged/vim-snippets/snippets",
".vim/UltiSnips"]
"自定义snippet 的代码片段在哪里寻找


"=============== vim-latex-live-preview
let g:livepreview_previewer = 'zathura'
let g:livepreview_engine = '/usr/local/texlive/2018/bin/x86_64-linux/xelatex'
let g:livepreview_cursorhold_recompile = 0
autocmd Filetype tex setl updatetime=1  "PDF文件刷新频率
nmap <F12> :LLPStartPreview<cr>



"------------------------------------------------------------------------------
" Vim的autocmd FileType 判断语言类型，<C-i>来进行编译运行
"------------------------------------------------------------------------------
" Filetype based Mappings----{
        " Get current filetype -> :echo &filetype or as variable &filetype
        " [ Builds / Compiles / Interpretes  ]

        " C Compiler:
        autocmd FileType c nnoremap <buffer> <C-i> :!gcc % && ./a.out <CR>

        " C++ Compiler
        autocmd FileType cpp nnoremap <buffer> <C-i> :!g++ % && ./a.out <CR>

        " Python Interpreter
        autocmd FileType python nnoremap <buffer> <C-i> :!python % <CR>

        " Bash script
        autocmd FileType sh nnoremap <buffer> <C-i> :!sh % <CR>

        " Executable
        nnoremap <buffer> <C-i> :!./% <CR>
```

```
        "nnoremap <buffer> <C-i> :! %:p <CR>

        " RCs (Configs)
        autocmd FileType vim,zsh,tmux nnoremap <buffer> <C-i> :source % <CR>

        " Java
        autocmd Filetype java nnoremap <buffer> <C-i> :!javac % && java %:r <CR>

" }
```