NeoVim Configuration

Neovim config files, including configurations for Latex, Haskell, Grammarly and C/C++ (https://dotfyle.com/andregpss)

Table Of Contents

Install

In addition to the `.vim` files contained in this project, it is necessary to install the following softwares: 1. Latex - Latex compiler - SumatraPDF - TexLab (download here) 2. Grammarly - Grammarly app (Premium user) - Grammarly language Server (download here or build as described on troubleshooting) - Node.js 3. Haskell - GHC Compiler - Haskell Language Server - Stylish haskell (optional) 4. C/C++ - `clangd` LSP client (`winget install llvm`). - All warnings, such as unused variables, are displayed only if configured in the `.clangd` file and when there is no compilation error in the file. - `gcc` compiler is optional (i think) - Comments about `clangd` and `ccls` installation here. 5. Markdown Readme live view (`npm install -g livedown`) 6. Advanced find - fzf (fuzzy) - fd (file system) - BurntSushi.ripgrep.MSVC (winget)
7. Other - Universal CTags (download here) (useful for file formats whose LSP clients are not installed)

Languages Shortcuts

- **LSPConfig (Grammarly, Latex(TexLab), Haskell, C/C++)**
  - `[d, ]d` - navegate between erros and warnings
  - `<space> e` - shows the popup menu
  - `<space> ca` or `F4` - code action
  - `<space> q`- diagnostic list
  - `<space> rn`- rename
  - `\cr` - shows code lens
  - `K, gd, gD, gi, gr`, `gs, go` - Go commands
    - (d=definition, D=declaration, i=implementation, r=references, s=signature, o=type definition)
  - `<space>D` - Type definition
  - `\sy`- symbols outline (other keys: +,-,f,u)
  - `<space>rn` - rename
  - `<space>wl` - list workspace folders

  - `<space>f` - format
  - `<ctrl>+k, <ctrl>+L, <ctrl>+J`, `<ctrl>+E` (Snippets or code constructors)(works only for Haskell snippets?)

Up

General Shortcuts

- **Telescope**
  - `:Telescope <command>`
  - `\tc` (shows main commands (pickers))
  - `\ff` (find files) (see also file_browser)
  - `\fg` (live grep)(requires ripgrep)
  - `\fb` (search on open buffers)
  - `\td` (diagnostics)
  - `\tq` (quick fix)
  - `\\` (recent files) (see also oldfiles)
  - `\tp` (recent projects)
  - file_browser:
    - `<space>fb`, `<space>bb`
    - Files manipulation: `alt+c` (create file), `alt+r` (rename)
  - Other commands: `hoogle, luasnip, ctags_outline, keymaps, colorscheme, command_history, help_tags, vim_options, builtin, current_buffer_fuzzy_find, grep_string, git_files`.
- NeoTree
  - backspace - go to the parent node
  - P - file preview
  - i - file info
  - o - reorder files
  - / - search for files
  - D - search for directory
  - . - turns current directory into root directory
  - C - closes current node
  - z - closes all nodes
  - a - add file
  - A - add directory
  - d,r - delete and remove a file
  - y,x,p - copy, cut and paste a file
  - m - move a file
  - R - refresh Neotree
  - ? - help
  - Show diagnostic window `:Neotree diagnostics reveal bottom`
- **Navbuddy**
  - Works only with LSPServer!! (not works with Coc)
  - `\b` or `:Navbuddy` - theres an initial error, but it works when you move to the right ('l' key)

  - `j`,`k`,`h`,`l` - move down, up, left, right
    * Some fonts does not have all the icons used by Navbuddy. One of the fonts that have all the icons is Agave Nerd Font.
    * There is an error when Navbuddy is called and the cursor is on the first line of a Haskell file.
- UFO (Folds *indented* code)
  - `zo`,`zc` - open or close fold on current code
  - `za` - alternates (toogle) between open or close current fold
  - `zk` - shows preview Window when code is folded
  - `zR`,`zM` - open or close all folds.
  - `zf`,`zd` - creates or deletes a fold on the selected area
- **Surround**
  - `S<characters><enter>` - surround Visual selection with `<carachters>`
  - `ysiw<character><enter>` - surround word with `<character>`
  - `yssc <digitar comando>` (envolve a linha posicionada com o comando). Ex: yssc textit
  - `ysse <digitar environment>`(idem para o environment). Ex: `ysse tabular`
  - `ysee ou ysec` (mesma coisa para a palavra posicionada)
- **Several plug-ins (Git, Airline, Syntastic, Tabbar, Tagbar,FZV,VimProc)**
  - Livedown Previews (Markdown, Readme)
    * :LivedownPreview
  - Airline Tabs
    * `Tab`, `Shift+Tab`, `\Tab`, `\1`, `\2`, `\3`
    * :bnext, :bprevious, :bfirst
    * :blast, :b10, `:b <buffer-name>`, :bdelete[!], :badd
  - Tabularize
    * `:Tab /=` - Alinha(Tabula) verticalmente o símbolo '=' em todas as linhas selecionadas
    * `\sy`
  - SymbolOutline
    * Others: `+, -, f, u`
  - TagBar
    * F8 - show tagbar
    * `Ctags` executable is necessary
  - Undo Tree
    * `F5`
  - Airline Theme
    * :echo g:airline_symbols - símbolos usados na linha de status
    * `:AirlineTheme <theme>`
  - FZV, VimProc, VIM-FUGITIVE (git), rhubarb
    * :History - últimos arquivos usados

    - `:VimProcBang <comando do SO em uso>`
  - Github
    - :G - git status
    - :Gcommit, :Gpush, :Gpull
    - :Git

Up

Troubleshooting

- HLS (Haskell)
  - Plugin doesn't work:
    - Create a `hie.yaml` file in the project's root folder.
    - There might be an issue with the .cabal file or the existing hie.yaml.
    - Run `haskell-language-server` or `haskell-language-server-wrapper` in the project's root folder and check for errors.
    - Open a code file in the interpreter and see the errors using the following command: `cabal repl <file name>`
  - The `hie.yaml` file seems to be incorrect.
    - Download a program that generates this file automatically from: https://github.com/Avi-D-coder/implicit-hie
  - It appears that the evaluated `.cabal` or `.stack` file is not the one in the project's root folder.
    - Check if there are any other files of these types in subfolders of the project; if so, remove those files from subfolders or from the root of the project.
  - Import is not recognized
    - Add module to `other-modules` clause in cabal
  - No cradle for module (module definition error)
    - Add component to `hie.yaml`
  - Does not show documentation for functions
    - The documentation on hover only works for the parts of the code that were saved and successfully compiled.
    - https://github.com/haskell/haskell-language-server/issues/52
- Latex
  - The LaTeX compiler has stopped working; the error log indicates that there is no error.
    - This happened due to an error in the document; delete the PDF file, recompile again, and pay attention to the error log.
  - New bibliographic references appear as `undefined reference` or on the refference shows `??`
    - Delete the `bbl` file, recompile the project, a new 'bbl' version will be generated, and the reference will appear. If it doesn't work,

delete all temporary files, including the pdf.
  - Bibliographic reference in the wrong format (There is an Error in LaTeX compilation):
    * Change the reference type from online to Misc.
  - In the bbl file: `Missing $ inserted`.
    * Open the `bbl` file and locate the line where the error is. This line contains an invalid character. For example, the character `_` should be `_`.
- C/C++
  - Alternatively, consider using the ALE plugin, which displays messages from GCC, Clangd, and clang-tidy.
  - I attempted to install the CCLS plugin but encountered issues using it in nvim. Here are some observations:
    * Initially, CCLS was my preferred Language Server Protocol (LSP) plugin for C. However, after successfully configuring clangd (as described above), I found that ccls provides similar functionality to clangd.
    * To download CCLS: `choco install ccls`.
    * Before discovering the installation via Chocolatey (choco), I attempted to build CCLS from source but faced challenges. Additional observations include:
      · Couldn't install CCLS using GCC and the Clangd+LLVM installation via winget.
      · Encountered errors when trying to build LLVM with GCC; the ninja command threw an error.
      · Considered using the Visual Studio CL compiler, but the installation demanded significant disk space.
      · Tutorial URL for CCLS build here.
- Grammarly
  - Language server download on the above link may be updated. Solution: build manually
    * How to build: download project -> pnpm install -> pnpm run build -> on your grammarly LSP require script, update the path.
- Others
  - Error installing `Telescope Treesitter` (cant find .h files when using CLang, necessary when installing languages parsers)
  - Error installing `Telescope-media-files` (not compatible with windows, even using `Chafa`).
  - Error when cursor is on Haskell Pragmas and then calls `Navbuddy`

Up

PlugIns to evaluate

- List of main plugins: https://github.com/rockerBOO/awesome-neovim
- https://github.com/ray-x/navigator.lua

- https://github.com/smjonas/inc-rename.nvim (rename variables)
- https://github.com/rmagatti/goto-preview (LSP's goto definition, type definition, implementation, declaration and references calls in floating windows.)
- https://github.com/ldelossa/litee.nvim (Set of Plugins to view code structure)
- https://github.com/linrongbin16/lsp-progress.nvim (Show LSP status on status bar)
- https://github.com/Wansmer/symbol-usage.nvim (display references, definitions, and implementations of document symbols)

Up

In Desuse

- **COC**
  - Neovim's native LSP support offers basic LSP functionality, many users prefer to use plugins like Coc to have a richer and more customizable experience.
  - Coc plugin is an LSP manager for Neovim that offers more advanced and customizable features.
  - Coc plugin (Necessary to complete citations and references) - :CocInstall coc-texlab
  - `:CocInstall coc-hls` (haskell language server)
  - `K` - show the documentation for the current item
  - `[g, ]g` - move between lint comments
  - `\cl` - applies fix suggested by CodeLens
  - `\qf` - quickfix window
  - `\a` - Code Action; lists on a popup the action to the command where the cursor is in.
  - `\ac` - lists all the code actions
  - `\gq` - applies (what?)
  - `\p` - applies code lens (aquelas exibidas como texto virtual)
    - Also used to add a function signature. Put the cursor on a function without signature and then press `\p`
  - `to` - Apply one hint at cursor position (?)
  - `ta` - Apply all suggestions in the file (?)
  - `<space>+a`- shows all the code diagnostic
  - `<space>+o`- shows the functions list in a file
  - `<leader>rn` - rename
  - `<c-space>`to trigger completion - Coc
  - `\f` - format selected Code
  - `:Format` - format all the code in the current file
  - `gd` (coc-definition)
  - `gy` (coc-type-definition) Jump to type definition(s) of current symbol

by invoke
  - `gi` (coc-implementation)
  - `gr` (coc-references) Lists all the references for a type in a project
- **ALE (C Programming**)
  - `[e`, `]e` - navegate the erros (Similar to :lnext, :lprevious)
  - `[a`, `]a` - move between warnings
  - `<ctrl>k`,`<ctrl>j` - move between wraps
  - `:Errors` - shows the error window

  - `F9` - Compile .c files
  - `F11` - Compile and Run .c files
- **NerdTree**
  - `<F2>` or `<Leader>v` or `:NERDTreeFind` (shows the current file on NerdTree)

  - `<F3>` or `<Leader>n` or `:NERDTreeToggle`
  - `hjkl` - navega similar às teclas de navegação
  - `r` - atualiza o diretório corrente
  - `m` - mostra o menu
  - `B` - mostra/oculta os bookmarks
  - `x` - fecha o diretório corrente
  - `X` - fecha todos os diretórios abertos
  - `e` - abre o diretório corrente na janela principal
  - `p` - move para o diretório pai
  - `CD` - muda o diretório root
  - `U` - sobe o diretório root

Up