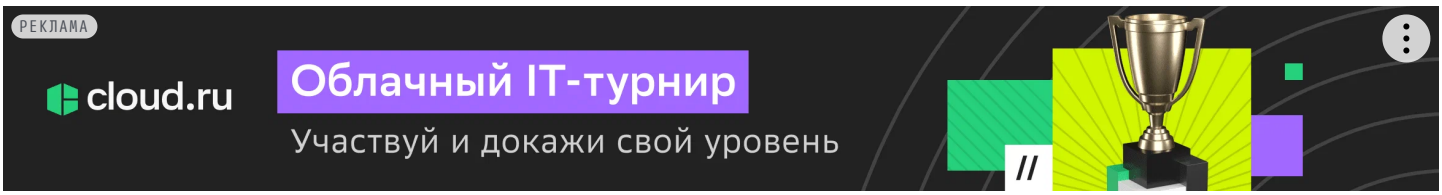


Rembo123
22 янв 2023 в 14:12

Про Vim " Файлы и плагины

9 мин 8.3K

VIM*

Туториал

Пути

Необходимо немного поговорить о способе размещения конфигураций. Существует два радикально противоположных подхода к способу размещения конфигурационных файлов Vim. Одни аккуратно разбивают конфигурацию на несколько файлов и кладут их в разные места, оставляя в основном файле `.vimrc` только вызовы `:source`. Это то как бы поступил адекватный программист. Другие не считают конфигурацию Vim настолько серьезным предприятием и советуют класть всё в кучу, поделив настройки лишь на некие логические секции. Общее мнение - не набивать конфигурацию совсем рандомно.

Я бы наверное изначально поступил первым способом и разложил бы всё по неким "модулям", однако с Vim не всё так просто. Дело в том, что различные расширения подгружаются в редактор последовательно и более того часто взаимодействуют друг с другом - зависят друг от дружки. То есть придумать способ организовать файлы, когда в одном месте у нас только горячие клавиши, в другом цветовая схема, по большому счету нельзя. Каждый плагин отдельно будет иметь свои горячие клавиши, возможно какие-то тонкие настройки цвета, интеграцию с другими плагинами.

Кроме этого, в Vim встроен плагин, который подгружает дополнительные конфигурации в зависимости от типа файла (ftplugin). Что делает картину еще более не очевидной. Другими словами, разбивая конфигурацию на отдельные файлы велик риск наоборот только запутать себя. Поэтому предлагаю еще один компромисс.

В основной файл `.vimrc` я положил только то, что будет работать практически в любом случае в любом окружении. И в конце подгрузил `~/.vim/plugins.vim` в котором всё то что касается расширений. То есть всё что в директории `.vim` можно безопасно отключить в любой момент. Внутри же максимально попытался разделить файл на секции с комментариями по которым потом можно будет найти интересующий плагин или его конкретные настройки.

Соответственно в родных дополнительных конфигурационных файлах в директориях `autoload` `ftplugin`, `after`, `spell` я максимально постараюсь избегать ситуаций, когда они влияют на глобальные настройки. Например, декларации команд будут как в конце основного `.vimrc` так и во всех остальных файлах согласно контекста.

Плагины

Для управления плагинами я использую vim-plug, которого, мне кажется, хватает за глаза. Это единственный плагин, который надо установить вручную.

```
$ curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
    https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```

Скачивать вручную и следить за всеми плагинами тоже не сложно, но всё-таки здесь это нецелесообразно.

Итак плагины. Сперва я бы хотел остановиться опять же на некоем джентльменском наборе программиста без специфики. Это те плагины, которые, как мне показалось, во-первых - не сильно дублируют или заменяют встроенный функционал, не выходят за рамки "идеологии", во-вторых - значительно влияют на производительность, но и не дают

эффекта привыкания. Из-за отсутствия которых не сильно теряешься, запуская голый Vim. Может быть, это то с чего бы следовало начинать знакомиться изначально. Но лично я проделал немного обратный путь. Сперва посмотрел совсем готовые решения. Затем, это мне не очень понравилось, и я выключил плагины, которые я пока не понял. Это мне опять не понравилось и я решил изучить функционал доступный без использования плагинов, уже отвыкая от заученных способов работы. И теперь вот пытаюсь осознать нужны ли какие-то расширения и горячие клавиши в принципе или встроенных возможностей Vim хватает.

Об основных настройках было уже обозначено в предыдущих статьях цикла (раз, два, три). Туда я добавил еще несколько строчек, но о них отдельно.

Основное же блюдо в сегодняшнем меню `.vim/plugins.vim` (https://github.com/johnrembo/provim)

+4 43 5

```
" Plugin section

call plug#begin()

" import .env file
Plug 'tpope/vim-dotenv'

" import .editorconfig file
Plug 'editorconfig/editorconfig-vim'

" fast motions using f F like behavior
Plug 'easymotion/vim-easymotion'

" tag outline 'preservim/tagbar'
Plug 'majutsushi/tagbar', { 'on': 'TagbarToggle' }

" hotkey helper
Plug 'liuchengxu/vim-which-key'

" enable fuzzy finder
Plug 'junegunn/fzf', { 'do': { -> fzf#install() } }
Plug 'junegunn/fzf.vim'

" find project root
Plug 'dbakker/vim-projectroot'

" comment/uncomment blocks and selections
Plug 'preservim/nerdcommenter'
```

```
" more informative status bar
Plug 'vim-airline/vim-airline'
Plug 'vim-airline/vim-airline-themes'

" sonokai full color scheme
Plug 'sainnhe/sonokai'

call plug#end()
```

Первой строчкой идет вызов vim-plug, который выполняет команду `plug#begin()` из файла `.vim/autoload/plug.vim`.

Далее по одному подключаются плагины, причем именно в указанной последовательности, что порой важно.

vim-dotenv

Простейший плагин, позволяющий импортировать переменные и их значения из файла `.env` расположенного по тому же пути что и открываемый буфер. Полезно для некоторых плагинов, которые затем могут использовать их для инициализации.

editorconfig-vim

Импорт универсальных настроек редактора http://editorconfig.org/: отступы, символы конца строк, кодировка и тому прочее. Устанавливает индивидуальные настройки редактора для проекта или директории. Как минимум в одном проекте такую настройку увидеть приходилось и, надо сказать, это оказалось удобно, так как на проекте установились настройки автоматического форматирования такие какими их задумал автор.

vim-easymotion

Этот плагин я упоминал ранее: он должен быть нужен для быстрого перемещения по документу. Всё еще не могу встать на рельсы его использования при перемещении внутри длинных строк, но предположу, что для такого сценария он практически незаменим. Всё-таки `f` и `F` такого простора не дают, хотя научиться использовать их не задумываясь тоже, я считаю, хороший навык.

tagbar

Теги - одно из универсальных понятий для всего что связано с исходным кодом. Плагин с говорящим названием просто отображает эти самые теги сбоку в виде списка в отдельном окне, позволяя передвигаться по ним. Что-то похожее есть в любой современной IDE.

Однако если в IDE такая панель чаще называется "outline" и специфична для языка программирования, то здесь теги используются универсальные. Универсальность может быть немного ограничивает возможности когда дело касается профессиональной разработки, но в среднем стандартного механизма достаточно. Если уж нужны очень специфические и более точные навигаторы по коду, то их как правило можно установить дополнительно.

Для создания соответствующего файла для дерева проекта я использую следующую команду.

```
command! MakeTags !git ls-files | ctags --links=no -L -
```

Она позволяет размечать только релевантные для проекта файлы и пропускать символические ссылки. Если нужно разметить принудительно все файлы, то можно обойтись вызовом консольной команды `:!ctags -R .`. Можно выполнять создание тегов так же при открытии или даже сохранении файлов, но до этого как-то не доходило - проще вызвать вручную, когда есть смысл обновить список.

vim-which-key

Об этом плагине я уже отдельно отзывался в одной из предыдущих статей. Для сегодняшнего выпуска я немного причесал его и добавил имена для некоторых горячих клавиш в конфигурации. Могу дополнительно заметить, что это один из тех примеров, когда настройку плагина не стоит выделять в отдельный файл или даже в отдельную секцию. Лучше создавать метки там, где эти горячие клавиши по факту появляются. Разделение определения и установки меток вскоре приведет к расхождению, а потом и к полному бардаку.

Где-то в одном месте можно сгруппировать универсальные метки и инициализацию

```
" whichkey

let g:which_key_map = {}
let g:which_key_map['w'] = {
      \ 'name' : '+windows' ,
      \ 'w' : ['<C-W>w'     , 'other-window']          ,
      \ 'd' : ['<C-W>c'     , 'delete-window']         ,
      \ '-' : ['<C-W>s'     , 'split-window-below']    ,
      \ '|' : ['<C-W>v'     , 'split-window-right']    ,
      \ '2' : ['<C-W>v'     , 'layout-double-columns'] ,
      \ 'h' : ['<C-W>h'     , 'window-left']           ,
```

```
      \ 'j' : ['<C-W>j'     , 'window-below']          ,
      \ 'l' : ['<C-W>l'     , 'window-right']          ,
      \ 'k' : ['<C-W>k'     , 'window-up']             ,
      \ 'H' : ['<C-W>5<'    , 'expand-window-left']    ,
      \ 'J' : [':resize +5'  , 'expand-window-below']   ,
      \ 'L' : ['<C-W>5>'    , 'expand-window-right']   ,
      \ 'K' : [':resize -5'  , 'expand-window-up']      ,
      \ '=' : ['<C-W>='     , 'balance-window']        ,
      \ 's' : ['<C-W>s'     , 'split-window-below']    ,
      \ 'v' : ['<C-W>v'     , 'split-window-below']    ,
      \ '?' : ['Windows'    , 'fzf-window']            ,
      \ }
let g:which_key_map.b = { 'name' : '+buffers' }
let g:which_key_map.c = { 'name' : '+comment' }
let g:which_key_map.f = { 'name' : '+fzf' }
let g:which_key_map.t = { 'name' : '+tabs tags term' }
let g:which_key_map_visual = {}

nnoremap <silent> <leader> :<c-u>WhichKey '<Space>'<CR>
vnoremap <silent> <leader> :<c-u>WhichKeyVisual '<Space>'<CR>

call which_key#register('<Space>', "g:which_key_map", 'n')
call which_key#register('<Space>', "g:which_key_map_visual", 'v')
```

а метки к конкретным сочетаниям будут разбросаны по всей конфигурации.

```
" tagbar toggle
nmap <leader>tt :TagbarToggle<CR>
let g:which_key_map.t.t = 'Toggle Tagbar'
```

fzf.vim

Еще один плагин, полагающийся на внешнюю программу - fzf. Нужна ли эта программа в целом в системе - вопрос риторический. Мне вот до сих пор было как-то фиолетово. Но это одна из таких программ, после использования которой возникает сакраментальный вопрос: "А что так можно было?". И естественным желанием было встроить её функционал в редактор. Встроили. Получилось, по моему, здорово.

Небольшой эффект привыкания в данном случае думаю во многом оправдан. Тем более что и сам Vim благодаря `set path+=**` позволяет выполнять похожий фокус с путями. А

когда это работает не только с файлами, но и с буферами и регистрами, то жизнь становится чуть проще.

```
" fuzzy finder
nmap <leader><tab> <plug>(fzf-maps-n)
xmap <leader><tab> <plug>(fzf-maps-x)
omap <leader><tab> <plug>(fzf-maps-o)

imap <c-x><c-k> <plug>(fzf-complete-word)
imap <c-x><c-f> <plug>(fzf-complete-path)
imap <c-x><c-l> <plug>(fzf-complete-line)

nmap <leader>ff :Files<CR>
nmap <leader>fg :GFiles<CR>
nmap <leader>fb :Buffers<CR>
nmap <leader>fc :Rg<CR>
nmap <leader>fk :Maps<CR>
let g:which_key_map.f.f = 'FZF files'
let g:which_key_map.f.g = 'FZF Git files '
let g:which_key_map.f.b = 'FZF buffers'
let g:which_key_map.f.c = 'FZF lines'
let g:which_key_map.f.k = 'FZF keymaps'
```

vim-projectroot

Всё вышесказанное работает более естественным образом когда Vim знает где находится корень проекта и сам умеет вставать на нужную директорию. Снова можно проделывать такие манипуляции врукопашную, но когда проектов у вас больше чем пара штук, то скоро установка в правильную директорию начинает немного надоедать. Ничего сверхъестественного этот плагин не делает - ищет в родительских директориях каталог `.git` или любые другие имена указанные в переменной (`let g:rootmarkers = ['.svn', '.git']`), и устанавливает рабочую директорию туда (`:cd projectroot#guess()`).

nerdcommenter

Ну и какой же программист без блочного комментирования. Встроенные средства Vim позволяют разными способами вставить комментарии впереди блока или обрамить выделение символами комментариев, но делается это, мягко говоря, не очень интуитивно. Во всяком случае не совсем так, как это предлагают делать большинство других редакторов.

```
" nerd commenter
map <leader>/ <Plug>NERDCommenterToggle
```

Раскраски

Ниже идут плагины не самые, на первый взгляд, функциональные, но так же значительно влияющие на производительность в целом. Дополнительные цвета и информация не только позволяют лучше ориентироваться в процессе редактирования, но и имеют эстетическую составляющую.

```
" color scheme setup
let g:sonokai_style = 'default'
let g:sonokai_better_performance = 1

let g:airline_theme='sonokai'
let g:airline#extensions#tabline#enabled = 1
let g:airline_powerline_fonts = 1
let g:sonokai_transparent_background = 1
let g:sonokai_diagnostic_text_highlight = 1
let g:sonokai_spell_foreground = 'colored'

" color scheme with enabled plugins
colorscheme sonokai

" scheme fine tuning
hi Comment guifg=#707070 ctermfg=darkgray
hi Visual guifg=#333333 guibg=darkgray
```

Разумеется, какая конкретно цветовая схема приятна для глаза пользователя дело субъективное и личное. Правда одно - родные 16 цветовые схемы как ни крути не могут охватить весь спектр современных задач подсветки. Следует отметить, что `:set termguicolors` может несколько менять палитру, поэтому она должна быть неким образом согласована. Либо можно эту опцию отключить.

Вместо выводов

Далее внизу конфигурации плагинов я буду класть (авто)команды и какие-то дополнительные настройки.

```
" Commands and functions

" change current dir to project root on open
function! <SID>AutoProjectRootCD()
    try
      if &ft != 'help'
        ProjectRootCD
      endif
    catch
      " Silently ignore invalid buffers
    endtry
endfunction
autocmd BufEnter * call <SID>AutoProjectRootCD()
```

Например эта выставляет текущую директорию в корень проекта автоматически. Если, конечно, таковая есть.

Vim без плагинов всё еще мощный и в каком-то своем смысле достаточно удобный редактор, однако нет смысла через силу избегать их использования. Для меня работает следующая формула: посмотри как работает в других редакторах, поищи плагин реализующий понравившийся функционал и попробуй обойтись встроенными средствами. Если не получается совсем или очень уж сложно, то поставь плагин. Сразу хватать всё подряд что советуют люди по каким-то своим причинам не следует. Я искренне убежден, что до каждого дополнительного расширения нужно дойти самостоятельно. С другой стороны надо представлять себе что возможно в принципе и как бывает.

В следующей части, наверное, следует остановиться еще на одном промежуточном шаге, без которого создание собственной IDE может уйти не в ту степь. Пожалуй надо будет остановиться более подробно на filetype, встроенном автодополнении, управлении буферами, окнами и вкладками. А пока прошу оставить в комментариях к статье какие плагины вы считаете должны входить в минимальный и более или менее универсальный набор.

```
:x
```

Теги: vim, plugin, vim-plug, tagbar, easymotion, nerdcommenter, whichkey, fzf

Хабы: VIM

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

Электропочта

22 Карма 0 Рейтинг

Одиночная палата @Rembo123

Пользователь

Подписаться

Сайт

Комментарии 5

CrazyOpossum
22 янв 2023 в 16:03

Вся статья - по большому счёту вкусовщина, особенно в части плагинов. Наверное, правильный шаг - переходить на lua. Добавлю своих приколов, хотя они тоже про vimscript. Я свой конфиг держу одним файлом, на смеси markdown и vimscript. Для навигации можно открыть в любом markdown редакторе (плагин markdown-preview держит обновляемую веб-страницу, например). Плагин - https://github.com/thcipriani/literate-vimrc.

Ответить

domix32
23 янв 2023 в 12:44

> Наверное, правильный шаг - переходить на lua

То бишь переехать на neovim?

Ответить

Rembo123
24 янв 2023 в 04:15

Про переезд на lua и neovim я решил отдельно отписаться, потому что не так там всё однозначно. Но это связано уже скорее с LSP и DAP. Лично меня пока мучает шизофрения на этот счет. Вроде как можно обойтись и coc.nvim, а вроде бы и есть определенные плюшки в nvim, да и lua сама по себе интересна.

Ответить

domix32
24 янв 2023 в 20:40

Я только недавно причастился к modes редакторам и как-то не впечатлился дефолтным vim, в частности сильно неконсистентным поведением в разных окружениях - в tmux одно отваливается, в ssh - другое, на виртуалке с фрёй - третье, конфиги всего этого дела тоже выглядят несколько стрёмно в плане синтаксиса, в отличие от луа, это не считая багов в самом vimscript. Мимокрокодилил на ютубе и наткнулся на крашкурс по nvim и kickstart.nvim и даже почти доволен результатом.

Ответить

CrazyOpossum
24 янв 2023 в 15:23

Точно) Не заметил, что статья про чистый vim, уже забыл что он существует где-то кроме серверов с ванильным vimrc.

Ответить

Зарегистрируйтесь на Хабре, чтобы оставить комментарий

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

dalerank
21 час назад

486-го хватит всем

Простой 15 мин 29K

+86 120 125

breakmirrors
7 часов назад

Ключ от всех дверей: как нашли бэкдор в самых надежных* картах доступа

Средний 9 мин 2.7K

Кейс

+36 26 2

ag_belaya_raduga
11 часов назад

Почему не работает профилактическая медицина

12 мин 3.2K

+34 24 40

Lunathecat
9 часов назад

Прокачиваем «народную» электрогитару Cort KX100

Простой 8 мин 1.5K

Кейс

+31 12 2

the_bat
6 часов назад

Преобразование одноканального LVDS в двухканальный

Средний 3 мин 944

+30 12 7

yadro_team
7 часов назад

Взгляд в игольное ушко: какие дефекты открывает рентген на печатных узлах QFN, SON, DFN и QFP

7 мин 1.1K

Обзор

+27 16 2

PatientZero
6 часов назад

Когда есть разница регистров, но это не верхний и не нижний регистры?

Простой 2 мин 1.5K

Обзор Перевод

+26 13 12

BabayMazay
5 часов назад

Lampwork — декоративная стеклодувная техника. Часть 1. Работы подготовительные

Простой 8 мин 538

Туториал

+23 7 5

alizar
7 часов назад

Анонимный мессенджер — обязательный стандарт для каждого человека

8 мин 4.6K

+18 44 19

CyberPaul
7 часов назад

Трёхмерные. Почему в современных ОС не прижились 3D-интерфейсы?

Простой 6 мин 3.2K

Ретроспектива

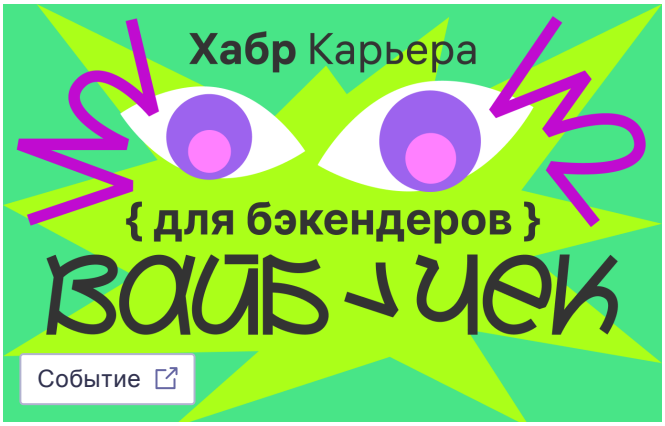
+17 13 35

«Омг, я буду работать среди станков» и другие мифы об айтишниках в металлургии

Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ



Бэкендеры, выбирайте команду по вайбам

Курс на автоматизацию и облака: о трендах DevOps в России

ВАКАНСИИ

Manual QA Engineer
от 30 000 до 60 000 ₽ · Hoohah Barrel · Можно удаленно

PHP-разработчик (middle backend php developer)
от 120 000 ₽ · ООО "М+1" · Новосибирск

Разработчик
от 80 000 ₽ · ИП Веденчук · Сочи

PHP разработчик
до 180 000 ₽ · FSD · Можно удаленно

Senior Golang Engineer в команду Отелло
до 550 000 ₽ · 2GIS · Можно удаленно

Больше вакансий на Хабр Карьере

Ваш аккаунт

Войти
Регистрация

Разделы

Статьи
Новости
Хабы
Компании
Авторы
Песочница

Информация

Устройство сайта
Для авторов
Для компаний
Документы
Соглашение
Конфиденциальность

Услуги

Корпоративный блог
Медийная реклама
Нативные проекты
Образовательные программы
Стартапам

Настройка языка

Техническая поддержка

© 2006–2024, Habr

ЧИТАЮТ СЕЙЧАС

Роскомнадзор рекомендует владельцам ресурсов в интернете отказаться от использования CDN-сервиса CloudFlare

30K 141

Пользователь лишился игрового ПК, который выбросила жена в окно после сообщения о разводе, но в Reddit собрали ему новый

78K 179

486-го хватит всем

29K 125

Найдена чёрная дыра, пожирающая материю со скоростью, превышающей теоретический предел в 40 раз

43K 131

Discord: решено, из-за недавних изменений услуги платформы могут быть недоступны в России и Турции

16K 33

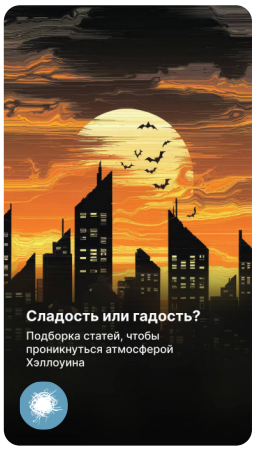
«Омг, я буду работать среди станков» и другие мифы об айтишниках в металлургии

Турбо

ИСТОРИИ

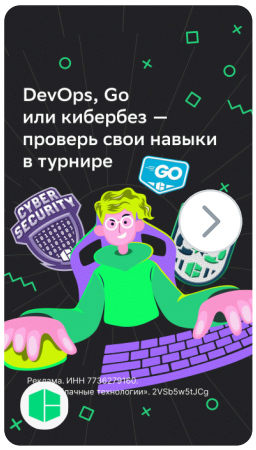
Выложены новые фото "Орла"

Спрошу у лида



Сладость или гадость?

Топ-7 годных статей из блогов компаний



Облачный IT-турнир

БЛИЖАЙШИЕ СОБЫТИЯ

8 октября – 4 декабря

ТурбоХакатон «Решения для электроэнергетики на базе искусственного интеллекта»

Онлайн

Разработка Другое

Больше событий в календаре

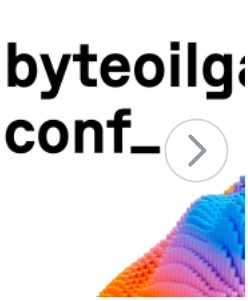
5 – 17 ноября

Вайб-чек для бэкендеров на Хабр Карьере

Онлайн

Разработка

Больше событий в календаре



7 – 8 ноября

Конференция byteoilgas_co

Москва • Онлайн

Разработка Мен

Больше событий в ка