

Introdução
oooooooooo

Normal
oooooooooooo

Inserção
ooooooo

Visual
oooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Minicurso Vim

André William Régis

UFSC

2024

Material de apoio

```
$ git clone --recursive  
https://github.com/AndreKuru/Minicurso-Vim.git
```

practical-vim-examples

Vim



Principais modos

Normal: navegar e editar

Insert: inserção de texto

Visual: editar através de seleção

Command-line: retentor de usuários de primeira viagem

Digitação ideal



Imagen não adaptada.
Disponível em: Wikimedia

Movimentação básica

j : ↓

k : ↑

l : →

h : ←

Jogo para praticar:
vim-adventures.com

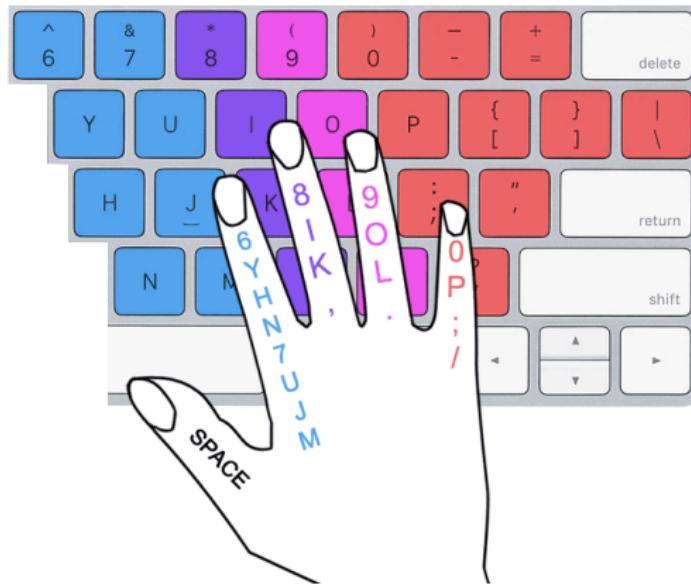


Imagen adaptada.
Disponible en: Wikimedia

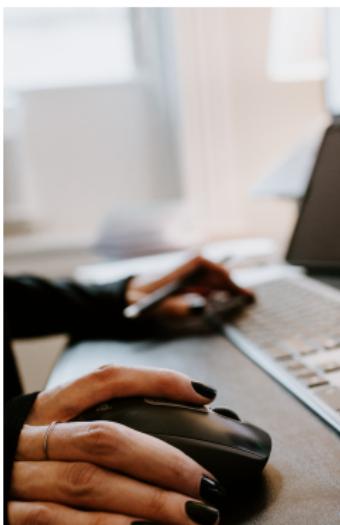
Quando não se precisa manter a mão no mouse

u: undo change

dd: delete line

yy: yank line

p: put text



Mas se quiser
pode

:set mouse=a

Imagen não adaptada.
Disponível em:
Unsplash

Introdução
oooooooo●○

Normal
oooooooooooo

Inserção
oooooo

Visual
oooo

Linha de comando
ooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Experimentando cada modo

i: switch to **I**nsert mode

v: switch to **V**isual mode

:: switch to Command-Line mode

Como sair do vim?

:q : quit

:q! : quit without writing

:w : write

:wq : write and quit

:x : write (if needed) and quit

Introdução
oooooooooo

Normal
●oooooooooooo

Inserção
ooooooo

Visual
ooooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Comando ponto

•: "Repeat last change [...]"

Mudanças comuns

x: delete [count] char(s)

d{motion} : delete

s: substitute [count] char(s)

c{motion} : change

[]: "[...]" are optional."

{}: "[...]" must appear, but which can take a number of different values."

[count] : "An optional number that may precede the command to multiply or iterate the command."

{motion} : "A command that moves the cursor."

Introdução
oooooooooo

Normal
oo●ooooooooo

Inserção
ooooooo

Visual
ooooo

Linha de comando
ooooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Movimentação na linha

0: start of the line

\$: end of the line

^: start non-blank of the line

g_: end non-blank of the line

Movimentação através de palavras

w: word foward

word: "[...]

e: word end foward

letters, digits and underscores,
or a sequence of other non-blank
characters [...]"

b: word backward

blank characters: "space and tab"

ge: word end backward

Movimentação através de PALAVRAS

W: WORD foward

E: WORD end foward

B: WORD backward

gE: WORD end
backward

WORD: "[...]

sequence of non-blank characters
[...]"

blank characters: "space and tab"

Buscas dentro da linha

f: find char
to the right

t: till before char
to the right

;: "Repeat latest f, t, F or T [...]"

F: find char
to the left

,: "Repeat latest f, t, F or T
in opposite direction [...]"

T: till after char
to the left

Buscas no arquivo

/: search forward for
the pattern

?: search backward for
the pattern

*****: search forward for
the nearest word

#: search backward for
the nearest word

n: "Repeat latest "/" or "?" [...]"

N: "Repeat latest "/" or "?"
in opposite direction [...]"

Introdução
oooooooo

Normal
oooooooo●oo

Inserção
oooooo

Visual
oooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Operadores

c: change

=: autoindent

d: delete

g~: swap case

y: yank

gu: make lowercase

>: shift right

gU: make uppercase

<: shift left

Introdução
oooooooooo

Normal
oooooooo●o

Inserção
oooooo

Visual
oooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Teclas especiais

<Enter> : Enter

<CR> : Carriage Return

<Esc> : Escape

<Up> : Up arrow key

<S-...> : Shift + ...

<S-Down> : Shift + Down arrow key

<A-...> : Alt + ...

Ctrl-A : Ctrl + a

<C-...> : Ctrl + ...

<C-x> : Ctrl + x

Aritmética simples

<C-a> : add [count] to the number at or after the cursor

<C-x> : subtract [count] to the number at or after the cursor

Introdução
oooooooo

Normal
oooooooooooo

Inserção
●ooooo

Visual
oooo

Linha de comando
ooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Nem tudo se transforma

i: insert text

a: append text

I: insert text at the start of the line

A: append text at the end of the line

o: obtain new line
below

O: obtain new line
above

Correções rápidas

<C-h> : delete back one character (**<BS>**)

<C-w> : delete back one word

<C-u> : delete back to the start of line

Introdução
oooooooooo

Normal
oooooooooooo

Inserção
oo●ooo

Visual
oooo

Linha de comando
ooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Transições

<Esc> : switch to Normal mode

<C-[> : switch to Normal mode

<C-o> : switch to Insert Normal mode

Acessar registradores

<C-r>{register} : "Insert the contents of a register."

{register} :

+ : the clipboard contents

***** : the clipboard contents (X11: primary selection)

0 : contains the text from the most recent yank command

1-9 : contains the text deleted in ascending order by
the most recent delete or change command

= : the expression register

Introdução
oooooooooo

Normal
oooooooooooo

Inserção
oooo●o

Visual
oooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Inserir caracteres por código

<C-v>191 : £

<C-k>{char1}{char2}

<C-v>u00bf : £

<C-k>?I : £

<C-v>u00b0 : ũ

<C-k>12 : ¡

<C-v>{nondigit} : e.g.

<Home>

ga : print the ascii value

Introdução
oooooooo

Normal
oooooooooooo

Inserção
oooooo●

Visual
oooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Mais um modo

R: Replace mode

gR: Virtual Replace mode

r: replace single character

gr: virtual replace single character

Cada modo uma orientação

<Esc>: switch to Normal mode

v: switch character-wise
Visual mode

<C-[>: switch to Normal mode

V: switch line-wise
Visual Line mode

gv: re-select last visual selection

<C-v>: switch block-wise
Visual Block mode

o: go to other end of selected text

Seleção entre delimitadores

a) or **ab**: a pair of (parentheses)

a]: a pair of [brackets]

a} or **aB**: a pair of {braces}

a>: a pair of <>

a': a pair of 'single quotes'

a‘: a pair of ‘backticks’

a": a pair of ""double quotes"

at: a pair of tags

Seleção entre delimitadores

i) or **ib**: inside of (parentheses)

i]: inside of [brackets]

i} or **iB**: inside of {braces}

i>: inside of <>

i': inside of 'single quotes'

i“: inside of ‘backticks’

i": inside of "double quotes"

it: inside of tags

Seleção de objeto

aw : around of word

iw : inside of word

aW : around of WORD

iW : inside of WORD

as : around of sentence

is : inside of sentence

ap : around of paragraph

ip : inside of paragraph

Falando do ex

:print : print current line

:3print : print line 3

:2,4print : print lines 2 to 4

:<,>print : print lines selected from visual mode

Comandos básicos

:[range]print

[range] : {address}{,{address}}

:[range]join

{address} :

number : absolute line number

. : current line

\$: last line in the file

% : current file

/{pattern}[/] : next line [...]

?{pattern}[?] : previous line [...]

:[range]copy {address}

:[range]move {address}

Comandos básicos

:[range]delete [x]

[x] : "[into register x]"

a, b, c, d, e, f, g, h,

i, j, k, l, m, n, o, p,

q, r, s, t, u, v, w, x,

y, z

:[range]yank [x]

:[range]put [x]

Os mais populares

:[range]normal {address}

:[range]substitute/{pattern}/{string}/[flags]

:[range]global/{pattern}/[cmd]

[cmd] : command from command line

Repetição é a chave

@: : execute the last ex command

@@ : execute the last **@**

@{register} : execute the contents of register

q : record/stop macro

Comandos externos

:shell : start shell

:!{cmd} : execute **{cmd}** in the shell

Comandos externos integrados com o arquivo atual

:read !{cmd} : execute **{cmd}** in the shell
and put the output as text below cursor

:[range]write !{cmd} : execute **{cmd}** in the shell
with **[range]** lines as input

:[range] !{filter} : execute **{cmd}** in the shell
and put the output as text below cursor
with **[range]** lines as input

Rolar texto

<C-E> : scroll window
(Extra lines)

<C-Y> : scroll window
(Yrineu)

<C-D> : scroll window
Donwards

<C-U> : scroll window
Upwards

<C-F> : scroll window
Forwards

<C-B> : scroll window
Backwards

Saltando pelo texto

[count]G : go to line number

% : jump to matching parenthesis

m{letter} : set **{mark}** **{letter}** at cursor position

`{mark} : jump to a **{mark}**

Navegando pelos saltos

<C-]> : "Jump to the definition of the keyword under the cursor."

<C-o> : go to older cursor position in jump list.

<C-i> : go to newer cursor position in jump list.

Autocomplete

<C-n> : find **n**ext match

<C-p> : find **p**revious match

O sucessor moderno

Neovim is exactly what it claims to be. It fixes every issue I have with Vim.

—Geoff Greer

Full-screen Neovim looks cool as hell!

—DHH

A nice looking website, that's one thing Neovim did right.

—Bram Moolenaar



Imagen não adaptada.
Disponível em: [Neovim](#)

O sucessor moderno

API versionada, documentada;

Completamente configurável em Lua;

Ecossistema extensivo de plugins desenvolvidos pela comunidade;

Analisador sintático *TreeSitter* permite highlighting e navegação de código a nível de objetos sintáticos;

Cliente de LSP embutido (mesmo protocolo do *VSCode!*) habilita funcionalidades de inteligência de código (refatoração automática, *code actions*, navegação por referência, etc...);

Language Server Protocol

O que é LSP?

“Adding features like auto complete, go to definition, or documentation on hover for a programming language takes significant effort. Traditionally this work had to be repeated for each development tool, as each tool provides different APIs for implementing the same feature. A Language Server is meant to provide the language-specific smarts and communicate with development tools over a protocol that enables inter-process communication[mic].”

Language Server Protocol

Protocolo

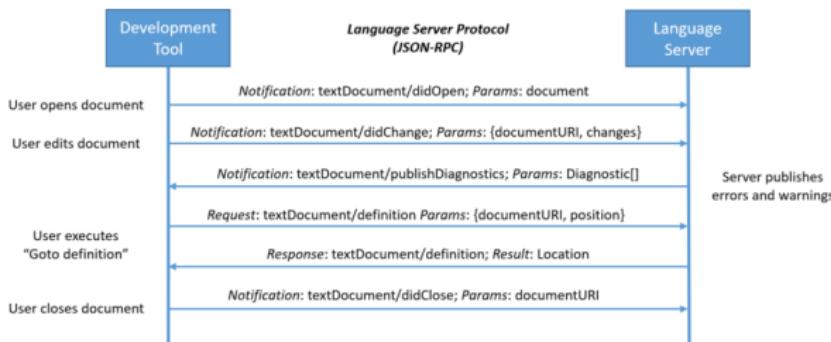


Imagen não adaptada.

Disponível em: Microsoft - Language Server Protocol

Language Server Protocol - Instalação e Configuração

Supporte nativo da API

```
2 lspconfig.rust_analyzer.setup {-
3   settings = {
4     ['pyright'] = {
5       python = {
6         analysis = {
7           | diagnosticMode = "workspace",
8           | typeCheckingMode = "off",
9           },
10        },
11      },
12      ['rust-analyzer'] = {},
13      ['gopls'] = {},
14      ['typescript-language-server'] = {},
15      ...
16    },
}
```

Além de oferecer *bindinds* nativas para a comunicação com LSPs, a configuração dos clientes pode ser feita de forma simples e direta através do pacote `nvim-lspconfig`

Language Server Protocol - Instalação e Configuração

`nvim-lspconfig + Mason.nvim = Configuração rápida, fácil, e portátil`



mason.nvim

Imagen não adaptada.
Disponível em: [Mason.nvim](#)

Portable package manager for Neovim that runs everywhere Neovim runs. Easily install and manage LSP servers, DAP servers, linters, and formatters. [?]

Language Server Protocol - Instalação e Configuração

Um package-manager para servidores LSP

Instalação de LSPs via *Mason.nvim*

TreeSitter

O motor sintático



“Tree-sitter is a parser generator tool and an incremental parsing library. It can build a concrete syntax tree for a source file and efficiently update the syntax tree as the source file is edited.”

TreeSitter

O motor sintático

General enough to parse any programming language;

Fast enough to parse on every keystroke in a text editor;

Robust enough to provide useful results even in the presence of syntax errors;

Dependency-free so that the runtime library (which is written in pure C) can be embedded in any application;

Introdução
oooooooooo

Normal
oooooooooooo

Inserção
ooooooo

Visual
ooooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooo●oooooooo

Encerramento
oo

TreeSitter

O motor sintático

“The goal of nvim-treesitter is both to provide a simple and easy way to use the interface for tree-sitter in Neovim and to provide some basic functionality such as highlighting based on it.”

TreeSitter

O motor sintático

```
15 int ts_lua_add_language(lua_State *L)
16 {
17     if (!lua_gettop(L) < 2 || !lua_isstring(L, 1) || !lua_isstring(L, 2)) {
18         return luaL_error(L, "string expected");
19     }
20
21     const char *path = lua_tostring(L, 1);
22     const char *lang_name = lua_tostring(L, 2);
23
24     if (pmap_has(cstr_t)(langs, lang_name)) {
25         return 0;
26     }
27
28 #define BUFSIZE 128
29     char symbol_buf[BUFSIZE];
30     snprintf(symbol_buf, BUFSIZE, "tree_sitter_%s", lang_name);
31     luaL_Buffer buf;
32     luaL_newbuffer(&buf, symbol_buf, BUFSIZE, "tree_sitter_%s", lang_name);
33     luaL_setmetatable(L, (char *)IObuff);
34
35     uvLib_t lib;
36     if (uv_clopen(path, &lib)) {
37         snprintf((char *)IObuff, IOSIZE, "Failed to load parser: uv_clopen: %s",
38                  uv_dlerror(&lib));
39         uv_dclose(&lib);
40         uv_dclose(&lib);
41         lua_pushstring(L, (char *)IObuff);
42         return luaL_error(L);
43     }
44
45     TSLanguage *lang_parser = lang_parser();
46     if (uv_dlsym(&lib, symbol_buf, (void **)&lang_parser)) {
47         snprintf((char *)IObuff, IOSIZE, "Failed to load parser: uv_dlsym: %s",
48                  uv_dlerror(&lib));
49         uv_dclose(&lib);
50         uv_dclose(&lib);
51         lua_pushstring(L, (char *)IObuff);
52         return luaL_error(L);
53     }
54
55     TSLanguage *lang = lang_parser();
56     if (lang == NULL) {
57         return luaL_error(L, "Failed to load parser: internal error");
58     }
59
60     pmap_put(cstr_t)(langs, xstzdup(lang_name), lang);
61
62     lua_pushboolean(L, true);
63     return 1;
64 }
```

```
12 int ts_lua_add_language(lua_State *L)
13 {
14     if (!lua_gettop(L) < 2 || !lua_isstring(L, 1) || !lua_isstring(L, 2)) {
15         return luaL_error(L, "string expected");
16     }
17
18     const char *path = lua_tostring(L, 1);
19     const char *lang_name = lua_tostring(L, 2);
20
21     if (pmap_has(cstr_t)(langs, lang_name)) {
22         return 0;
23     }
24
25 #define BUFSIZE 128
26     char symbol_buf[BUFSIZE];
27     snprintf(symbol_buf, BUFSIZE, "tree_sitter_%s", lang_name);
28     luaL_Buffer buf;
29     luaL_newbuffer(&buf, symbol_buf, BUFSIZE, "tree_sitter_%s", lang_name);
30     luaL_setmetatable(L, (char *)IObuff);
31     luaL_setmetatable(L, (char *)IObuff);
32     luaL_setmetatable(L, (char *)IObuff);
33     return luaL_error(L);
34
35 TSLanguage *lang = lang_parser();
36     if (lang == NULL) {
37         return luaL_error(L, "Failed to load parser: internal error");
38     }
39
40     pmap_put(cstr_t)(langs, xstzdup(lang_name), lang);
41
42     lua_pushboolean(L, true);
43     return 1;
44 }
```

Highlighting de sintaxe através da árvore construída

TreeSitter

O motor sintático

```
8     :type installation_id: str-
9
10    :return: Github App installation access token-
11    :rtype: str-
12    """
13
14    url = f"https://api.github.com/app/installations/{-
15        | installation_id}/access_tokens"-  

16    headers = {~ You, 3 months ago - API authentication
17        "Accept": "application/vnd.github+json",-
18        "Authorization": f"Bearer {jwt}"-
19    }-
20    response = requests.post(~
21        url,-
22        headers=headers,-
23        timeout=10,-
24    )-
25
26    # pylint: disable=fixme-
27    # TODO: handle errors-
28
29    # pylint: disable=fixme-
30    # TODO: remover cast desnecessário aqui quando incluir
31    # de dados da API-
32    return str(json.loads(response.content)[“token”])-
```

5 (identifier) ; [11, 31] - [11, 46]-
6 type: (type ; [11, 48] - [11, 51]-
7 (identifier))) ; [11, 48] - [11, 51]-
8 return_type: (type ; [11, 56] - [11, 59]-
9 (identifier)) ; [11, 56] - [11, 59]-
10 body: (block ; [12, 4] - [44, 53]-
11 (expression_statement ; [12, 4] - [24, 7]-
12 (string ; [12, 4] - [24, 7]-
13 (string_start) ; [12, 4] - [12, 7]-
14 (string_content) ; [12, 7] - [24, 4]-
15 (string_end))) ; [24, 4] - [24, 7]-
16 (expression_statement ; [26, 4] - [27, 39]-
17 (assignment ; [26, 4] - [27, 39]-
18 left: (identifier) ; [26, 4] - [26, 7]-
19 right: (string ; [26, 10] - [27, 39]-
20 (string_start) ; [26, 10] - [26, 12]-
21 (string_content) ; [26, 12] - [26, 53]-
22 (interpolation ; [26, 53] - [27, 24]-
23 expression: (identifier)) ; [27, 8] - [27, 23]-
24 (string_content) ; [27, 24] - [27, 38]-
25 (string_end))) ; [27, 38] - [27, 39]-
26 (expression_statement ; [28, 4] - [31, 5]-
27 (assignment ; [28, 4] - [31, 5]-
28 left: (identifier) ; [28, 4] - [28, 11]-
29 right: (dictionary ; [28, 14] - [31, 5]-
30 spaces: 4 utf-8 ♦ python linux 29:1 ┌

A ferramenta permite inspecionar a estrutura da árvore sintática em tempo real

Introdução
oooooooo

Normal
oooooooooooo

Inserção
oooooo

Visual
oooo

Linha de comando
ooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
oo

Plugins

Plugin managers

pathogen

vundle

vim-plug

packer.nvim

lazy.nvim

Plugins

Plugin managers

The screenshot shows the user interface of the `lazy.nvim` plugin manager. At the top, there's a navigation bar with links: Install (I), Update (U), Sync (S), Clean (X), Check (C), Log (L), Restore (R), Profile (P), Debug (D), and Help (?). Below the bar, it says "Total: 33 plugins".

The main area is divided into two sections: "Log (6)" and "Loaded (30)".

- Log (6):**
 - mason.nvim ■ updates available
alaf301 chore: update generated code (#767) (4 hours ago)
 - neogit ■ updates available
0d6002c feat: commit --fixup and --squash (4 days ago)
ee6f6ea refactor: rebase and commit selection (4 days ago)
2444769 fixup! fix: use parse_log for rebase commit view (4 days ago)
d12acd0 chore: ignore redefined-local (4 days ago)
 - noice.nvim ■ 0.7ms lualine.nvim ■ updates available
7dac8ce chore(build): auto-generate vimdoc (2 hours ago)
 - nvim-treesitter-context ■ updates available
d28654b ci: bump action versions (34 minutes ago)
 - nvim-treesitter-textobjects ■ 3.76ms nvim-treesitter ■ updates available
e2ee8fd ci: bump stylua version (58 minutes ago)
 - playground ■ 0.36ms nvim-treesitter ■ updates available
3421bbb ci: bump action versions (52 minutes ago)
- Loaded (30):**
 - animation.nvim ■ 0.08ms windows.nvim
 - dressing.nvim ■ 0.39ms > VeryLazy
 - drop.nvim ■ 0.48ms > VimEnter
 - firenvim ■ 0.16ms > start
 - flit.nvim ■ 0.02ms > leap.nvim
 - github-notifications.nvim ■ 0.04ms > lualine.nvim
 - gruvbox.nvim ■ 0.05ms > start
 - lazy.nvim ■ 4.7ms > init.lua
 - lean-ast.nvim ■ 0.02ms > lean.nvim

Instalação interativa de plugins com `lazy.nvim`

Lua

Brasil mentioned!?!?!



Imagen não adaptada.
Disponível em: www.lua.org

designed and developed at

PUC
RIO

Lua

The "Vim API" inherited from Vim: Ex-commands and builtin-functions as well as user-functions in Vimscript. These are accessed through `vim.cmd()` and `vim.fn` respectively.

The "Nvim API" written in C for use in remote plugins and GUIs; see `api`. These functions are accessed through `vim.api`.

Introdução
oooooooooo

Normal
oooooooooooo

Inserção
ooooooo

Visual
ooooo

Linha de comando
oooooooo

Extra
oooo

Neovim
oooooooooooooooooooo

Encerramento
●○

Recomendações

vim-adventures.com

```
$ vimtutor
```

```
:help
```

Referência

-  *Mason.nvim - portable package manager for neovim that runs everywhere neovim runs,*
<https://github.com/williamboman/mason.nvim>,
Accessed: 2024-09-1.
-  *Microsoft - language server protocol*, <https://microsoft.github.io/language-server-protocol/>,
Accessed: 2024-09-1.
-  Drew Neil, *Practical vim: Edit text at the speed of thought*,
Practical Vim (2015), 1–356.