

• Introdução ao Vim:

Aprendendo comandos básicos de navegação

- Ferramentas para escrita de código:

○ Integrated
Development
Environment



- Ferramentas para escrita de código:

Integrated
Development
Environment



Text
Editors



- Variações do Vim:

VI

- Variações do Vim:

VI



- Variações do Vim:

VI



- Por que usar Vim?

- - Não depende do mouse. Sua mão fica no teclado 100% do tempo.

- Por que usar Vim?

- - Não depende do mouse. Sua mão fica no teclado 100% do tempo.
 - Roda direto dentro do terminal, o que facilita na hora de trabalhar remotamente.

● Por que usar Vim?

- - Não depende do mouse. Sua mão fica no teclado 100% do tempo.
 - Roda direto dentro do terminal, o que facilita na hora de trabalhar remotamente.
- Completamente configurável, moldável para o seu uso pessoal.

● Por que usar Vim?

- - Não depende do mouse. Sua mão fica no teclado 100% do tempo.
 - Roda direto dentro do terminal, o que facilita na hora de trabalhar remotamente.
- Completamente configurável, moldável para o seu uso pessoal.
- Alta curva de aprendizado.

● Por que usar Vim?

- Não depende do mouse. Sua mão fica no teclado 100% do tempo.
- Roda direto dentro do terminal, o que facilita na hora de trabalhar remotamente.
- Completamente configurável, moldável para o seu uso pessoal.
- Alta curva de aprendizado.
- Eficiência 🚀



Conhecendo o Vim

- Estou com o Vim aberto. E agora?
- O Vim trabalha com “modos”. Por enquanto, vamos focar apenas em três:



COMANDO



INSERÇÃO

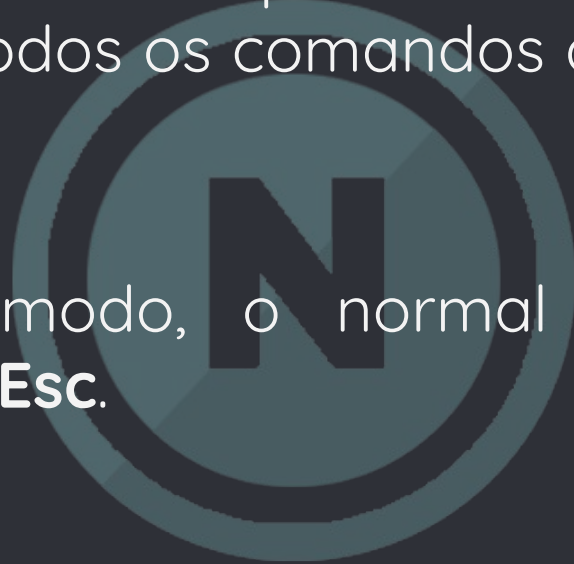


NORMAL

- Modo NORMAL:

- O modo normal é o modo principal, no qual você ficará a maior parte do tempo. Este é o modo usado para quase todos os comandos de movimento e formatação.

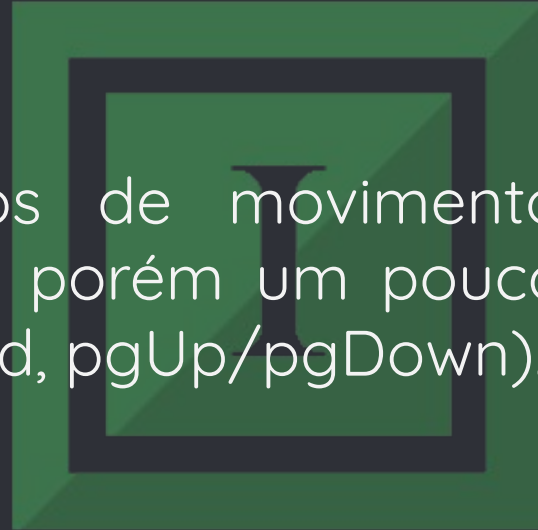
Caso esteja em outro modo, o normal é acessado através da tecla **Esc**.



- Modo **INSERÇÃO**:

- Este é o modo usado para a edição de texto do arquivo. É acessado de várias formas, mas a principal é a tecla **i**.

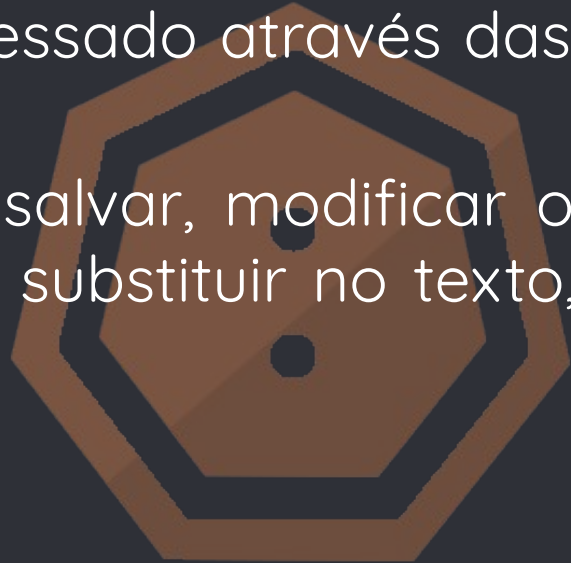
Também possui comandos de movimento comum de outros editores, porém um pouco limitado (setinhas, home/end, pgUp/pgDown).



- Modo COMANDO:

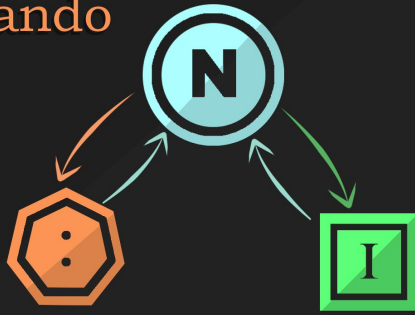
- O modo de linha de comando é pouco usado, mas é o equivalente ao menu do topo de outros programas. Ele é acessado através das teclas `:`, `/` ou `?`.

Exemplos de seu uso são: salvar, modificar o nome do arquivo, buscar e substituir no texto, abrir novas abas, etc.



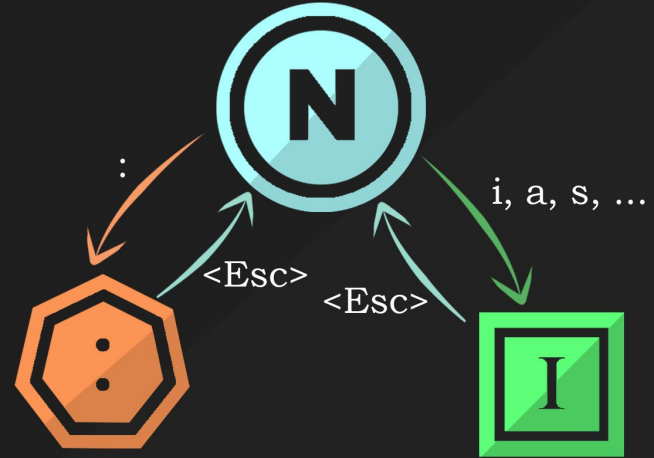
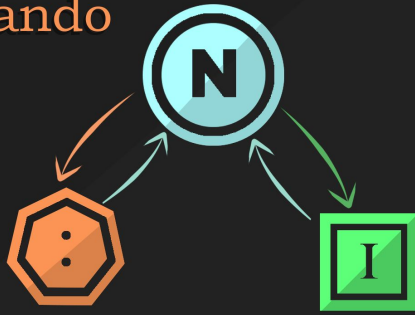
- Relação entre os modos:

- Normal
- Inserção
- Comando



- Relação entre os modos:

- Normal
- Inserção
- Comando





Utilizando os modos:

- Abrindo arquivos:

○ No terminal, digite:
`vim <arquivo>`

Se um arquivo com este nome existir, ele será aberto. Se não existir, o arquivo será criado (o caminho para o arquivo será o mesmo em que o comando do Vim foi executado).

- Abrindo arquivos:

○ No terminal, digite:
`vim <diretório>`

Se o arquivo especificado for um diretório, o Vim abrirá um explorador de arquivos para navegação, e a partir dele você pode abrir um arquivo.

- Alguns comandos básicos:

- :w ← Salva o arquivo.

- :q ← Fecha a janela atual

- :x ← Equivalente ao :wq



Podem ser combinados.

:wq salva o arquivo e fecha a janela logo em seguida.



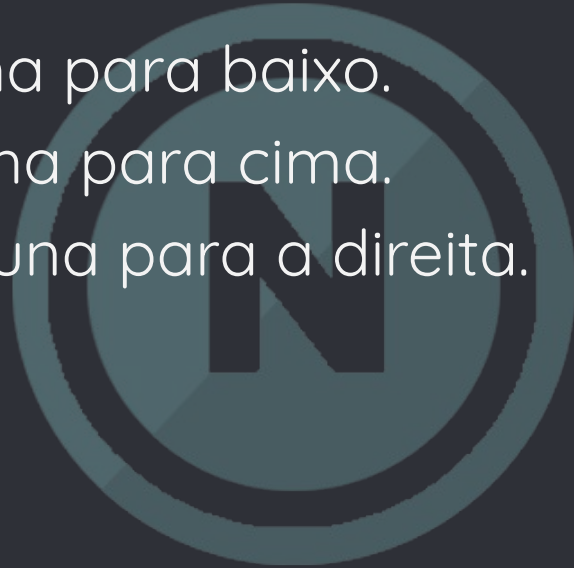
- Navegando pelo modo normal:

- $h \leftarrow$ Move o cursor uma coluna para a esquerda.

$j \leftarrow$ Move o cursor uma linha para baixo.

$k \leftarrow$ Move o cursor uma linha para cima.

$l \leftarrow$ Move o cursor uma coluna para a direita.



- Navegando pelo modo normal:

- h ← Move o cursor uma coluna para a esquerda.

j ← Move o cursor uma linha para baixo.

k ← Move o cursor uma linha para cima.

l ← Move o cursor uma coluna para a direita.



- Navegando pelo modo normal:

- h ← Move o cursor uma coluna para a esquerda.

j ← Move o cursor uma linha para baixo.

k ← Move o cursor uma linha para cima.

l ← Move o cursor uma coluna para a direita.

Alternativamente, você também pode usar as setas, mas elas geralmente ficam longe no teclado.

- Melhorando a navegação horizontal:

- $w \leftarrow$ Avança até o início da próxima palavra

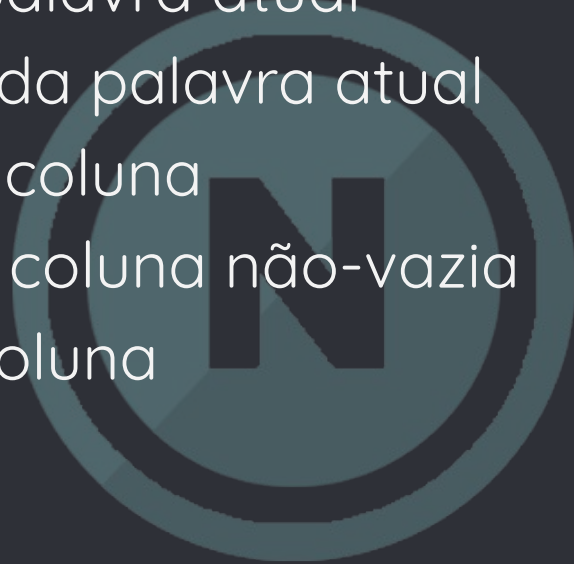
- $e \leftarrow$ Avança até o fim da palavra atual

- $b \leftarrow$ Volta para o começo da palavra atual

- $0 \leftarrow$ Volta para a primeira coluna

- $_ \leftarrow$ Volta para a primeira coluna não-vazia

- $\$ \leftarrow$ Avança até a última coluna



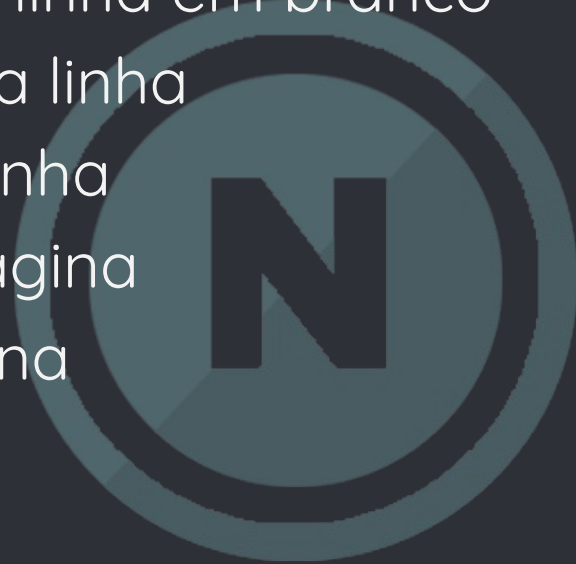
- Melhorando a navegação horizontal:

- $f\langle x \rangle \leftarrow$ Avança até a próxima ocorrência de $\langle x \rangle$
 $t\langle x \rangle \leftarrow$ Avança até o caractere antes da próxima ocorrência de $\langle x \rangle$

Ao pesquisar um caractere com **f** ou **t**, você pode navegar entre todas as ocorrências utilizando ; para avançar para a próxima, ou , para voltar para a anterior!

- Melhorando a navegação vertical:

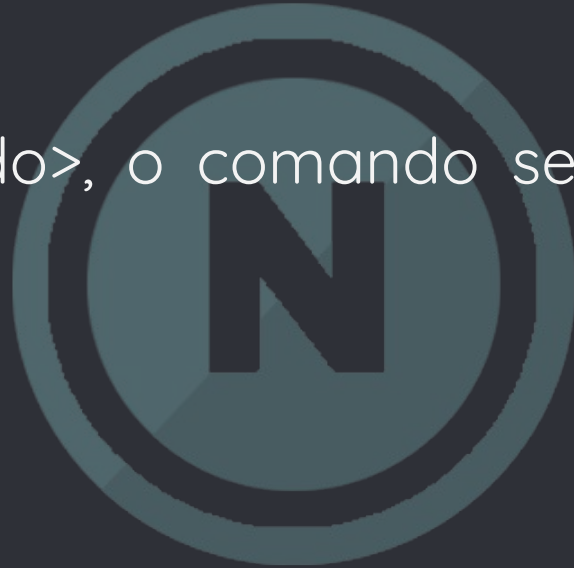
- { ← Volta para a linha em branco anterior
- } ← Avança até a próxima linha em branco
- gg ← Volta para a primeira linha
- G ← Avança até a última linha
- Ctrl + d ← Avança meia página
- Ctrl + u ← Volta meia página



- Repetição de comandos

- Os comandos do modo normal aceitam “combos” de teclas:

Ao digitar <num><comando>, o comando será repetido <num> vezes.



- Muitos comandos para lembrar?

○ O Vim de fato possui uma grande quantidade de atalhos, mas muitos deles foram pensados com mnemônicos para ajudar na fixação.

Dos que vimos até agora, temos:

word, **e**nd, **b**eginning, **f**ind e **t**ill.



N

- Mini spoiler - Mnemônicos:

append

beginning

change

delete

end

find

go

insert

mark

next

open (line)

paste

replace

substitute

till

undo

visual

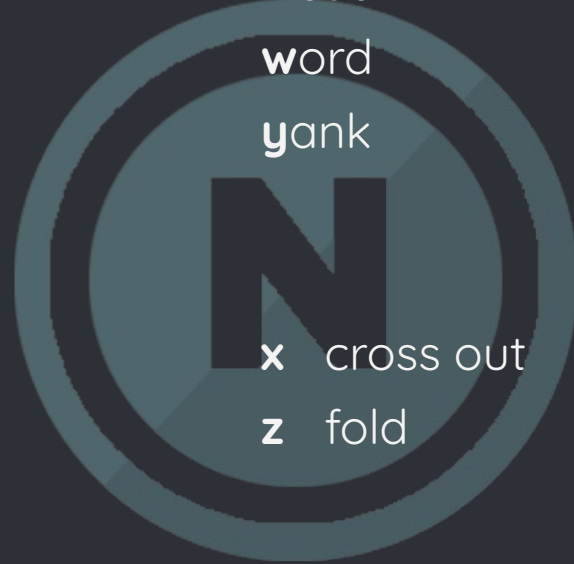
word

yank

N

x cross out

z fold



- Repetição de comandos (parte 2)

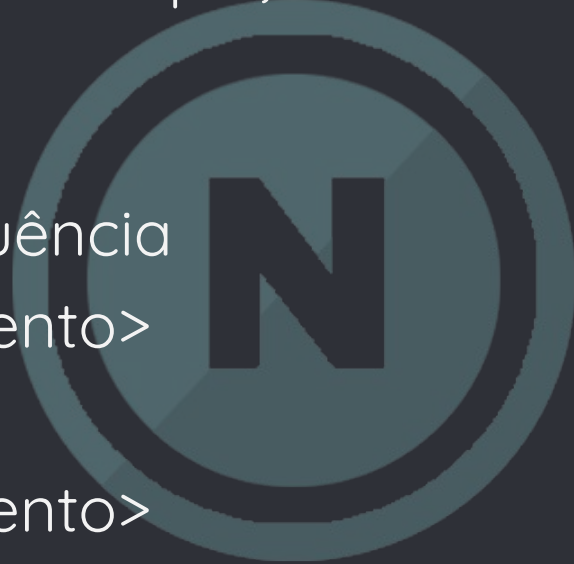
- Alguns comandos exigem alguma instrução adicional. É o caso, por exemplo, do **d**delete, **y**ank e **c**hange.

Geralmente seguem a sequência

<num><comando><movimento>

ou

<comando><num><movimento>

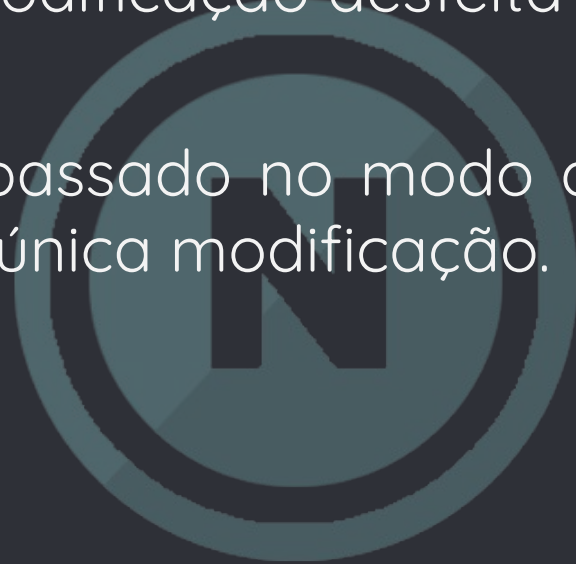


- Corrigindo comandos:

- $u \leftarrow$ desfaz a última modificação

Ctrl + r \leftarrow refaz a última modificação desfeita

Cuidado! Todo o período passado no modo de inserção conta como uma única modificação.



- Introduzindo o modo de INSERÇÃO:

- $i \leftarrow$ Entra no insert mode

- $a \leftarrow$ Entra no insert mode no próximo caractere

Tome cuidado que, ao sair do modo de inserção, o cursor “recuará” uma posição!



- Mais maneiras de entrar no modo INSERÇÃO

- I ← Entra no insert mode no primeiro caractere

- A ← Entra no insert mode no último caractere

- o ← Adiciona uma nova linha abaixo e entra no modo de inserção nela.

- O ← Adiciona uma nova linha acima e entra no modo de inserção nela.



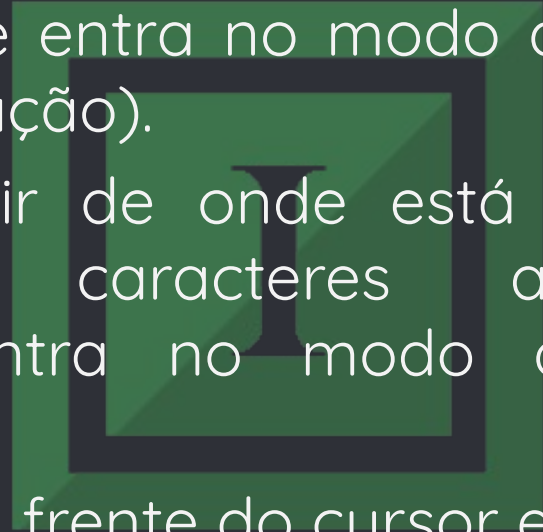
- Mais maneiras de entrar no modo INSERÇÃO

- s ← Apaga o caractere atual, e entra no modo de inserção.

S ← Apaga toda a linha e entra no modo de inserção (mantém a indentação).

c<movimento> ← A partir de onde está o cursor, deleta todos caracteres até <movimento> e então entra no modo de inserção.

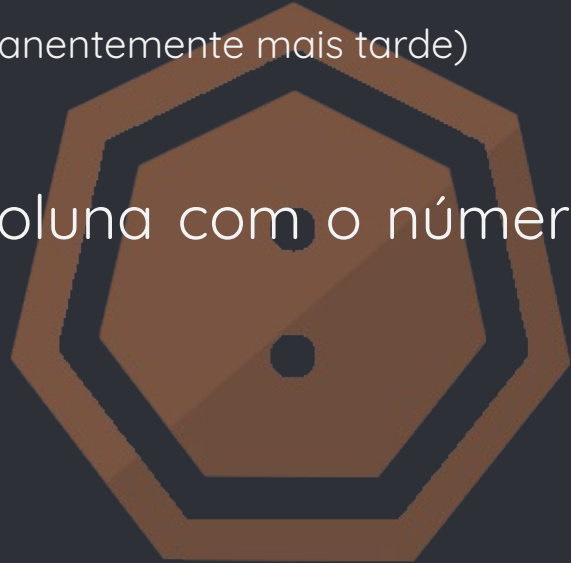
C ← Apaga tudo que está à frente do cursor e...



- Alguns comandos básicos:

○ Você também pode adicionar configurações temporárias ao seu Vim com a linha de comando! (Veremos como fazer permanentemente mais tarde)

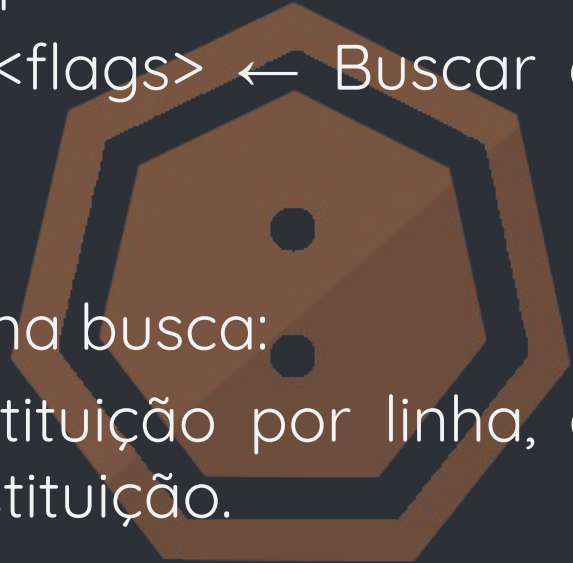
:set number ← Adiciona a coluna com o número das linhas na esquerda.



- O modo comando também ajuda com texto!
- /<termo> ← busca o termo a partir do cursor
- ?<termo> ← busca o termo para trás do cursor
- :<intervalo>s/<old>/<new>/<flags> ← Buscar e substituir

As flags adicionam opções na busca:

g para mais de uma substituição por linha, **c** para perguntar a cada substituição.



- Repetição de comandos (Parte 3)

- . ← Repete a última edição de texto.

Não repete movimentos!



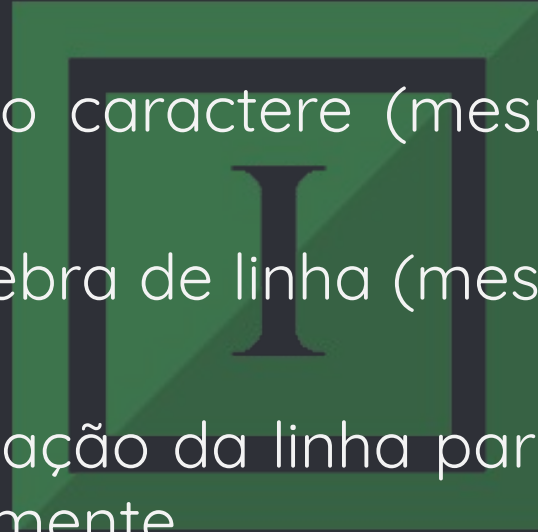
- Atalhos do modo inserção

- Ctrl + p/n ← Autocomplete nativo do Vim, caminha pela primeira/última ocorrência respectivamente.

Ctrl + h ← Apaga o último caractere (mesma coisa que o Backspace)

Ctrl + j ← Adiciona uma quebra de linha (mesma coisa que o Enter)

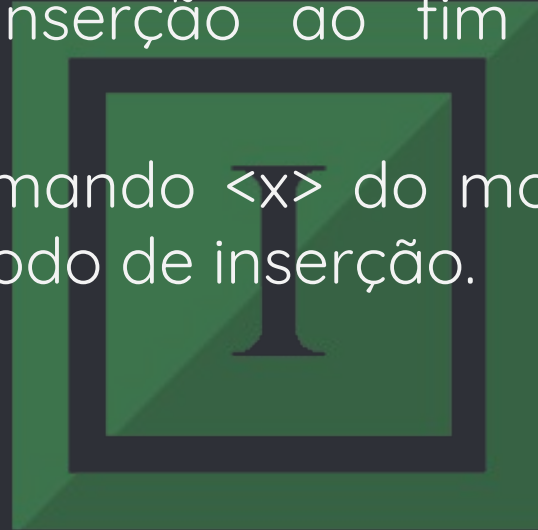
Ctrl + d/t ← Move a indentação da linha para a esquerda/direita respectivamente.



- Atalhos do modo inserção

- Ctrl + o <x> ← Temporariamente volta para o modo normal para executar o comando <x>. Volta para o modo de inserção ao fim da execução.

Alt + <x> ← Executa o comando <x> do modo normal. Não volta para o modo de inserção.



- Edições de texto do modo normal

- d ← Deleta

y ← Copia

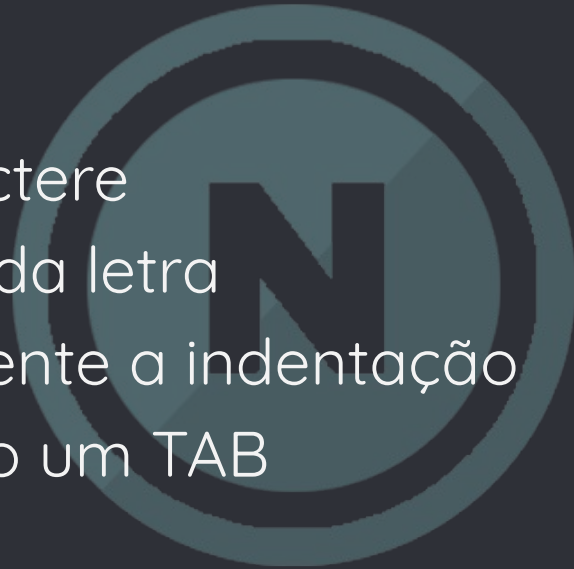
p ← Cola

x ← Deleta um único caractere

~ ← Altera a capitalização da letra

= ← Corrige automaticamente a indentação

< e > ← Move a indentação um TAB



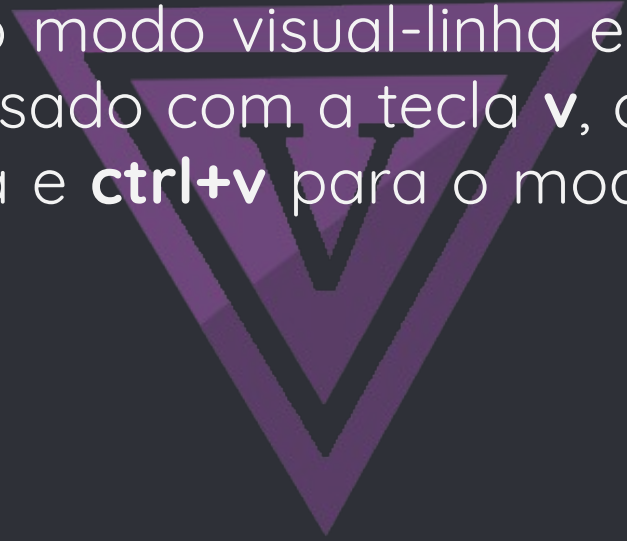


HERE COMES A NEW CHALLENGER!

Um quarto modo do Vim: o modo **VISUAL**

- Modo VISUAL:

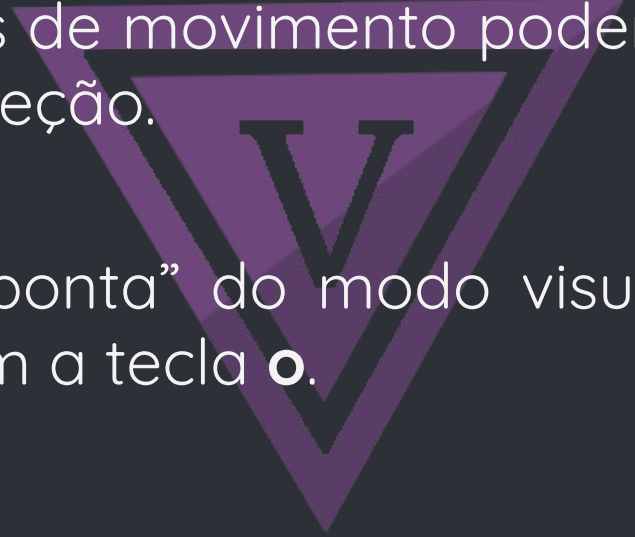
- O modo visual é o equivalente à seleção de texto em outros editores. Além do modo visual, existem dois outros “submodos”, o modo visual-linha e o modo visual-bloco. É acessado com a tecla **v**, ou **shift+v** para o modo linha e **ctrl+v** para o modo bloco.



- Controlando a seleção

- Entrando no modo visual, qualquer comando de movimento fará uma seleção a partir do ponto inicial. Múltiplos comandos de movimento podem ser feitos sem perder a seleção.

Você pode mudar qual “ponta” do modo visual está sendo controlada com a tecla ○.

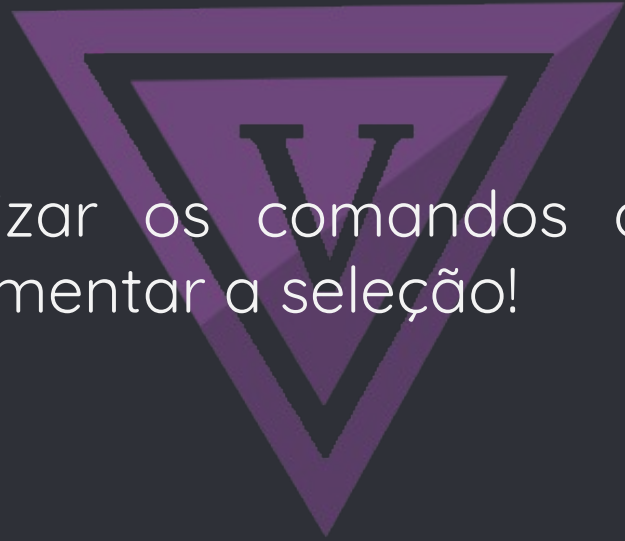


- Comandos

- Feita a seleção, qualquer atalho do modo normal pode ser utilizado e ele será aplicado a toda a seleção.

Você também pode utilizar os comandos de objeto de texto para movimentar a seleção!

Ex: vip, vi(...



- ## Objetos de texto

- São argumentos de movimento para algum comando. Podem começar com **i** para pegar a parte “interna” do objeto, ou **a** para pegar todo o objeto (incluído seus extremos).

Por exemplo, `diç`, deletará todo o conteúdo entre parênteses, enquanto `daç`, deletará todo o conteúdo e também os parênteses.



- Objetos de texto

- $viw \leftarrow \text{palavra}$. Tudo que estiver entre dois caracteres diferentes de 0-9, a-z, A-Z e `_`.

$viW \leftarrow \text{Palavra}$. Tudo que estiver entre dois caracteres de espaço.

$vib, i(, i) \leftarrow \text{bloco}$. Tudo que estiver entre `(` e `)`

$viB, i\{, i\} \leftarrow \text{Bloco}$. Tudo que estiver entre `{` e `}`

$vi[, i] \leftarrow \text{Tudo que estiver entre } [\text{ e }]$

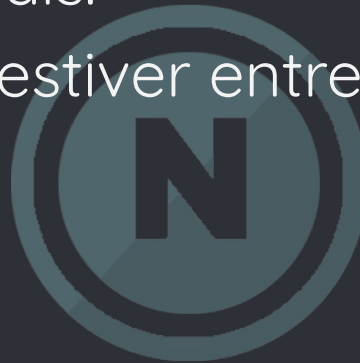
$vi<, i> \leftarrow \text{Tudo que estiver entre } < \text{ e } >$

- Objetos de texto

- ip ← Parágrafo. Todas as linhas entre duas linhas em branco.

is ← Sentença. Parecido com o do parágrafo, mas para em alguns casos específicos quando encontra pontos finais.

it ← tag. Tudo que estiver entre <aaa> e </aaa>



- Comandos “avançados”

- $q\langle x \rangle \leftarrow$ Começa a gravar o macro $\langle x \rangle$. Aperte q de novo para parar a gravação. $@\langle x \rangle$ repetirá os comandos utilizados durante a gravação.

$m\langle x \rangle \leftarrow$ Faz uma marca $\langle x \rangle$ na posição atual do cursor. $'\langle x \rangle$ voltará para a posição da marca.

$\langle x \rangle y \leftarrow$ Copiará o conteúdo para o registrador $\langle x \rangle$ em vez do registrador padrão.

Registradores interessantes são “_y, buraco negro e “+y, clipboard padrão do sistema.

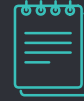
- Saindo do Vim

- ZZ (shift + zz) ← “Atalho” para o comando :x
ZQ (shift + zq) ← “Atalho” para o comando :q

Ctrl + Z ← Deixa o Vim em segundo plano e volta para o terminal temporariamente. O comando **fg** no terminal volta o Vim para primeiro plano.

(Comando do terminal, não se limita ao Vim)

- Alguma dúvida sobre algo?

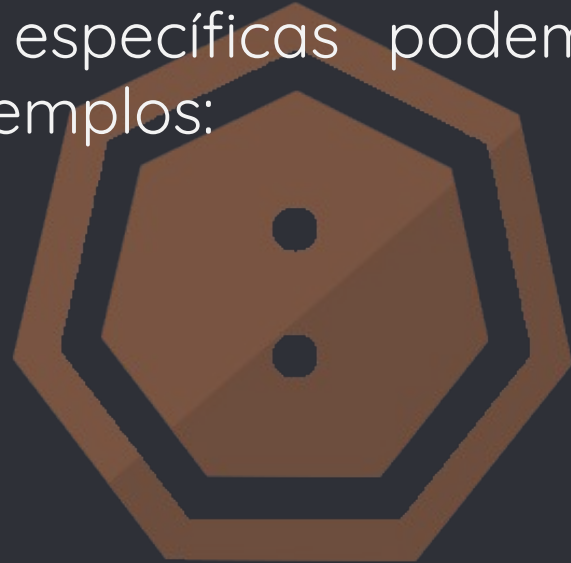


O Vim tem um manual interno que explica tudo sobre o editor. Este manual pode ser acessado por **:h** ou **:help**. Seções específicas podem entrar como argumento. Exemplos:

:h text-objects

:h :s

:h filetype





Seu Vim, do seu jeito!

Criando um arquivo de configurações 

- Criando o arquivo

- Ao iniciar o Vim, ele procurará um arquivo de configurações em diversos lugares. O mais simples de se usar é a home.

Comece criando em sua home o arquivo `.vimrc`

- Babysteps

- Já conhecemos uma configuração útil de se ter ligada todo o tempo: lembram do **set number**?

Basta entrar no modo insert, escrever o comando e salvar o arquivo, agora a próxima vez que você abrir o Vim, a coluna de números já vai estar lá para você!

- Babysteps

- Mas sair e voltar a cada instrução é muito trabalhoso!

Para isso existe o comando **:source**, ou apenas **:so**, que força a releitura do arquivo de configurações sem precisar sair do Vim.

- Algumas sugestões iniciais

- `syntax on` ← Necessário para o texto destacar coisas com cores diferentes.
- `set rnu` ← Coluna com a distância relativa à linha atual. Útil para comandos de movimento.

<code>set expandtab</code> <code>set shiftwidth=x</code> <code>set tabstop=x</code>	}	Mostram um TAB como espaços contíguos e controlam o tamanho do TAB.
---	---	---

- Algumas sugestões iniciais

- `set smarttab` ← Se as opções do slide anterior estiverem ligadas, faz com que um tab coloque apenas espaços o suficiente para acompanhar a indentação da linha anterior.

`set autoindent` ← Mantém a indentação da linha anterior na próxima linha.

`filetype plugin indent on` ← Uma série de comandos que modificam algumas configurações de acordo com o tipo de arquivo.

- Você também pode refazer atalhos!

○ `nnoremap <tecla-atalho> <Ação>`

Utilizando os atalhos:

`<C-x>`, `<A-x>`, `<S-x>` ← Ctrl, Alt e Shift+x,
respectivamente

`<CR>` ← Enter

`<BS>` ← Backspace

`<Esc>`, `<Space>`, `<F1>` ← Intuitivos

- Você também pode refazer atalhos!

○ Exemplo: **nnoremap <C-s> :w<CR>**

Faz com que, no modo normal, apertar **Ctrl+s** salve o arquivo.

Outro exemplo: **nnoremap <A-l> \$**

Faz com que, ao apertar **Alt+l**, o cursor salte para o último caractere da linha.

- Você também pode refazer atalhos!

○ O remap não precisa ser uma única ação! Você pode criar um atalho para uma sequência de quantas ações você quiser:

nnoremap § migg=G'izz

- Você também pode refazer atalhos!

○ O remap não precisa ser uma única ação! Você pode criar um atalho para uma sequência de quantas ações você quiser:

nnoremap § migg=G'izz



mi (cria a marca i)

gg (vai para o início)

= (indentar até...)

G (vai para o fim)

'i (volta para a marca i)

zz (centraliza a linha)

- Você também pode refazer atalhos!

○ A primeira letra indica o modo do remap:

nnoremap ← Modo normal

inoremap ← Modo inserção

vnoremap ← Modo visual



Um pouco mais a fundo nas
configurações

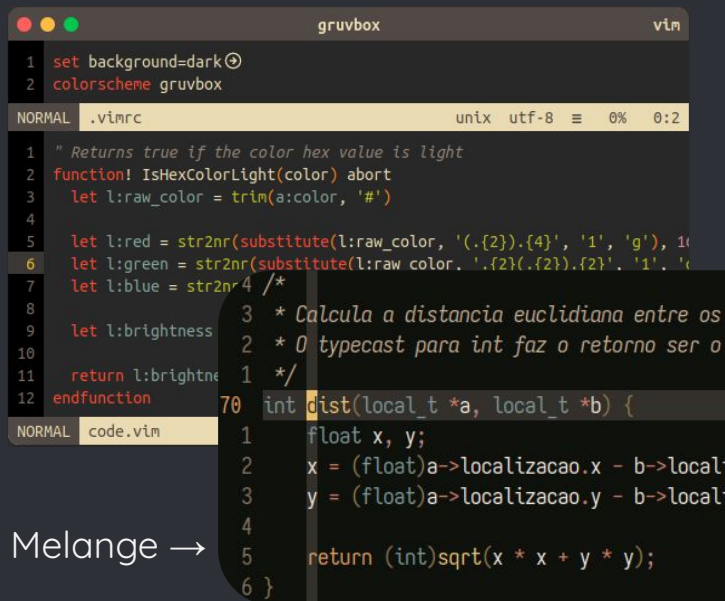
- Plugins

- Plugins mudam a experiência do usuário completamente.

Recomenda-se o uso de um gerenciador de plugins, como o vim-plug.

• Plugins

○ Plugins podem mudar o tema do seu Vim:



```
1 set background=dark
2 colorscheme gruvbox

NORMAL .vimrc unix utf-8 0% 0:2

1 " Returns true if the color hex value is light
2 function! IsHexColorLight(color) abort
3   let l:raw_color = trin(a:color, '#')
4
5   let l:red = str2nr(substitute(l:raw_color, '([0-9a-fA-F]{4})', '1', 'g'), 16)
6   let l:green = str2nr(substitute(l:raw_color, '([0-9a-fA-F]{4})', '2', 'g'), 16)
7   let l:blue = str2nr(substitute(l:raw_color, '([0-9a-fA-F]{4})', '3', 'g'), 16)
8
9   let l:brightness = (l:red * 299 + l:green * 587 + l:blue * 114) / 1000
10
11   return l:brightness > 0.5
12 endfunction

NORMAL code.vim
70 int dist(local_t *a, local_t *b) {
1   float x, y;
2   x = (float)a->localizacao.x - b->localizacao.x;
3   y = (float)a->localizacao.y - b->localizacao.y;
4
5   return (int)sqrt(x * x + y * y);
6 }
```

Melange →



```
1 set background=dark
2 colorscheme papercolor

NORMAL .vimrc unix utf-8 0% 0:2

1 " Returns true if the color hex value is light
2 function! IsHexColorLight(color) abort
3   let l:raw_color = trin(a:color, '#')
4
5   let l:red = str2nr(substitute(l:raw_color, '([0-9a-fA-F]{4})', '1', 'g'), 16)
6   let l:green = str2nr(substitute(l:raw_color, '([0-9a-fA-F]{4})', '2', 'g'), 16)
7   let l:blue = str2nr(substitute(l:raw_color, '([0-9a-fA-F]{4})', '3', 'g'), 16)
8
9   let l:brightness = (l:red * 299 + l:green * 587 + l:blue * 114) / 1000
10
11   return l:brightness > 0.5
12 endfunction

NORMAL code.vim
70 int dist(local_t *a, local_t *b) {
1   float x, y;
2   x = (float)a->localizacao.x - b->localizacao.x;
3   y = (float)a->localizacao.y - b->localizacao.y;
4
5   return (int)sqrt(x * x + y * y);
6 }
```

Catpuccin Frappé ↑

• Plugins

○ Plugins podem adicionar funcionalidades:

NERDtree adiciona um navegador de arquivos.

```
ss ? for help | 4 Aquele que quer aprender a v**oar um dia precisa primeiro#
. (up a dir) | 3 apren..der a ficar de pé, caminhar, correr, escalar e dançar;
</Documents/2023.1/bem-VIMdo/ | 2 ninguém consegue voar só aprendendo voo.
  • imagens_oficina/ | 1 Friedrich Nietzsche
    * modos/ | 5
      command.png | 1 Se você acha que é pequeno demais para causar impacto,
      insert.png | 2 tente ir para a cama com um mosquito.
      normal.png | 3 Anita Roddick
      visual.png | 4
      dois_modos_ampliado.png | 5 A diferença entre ganhar e perder na maioria das vezes é não desistir.
      dois_modos_com_legenda.p | 6 UAU Disney
      mnemonicos.png | 7
      todos_modos_com_legenda. | 8 Se você pensa que pode, ou pensa que não pode, você provavelmente está certo.
    navegando | 9 Henry Ford
    rascunho | 10
    README.md | 11 Primeiro princípio: nunca se deixar abater por pessoas ou eventos.
    recupera.sh* | 12 Marie Curie
    roteiro.txt | 13
                | 14 Lembre-se sempre, no entanto, de que geralmente há uma maneira mais simpler
```

● Plugins

○ Plugins podem adicionar funcionalidades:

LSP-Zero adiciona todo um ambiente para gerenciar LSPs:

```
1   for (i = 0; i <= N_LOCAIS; i++) {  
25  |   info->dados_missao[i] = de  
1   }  
2   }  
3  
4   /* Inicia struct informacoes_t  
5   informacoes_t *cria_info(){  
6       informacoes_t *informacoes  
7       if (!informacoes)  
8       |   return NULL;  
9       informacoes->dados_missao =  
10      if (informacoes->dados_miss  
11      |   free(informacoes);  
12      |   return NULL;  
13      }  
14      return informacoes;  
15  }
```

desaloca_missao...	f Function
destroi_cjt()	f Function
#deff~	↔ Snippet
dateMDY~	↔ Snippet
#def~	↔ Snippet
habilidades_das...	f Function
gera_tempo_desl...	f Function
destroi_lef()	f Function
datetime~	↔ Snippet
destroi_fila()	f Function
*dev_t~	⚡ Interface
date~	↔ Snippet

void
Desaloca o conjuntos dos ponteiros do vetor da struct informacoes_t.

AIS+1));

- Tarefa pra casa - vale a pena pesquisar:
 - Airline - Barrinha de status bonitinha
 - vim-devicons - Icones diversos que muitos outros plugins usam
 - NERDtree - Navegador de arquivos
 - ALE - Corretor de sintaxe e semântica assíncrono
 - VimTeX - LaTeX no Vim
 - NEOVIM!!

- Muito obrigado pelo seu tempo!



Bruno Aquiles de Lima

3º período.

github.com/Aquiles-b

bal22@inf.ufpr.br



Vinicius Evair da Silva

3º período.

github.com/evaiir

ves22@inf.ufpr.br