

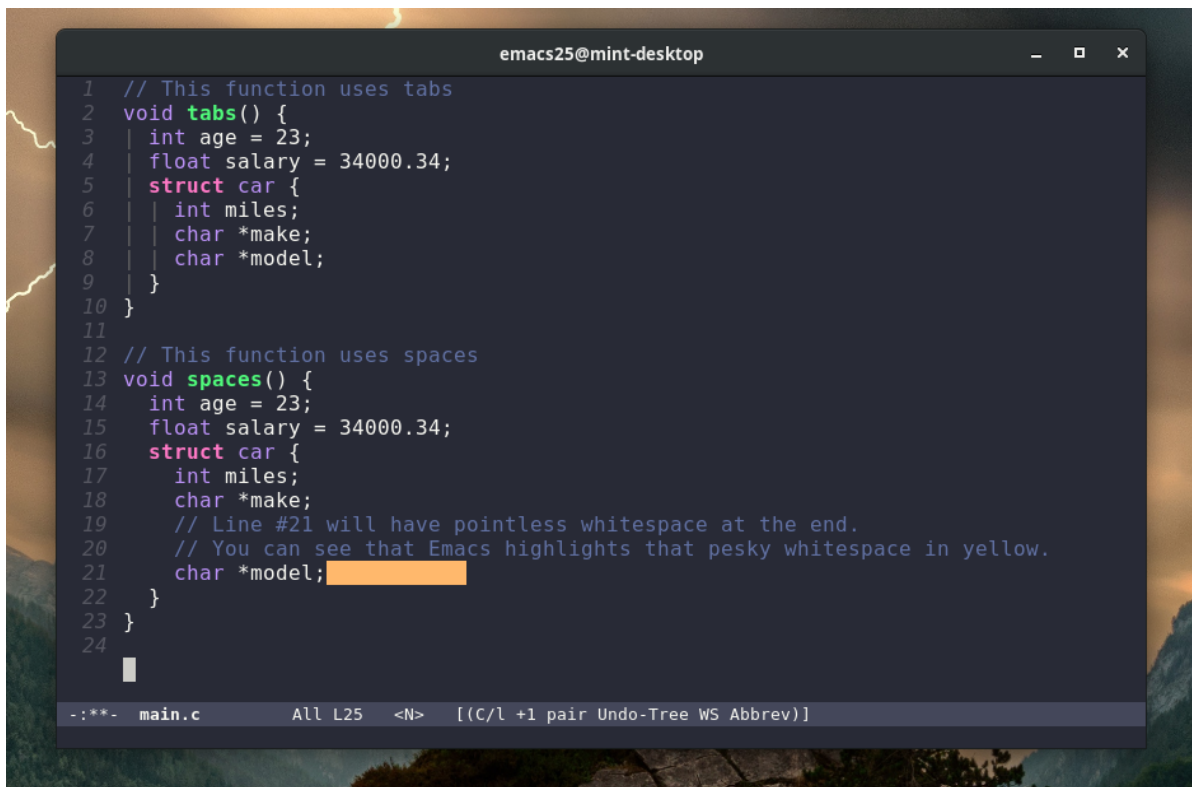
# O melhor Guia Para a Indentação no Emacs (Tabulações e Espaços)

---

SEGUNDA-FEIRA - 16 DE JULHO DE 2018

[fonte](#)

1. [TL;DR: A Configuração Completa:](#)
2. Quebrá-Lo Para Baixo
  1. [Funções para Ativar/Desativar guias](#)
  2. [Usando Tabulações ou Espaços em Arquivos Diferentes](#)
  3. [Alterar a largura de guia](#)
  4. [Fazer O Recuo De Se Comportar Sadia \(Elétrico Travessão\)](#)
  5. [Avanço de uma seleção para a esquerda ou para a direita](#)
  6. [Destacando Tabulações e Espaços de forma Diferente](#)
  7. [Fazendo Backspace Excluir Corretamente Guias](#)
  8. [SmartTabs \(Bônus\)](#)
  9. [Python Guias De Apoio](#)
  10. [Notas Finais](#)



```
1 // This function uses tabs
2 void tabs() {
3     int age = 23;
4     float salary = 34000.34;
5     struct car {
6         int miles;
7         char *make;
8         char *model;
9     }
10 }
11
12 // This function uses spaces
13 void spaces() {
14     int age = 23;
15     float salary = 34000.34;
16     struct car {
17         int miles;
18         char *make;
19         // Line #21 will have pointless whitespace at the end.
20         // You can see that Emacs highlights that pesky whitespace in yellow.
21         char *model;
22     }
23 }
24
```

## TL;DR: A Configuração Completa:

Esta configuração serve para usuários que preferem guias sobre espaços. Para saber como personalizar tabulações e espaços comportamento diferente, por favor, consulte a [quebrá-Lo para Baixo](#) da seção.

```
; START TABS CONFIG
;; Create a variable for our preferred tab width
(setq custom-tab-width 2)

;; Two callable functions for enabling/disabling tabs in Emacs
(defun disable-tabs () (setq indent-tabs-mode nil))
(defun enable-tabs ()
  (local-set-key (kbd "TAB") 'tab-to-tab-stop)
  (setq indent-tabs-mode t)
  (setq tab-width custom-tab-width))

;; Hooks to Enable Tabs
(add-hook 'prog-mode-hook 'enable-tabs)
;; Hooks to Disable Tabs
(add-hook 'lisp-mode-hook 'disable-tabs)
(add-hook 'emacs-lisp-mode-hook 'disable-tabs)
```

```

;; Language-Specific Tweaks
(setq-default python-indent-offset custom-tab-width) ;; Python
(setq-default js-indent-level custom-tab-width)      ;;
Javascript

;; Making electric-indent behave sanely
(setq-default electric-indent-inhibit t)

;; Make the backspace properly erase the tab instead of
;; removing 1 space at a time.
(setq backward-delete-char-untabify-method 'hungry)

;; (OPTIONAL) Shift width for evil-mode users
;; For the vim-like motions of ">>" and "<<".
(setq-default evil-shift-width custom-tab-width)

;; WARNING: This will change your life
;; (OPTIONAL) Visualize tabs as a pipe character - "|"
;; This will also show trailing characters as they are useful to
spot.
(setq whitespace-style '(face tabs tab-mark trailing))
(custom-set-faces
 '(whitespace-tab ((t (:foreground "#636363")))))
(setq whitespace-display-mappings
 '((tab-mark 9 [124 9] [92 9]))) ; 124 is the ascii ID for '|'
(global-whitespace-mode) ; Enable whitespace mode everywhere
; END TABS CONFIG

```

## Quebrá-Lo Para Baixo

---

### Funções para Ativar/Desativar guias

```
;; Our Custom Variable
(setq custom-tab-width 2)

(defun disable-tabs () (setq indent-tabs-mode nil))
(defun enable-tabs ()
  (local-set-key (kbd "TAB") 'tab-to-tab-stop)
  (setq indent-tabs-mode t)
  (setq tab-width custom-tab-width))
```

Estas são duas funções diferentes, que podemos chamar de nosso costume de ganchos. Se você também quiser activar estas funções na mosca, usando `M-x`, você pode usar a `(interactive)` função em Emacs Lisp. [Aqui está a documentação](#).

A primeira coisa que fazemos na `enable-tabs` função é definir a tecla de TABULAÇÃO para `tab-to-tab-stop`. Na minha opinião, é uma mais sã padrão. Quando você pressiona a tecla tab, vai recuar um guia como o esperado. Não há magia ou confusão o que vai acontecer quando você pressionar a tecla tab.

Depois disso, vamos habilitar o [travessão-guias-modo](#) e definir o nosso [guia de largura](#).

## Usando Tabulações ou Espaços em Arquivos Diferentes

```
(add-hook 'prog-mode-hook 'enable-tabs)

(add-hook 'lisp-mode-hook 'disable-tabs)
(add-hook 'emacs-lisp-mode-hook 'disable-tabs)
```

Depois de criar [os ganchos](#), é muito fácil decidir quais tipos de arquivos/modos de nós deseja habilitar guias/espacos.

Neste exemplo, podemos habilitar guias no [prog-mod](#). Prog-mode é um bom gancho para usar se pretender efectuar definições para praticamente todos os tipos de arquivos de código.

Depois que nós desativar guias (utilize espaços) em Lisp e o diretório elisp arquivos. Lisp é um tipo especial de linguagem de programação que realmente não funciona bem com guias, então eu recomendo espaços. Por que é que, exatamente? O conceito de recuo realmente não existe em Lisp. É tudo sobre o alinhamento, e guias de parafuso até o alinhamento preciso que o Lisp e requer.

## Alterar a largura de guia

```
;; Our Custom Variable
(setq custom-tab-width 2)

(setq-default python-indent-offset custom-tab-width)
(setq-default evil-shift-width custom-tab-width)
```

Lembre-se de não definir a variável `custom-tab-width` duas vezes! Ele deve ser definido acima ativar/desativar guias funções. Eu incluído neste trecho de código apenas para fins de ilustração.

Neste exemplo, definimos a largura de guia para o nosso guia personalizado largura variável. A primeira linha usa o básico de `tab-width` propriedade. Eu recomendo a não definição do guia de largura na forma como eu descrevi acima e, em vez de colocá-lo dentro de uma função, para ser usado em ganchos. [Detalhes sobre como fazer exatamente isso](#).

A seguir vamos definir o Python travessão tamanho para o nosso guia personalizado largura variável. Eles fazem 4 espaços por padrão para cumprir com [pep8](#), mas se você preferir ter seu próprio valor predefinido, pode alterá-la através dessa variável.

Por fim, definimos o mal-shift-largura para a nossa guia personalizado largura variável. Isto só é útil se estiver a utilizar o Mal pacote para obter Vim-como combinações de teclas em vez de usar glorioso Emacs combinações de teclas. `evil-shift-width` controla o tamanho do separador quando você estiver usando o `>>` ou `<<` movimento de avanço ou de recuo de texto.

## Fazer O Recuo De Se Comportar Sadia (Elétrico Travessão)

Algo que estava dirigindo-me louco foi o Emacs elétrica-recuo recuo da linha anterior, quando eu pressionar enter. Felizmente, me deparei com uma correção de um dia.

```
(setq-default electric-indent-inhibit t)
```

## Avanço de uma seleção para a esquerda ou para a direita

Para fazer isso, consulte o meu [Como Avanço de uma Seleção no Emacs](#) guia.

## Destacando Tabulações e Espaços de forma Diferente

Algo que eu sinto que é muito importante ter um editor é uma forma de identificar os espaços e tabulações facilmente. Eles são ambos caracteres de espaço em branco que pode ser facilmente confundido um para o outro.

Vamos fazer isso, fazer guias aparecem visível como uma "|" (pipe) de caracteres. Se você quiser realçar espaços demais, por favor, consulte [ErgoEmacs " artigo em fazer espaço em branco visível](#).

```
(global-whitespace-mode)
(setq whitespace-style '(face tabs tab-mark trailing))
(custom-set-faces
 '(whitespace-tab ((t (:foreground "#636363")))))

(setq whitespace-display-mappings
 '((tab-mark 9 [124 9] [92 9])))
```

Eu tenho separado a `whitespace-display-mappings` parte do trecho com uma nova linha, porque é o mais confuso de ler a parte dele.

As quatro primeiras linhas do trecho são muito fáceis de entender.

1. Primeiro você permitir `global-whitespace-mode`. Isso faz com que nosso configurado espaço visível em todos os buffers automaticamente.
2. Ao lado, você está definindo o espaço em branco de estilo para mostrar abas à esquerda e à direita o espaço em branco. Você precisa `face` e `tabs` incluída existe. Eu não estou completamente certo porque, mas eles são necessários para definir a cor do caractere de pipe.
3. Depois disso, você está chamando `custom-set-faces` para personalizar a aparência da guia de espaço em branco. Eu definir a cor do texto de nosso tubo de caracteres para ser **#636363** que é uma cor agradável se estiver a utilizar um escuro temática Emacs tema. Se você estiver usando um tema claro no Emacs, que você vai querer uma cor mais clara como **#c1c1c1**.

This is an example of #636363 on a dark background. It is supposed to be subtle.

Este é um exemplo de #c1c1c1 uma luz no fundo. É suposto para ser sutil.

A linha que é um pouco confuso para ler é onde nós realmente definir o caractere de pipe.

Tudo o que você precisa saber sobre ele é que `124` é o ascii ID do caractere de pipe ("|"). Você pode exibir [uma lista de caracteres ascii IDs aqui](#).

## Fazendo Backspace Excluir Corretamente Guias

O Emacs tem um estranho comportamento padrão quando backspacing guias. Em vez de backspacing todo o guia, ele retrocede a aba de um espaço ao mesmo tempo.

Você pode corrigir isso da seguinte maneira.

```
(setq backward-delete-char-untabify-method 'hungry)
```

## SmartTabs (Bônus)

O [Smart-guias-de modo](#) pacote de ajuda Emacs travessão, com guias e alinhar-se com espaços em vários idiomas.

Eu, pessoalmente, não uso, mas pode agradar a sua fantasia.

Se você não tiver certeza de como instalar pacotes no Emacs, que você pode se referir a [esta wiki entrada](#) ou [este vídeo](#).

Depois de instalá-lo, você pode habilitá-lo em várias línguas, como segue:

```
(smart-tabs-insinuate 'c 'javascript 'python)
```

## Python Guias De Apoio

UPDATE: Não costumava ser muito tedioso processo para obter guias funcionando corretamente em Python. Felizmente, depois de atualizar este artigo com os meus novos guias de configuração, ele funciona bem. Atualize o seu código com [o meu código de novo](#) e você vai estar



tudo pronto.

Você também pode querer verificar para fora [SmartTabs](#) também.