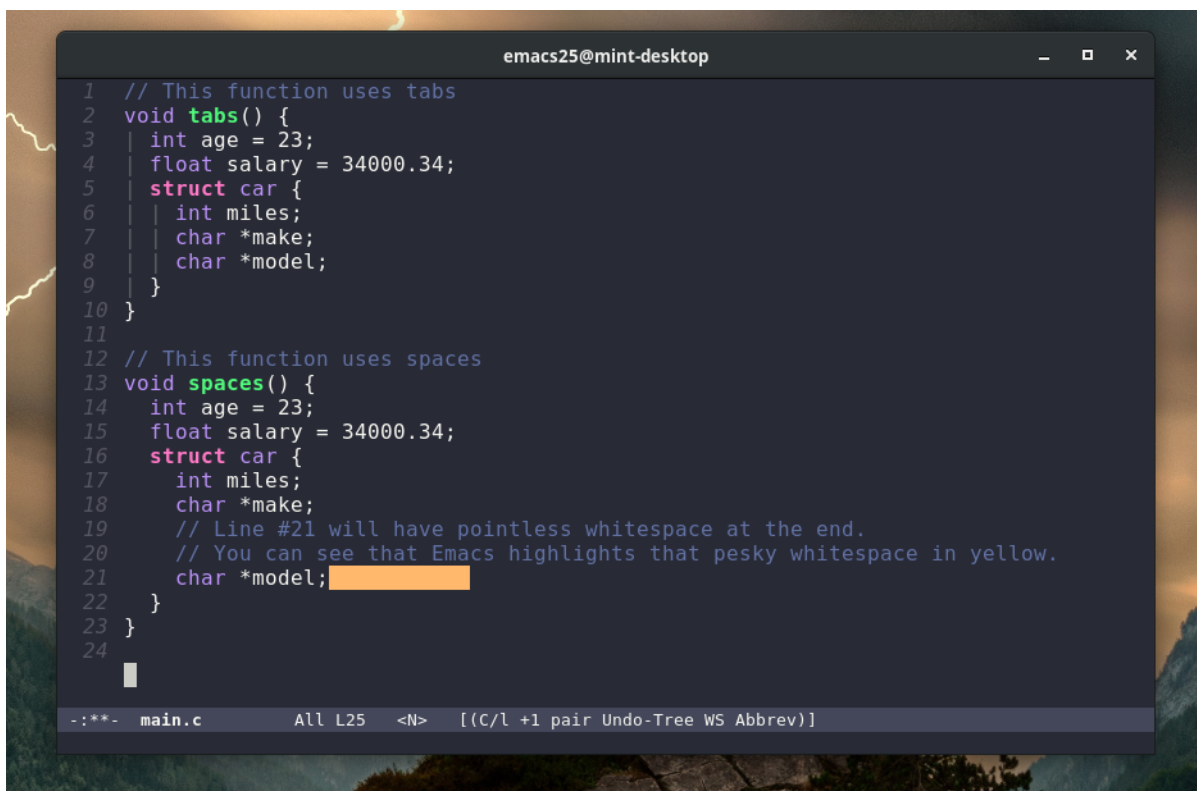


O guia definitivo para recuo no Emacs (tabulações e espaços)

[fonte](#)

SEGUNDA-FEIRA - 16 DE JULHO DE 2018

1. [DR: A configuração completa:](#)
2. Quebrando tudo
 1. [Funções para ativar/desativar guias](#)
 2. [Usando Tabulações ou Espaços em Arquivos Diferentes](#)
 3. [Alterando a largura da aba](#)
 4. [Fazendo o recuo se comportar de maneira saudável \(recuo elétrico\)](#)
 5. [Recuar uma seleção para a esquerda ou para a direita](#)
 6. [Destacando Tabulações e Espaços de Forma Diferente](#)
 7. [Fazendo o Backspace apagar as abas corretamente](#)
 8. [SmartTabs \(Bônus\)](#)
 9. [Suporte para guias Python](#)
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
```

DR: A configuração completa:

Esta configuração é destinada a usuários que preferem tabulações a espaços. Para aprender como personalizar o comportamento de guias e espaços de maneira diferente, consulte a seção Dividindo.

```
; 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
```

Quebrando tudo

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))
```

Essas são duas funções diferentes que podemos chamar facilmente em nossos hooks personalizados. Se você também quiser ativar essas funções na hora, usando `M-x`, você pode usar a `(interactive)` função no Emacs Lisp. [Aqui está a documentação](#).

A primeira coisa que fazemos na `enable-tabs` função é definir a tecla TAB para `tab-to-tab-stop`. Na minha opinião, é um padrão mais sensato. Quando você pressiona a tecla tab, ela recua uma tabulação conforme o esperado. Não há nenhuma mágica ou confusão sobre o que acontecerá em seguida quando você pressionar a tecla tab.

Depois disso, ativamos [o modo indent-tabs e definimos nosso tab-width](#) personalizado.

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 [esses ganchos](#), é muito fácil decidir em quais tipos de arquivos/modos queremos habilitar tabulações/espacos.

Neste exemplo, ativamos as guias no [modo prog](#). O modo Prog é um bom gancho para usar se você quiser fazer configurações para praticamente todos os tipos de arquivos de código.

Depois disso desabilitamos tabulações (usamos espaços) em arquivos Lisp e ELisp. Lisp é um tipo especial de linguagem de programação que realmente não funciona bem com abas, então eu recomendo espaços. Por que isso acontece exatamente? O conceito de recuo realmente não existe no Lisp. É tudo uma questão de alinhamento, e as guias atrapalharão o alinhamento preciso que o Lisp exige.

Alterando a largura da aba

```
;; 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! Deve ser definido acima das funções de ativar/desativar guias. Eu o incluí neste trecho de código apenas para fins ilustrativos.

Neste exemplo, definimos a largura da guia como nossa variável de largura de guia personalizada. A primeira linha usa a `tab-width` propriedade básica. Eu recomendo não definir a largura da guia da maneira que descrevi acima e, em vez disso, colocá-la dentro de uma função, para ser usada em ganchos. [Detalhes sobre como fazer exatamente isso](#).

A seguir, definimos o tamanho do recuo do Python para nossa variável de largura de guia personalizada. Eles fazem 4 espaços por padrão para cumprir [pep8](#), mas se você preferir ter seu próprio valor padrão, você pode alterá-lo através dessa variável.

Por fim, definimos o `evil-shift-width` para nossa variável de largura de tabulação personalizada. Isso só é útil se você estiver usando o pacote Evil para obter atalhos de teclado semelhantes ao Vim em vez de usar atalhos de teclado gloriosos do Emacs. `evil-shift-width` controla o tamanho da tabulação quando você está usando o movimento `>>` or `<<` para recuar ou desrecuar texto.

Fazendo o recuo se comportar de maneira saudável (recuo elétrico)

Algo que estava me deixando louco era o `electric-indent` do Emacs recuando a linha anterior quando eu pressionava enter. Felizmente, um dia, encontrei uma solução.

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

Recuar uma seleção para a esquerda ou para a direita

Para fazer isso, consulte meu guia [Como recuar uma seleção no Emacs](#).

Destacando Tabulações e Espaços de Forma Diferente

Algo que eu acho muito importante ter em um editor é uma maneira de identificar espaços e tabulações facilmente. Ambos são caracteres de espaço em branco que podem ser facilmente confundidos um com o outro.

Faremos isso fazendo com que as guias apareçam visíveis como um “|” (tubo) personagem. Se você quiser destacar espaços também, consulte o [artigo do ErgoEmacs sobre como tornar os espaços em branco visíveis](#).

```
(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])))
```

Separei a `whitespace-display-mappings` parte do trecho com uma nova linha porque é a parte mais confusa de ler.

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

1. Primeiro você habilita `global-whitespace-mode`. Isso torna nosso espaço em branco configurado visível em todos os buffers automaticamente.
2. Em seguida, você está configurando o estilo do espaço em branco para mostrar guias e espaços em branco à direita. Você precisa `face` e `tabs` está incluído lá. Não sei bem por que, mas eles são necessários para definir a cor do caractere de barra vertical.
3. Depois disso, você está ligando `custom-set-faces` para personalizar a aparência do espaço em branco da guia. Eu defini a cor do texto do nosso caractere pipe como **#636363**, que é uma cor legal se você estiver usando um tema Emacs com tema escuro. Se estiver usando um tema claro no Emacs, você desejará uma cor mais clara, como **#c1c1c1**.

Este é um exemplo de #636363 em um fundo escuro. É suposto ser sutil.

Este é um exemplo de #c1c1c1 em um fundo claro. É suposto ser sutil.

A linha que é um pouco confusa de ler é aquela onde realmente definimos a barra vertical.

Tudo o que você realmente precisa saber sobre isso é `124` o ID ascii do caractere vertical ("|"). Você pode ver [uma lista de IDs de caracteres ASCII aqui](#).

Fazendo o Backspace apagar as abas corretamente

O Emacs tem um comportamento padrão estranho ao retroceder as guias. Em vez de retroceder a guia inteira, ele retrocede a guia um espaço por vez.

Você pode corrigir isso da seguinte maneira.

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

SmartTabs (Bônus)

O pacote [Smart-tabs-mode](#) ajuda o Emacs a recuar com tabulações e alinhar com espaços em vários idiomas.

Eu pessoalmente não uso, mas pode agradar sua imaginação.

Se você não tiver certeza de como instalar pacotes no Emacs, consulte [esta entrada do wiki](#) ou [este vídeo](#).

Depois de instalá-lo, você pode habilitá-lo em vários idiomas da seguinte forma:

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

Suporte para guias Python

ATUALIZAÇÃO: Costumava haver um processo bastante tedioso para fazer as guias funcionarem corretamente em Python.

Felizmente, depois de atualizar este artigo com minha configuração de novas guias, ele funciona bem. Atualize seu

código com [meu novo código](#) e estará tudo pronto.

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

Notas Finais

É isso! Deixe-me saber se você tiver dúvidas sobre alguma dessas coisas ou tiver uma sugestão para melhorá-lo.