Emacs em 30 Segundos

Guaracy Monteiro

Sumário

TL;DR

Arquivo de configuração para ser utilizado em uma instalação nova do Emacs ou substituir uma antiga (excluir **.emacs** e **.emacs.d**). Automaticamente instala alguns pacotes definidos e configura o ambiente e os pacotes. Tudo bem explicadinho para ser alterado//melhorado facilmente.

1. Grave o arquivo .emacs, no seu diretório **home**. Se desejar, grave também o arquivo .myemacs no seu diretório **home**. Abra o Emacs e espere a configuração terminar.
2. Baixe e leia a documentação no formato desejado: README.pdf, README.epub, README.doc, README.odt.

Emacs em 30 segundos

A ideia base deste projeto é a possível dificuldade de instalação e configuração do Emacs por parte dos usuários. A instalação é simples mas, conforme o programa sai de *fábrica*, muitas pessoas encontram uma pequena dificuldade inicial e, como pensam: "- *Ah, é só mais um editor de textos. Vou continuar com o que eu já conheço.*", e perdem uma ferramenta das mais poderosas que poderiam usar. Apesar de ser possível efetuar algumas alterações pelo menu, muitas necessitam que o usuário edite um arquivo de configuração (**.emacs**). A instalação de novos pacotes, por exemplo, já foi uma tarefa bem mais complicada.

Alternativas

Existem algumas opções que possuem um Emacs configurado para o usuário final. É possível citar:

1. Spacemacs : É uma ótima alternativa e bastante indicada para usuários do Vim. Existem pessoas que tentaram usar o Emacs mas voltaram para o Vim. Depois de instalar o Spacemacs, estão usando é achando ótimo. Basicamente é uma cópia do diretório **.emacs.d** com toda a configuração automática. O usuário precisará ler a documentação se desejar fazer alguma alteração na configuração.
2. super-emacs : Também é uma cópia do diretório **.emacs.d** porém com uma configuração mais simples que o Spacemacs.
3. Ergoemacs : Deve ser uma ótima configuração para o Windows. No Linux é difícil de testar pois muitos atalhos conflitam com o desktop (KDE, Gnome, etc).
4. Emacs30 : Este que você está lendo. Baseia-se apenas em uma arquivo de configuração (**.emacs**) que deve ser gravado no diretório home do usuário. Ele irá baixar e configurar os diversos pacotes. Ao término, ele verifica se existe o arquivo (**.myemacs**). Se existir, continua a configuração por este arquivo.

Vejo como principal desvantagem dos itens 1 e 2 a não habilitação do CUA-mode. Esconder o menu principal também pode ser estranho para alguns usuários iniciantes. O item e eu não testei. A habilitação do CUA-mode, entre outras coisas, permite combinações que a maioria dos usuários está acostumada como: *Ctrl+c Ctrl+c* para copiar, *Ctrl+c Ctrl+v* para colar e *Ctrl+z* para desfazer.

Inicialmente o Emacs possui uma tela assim:

Ou seja, uma barra de ferramentas de gosto duvidoso (depois de um breve período de uso você vai achar melhor os atalhos do que tirar a mão do teclado, pegar o mouse e clicar em um botão) e uma tela inicial com muita informação, um *C-x* indicando que o usuário pressionou a sequencia *Ctrl+x*. Ficará assim:

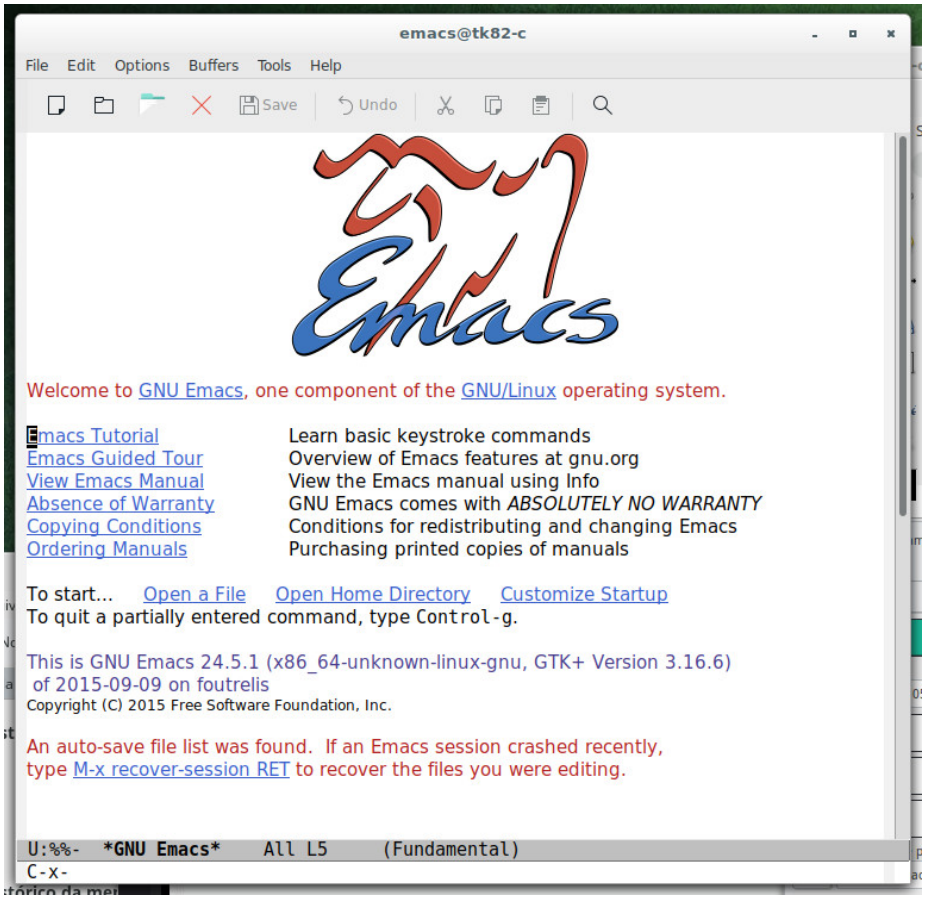

Figura 1: Tela inicial do Emacs

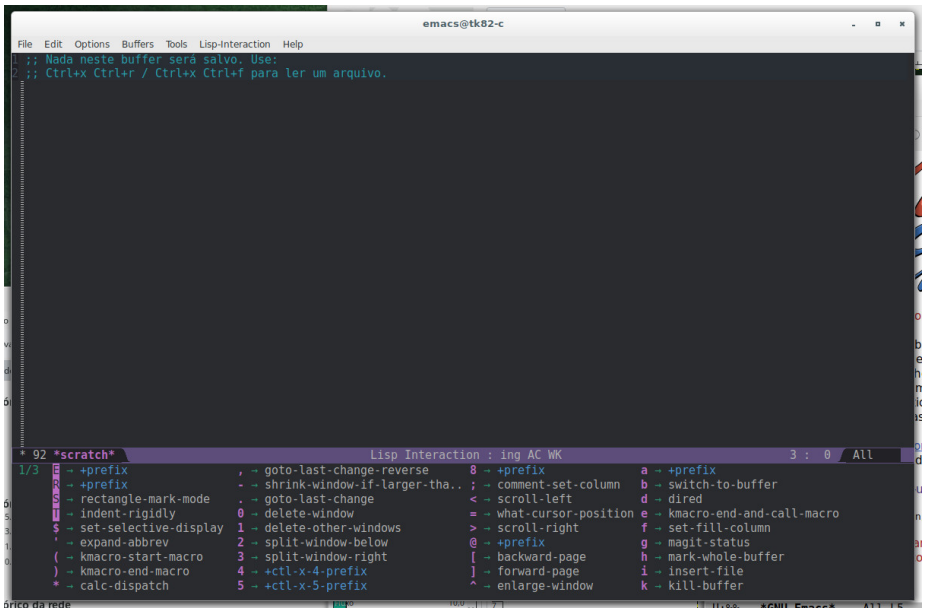

Figura 2: Tela final do Emacs

Uma tela mais limpa (o esquema de cores dependerá do escolhido pelo usuário), uma linha de status mais agradável e, se pressionar *Ctrl+x* uma relação das alternativas possíveis para complementar o comando.

Características do projeto

1. Habilitação do CUA-mode para que o usuário não precise aprender que *Alt+w* é usada para copiar um texto, por exemplo.
2. Alterar a aparência do Emacs, utilizando um tema com fundo escuro. O usuário poderá optar por um fundo claro a qualquer momento
3. Deixar a barra de status com uma aparência mais agradável.
4. Alterar diversas opções para deixar o Emacs mais amigável(?).
5. Instalar alguns pacotes para facilitar o trabalho do usuário em diversas áreas, podendo inclusive gerar arquivos .docx (Microsoft Word), .odt (LibreOffice), .pdf, .epub (eBook) entre diversos outros de forma simples (pode necessitar da instalação de programas externos como por exemplo, para gerar .pdf é necessário instalar o **MiKTeX** no Windows ou o **texlive** no Linux).

org-mode

Inicialmente estava previsto a instalação do pacote **markdown** para facilitar a criação e transformação de textos em uma linguagem de marcação simples para outros documentos mais complexos. Como o Emacs já vem com o org-mode e muitos usuários utilizam o Emacs basicamente pelo pacote, nada melhor do que utilizá-lo.

1. Utiliza uma linguagem de marcação relativamente simples.
2. É possível utilizar como arquivo de entrada e exportar para diversos formatos.
3. De brinde, o usuário ganha um poderoso organizador pessoal que permite até a sincronização dos dados com seu smartphone ou tablet.
4. Por último mas não menos importante, o GitHub aceita diretamente um arquivo README.org como o README.md.

Instalação

Copiar o arquivo .emacs e .myemacs para o diretório home (Linux : ~/ ou /**home**/**usuário**; Windows: *c:
Usuários
usuário
AppData
Rooming
) do usuário (caso já exista uma arquivo **.emacs** e/ou **.myemacs**, faça um backup por garantia). Apague o diretório **.emacs.d** (também faça backup se for o caso). Depois é só executar o Emacs que os pacotes serão baixados e instalados.

Arquivo .emacs

Configurações iniciais

Pacotes

É onde tudo acontece. Achei que seria melhor explicar com mais detalhes tudo o que acontece durante a execução do arquivo para que o usuário possa efetuar alterações básicas para deixar o Emacs mais ao seu gosto. As linhas que iniciam com ponto e vírgula indicam que são comentários e não serão interpretadas. Para um entendimento melhor, seria interessante que o usuário aprendesse um pouco sobre a linguagem **emacs-lisp** (uma variação de lisp) de onde vem toda a flexibilidade do Emacs.

Alterações das opções iniciais

Deixei estas alterações no início pois, se for feita alguma alteração utilizando o menu **Options** e o usuário selecionar **Options/Save Options**, esta parte do arquivo **.emacs** será alterada. Ficando no início é mais fácil de visualizar e não causa tanta confusão.

```
(custom-set-variables
 '(cua-mode t nil (cua-base)) (ref:cua)
 '(custom-enabled-themes (quote (misterioso))) (ref:theme)
 '(indicate-empty-lines t) (ref:empty)
 '(show-paren-mode t) (ref:paren)
 '(tool-bar-mode nil)) (ref:tool)
```

Ativamos o CUA-mode *(cua)*, inicializamos um tema (cores utilizadas para fundo, fontes e salientar diversas sintaxes no texto) diferente do original *(theme)*, indicamos que linha vazias devem conter um símbolo no início para diferencia de linhas que possuam espaço *(empty)*, dizemos que queremos uma visualização para abertura e fechamento de chaves, parêntesis e colchetes (muito útil para programação)*(paren)* e, finalmente, que não desejamos ver a barra de ferramentas (as teclas de atalho são mais eficientes e nada que dois níveis do menu não resolvam) *(tool)*.

Inclusão e atualização de fonte de pacotes

```
(require 'package)
(add-to-list 'package-archives
             '("melpa" . "http://melpa.milkbox.net/packages/")
             t)
(package-initialize)
```

Adiciona o repositório MELPA que contém um maior número de pacotes e com uma atualização constante.

```
(when (not package-archive-contents)
  (package-refresh-contents))
```

Atualiza o conteúdo das fontes de pacotes se não existe. Para você atualizar os pacotes, utilize o menu **Options/Manage Emacs Packages**. Na janela de gerenciamento de pacotes, pressione **U** para atualizar os pacotes (irá excluir o anterior e instalar a versão nova), **I** para instalar algum pacotes desejado (veja **.myemacs**), **D** para excluir algum pacote (atenção para o que você excluir) e,

quando tudo estiver pronto, pressione **X** para executar as ações de inclusão e exclusão.

Seleção e instalação dos pacotes pelo Emacs30

```
(defvar gbm-required-packages
  '(which-key
    hl-line+
    powerline
    hlinum
    hiwin
    ido-grid-mode
    ido-select-window
    imenu-anywhere
    smex
    pandoc-mode
    org-cua-dwim
    org-pandoc
    auto-complete
    smartparens
    goto-chg
    indent-guide
    theme-looper))
```

Não inclua nenhum pacote neste ponto. Utilize o arquivo **.myemacs** se deseja incluir outros pacotes. Para saber mais sobre cada pacote especificado, você pode ir no MELPA, digitar o nome do pacote e clicar no link da coluna *source*. Na grande maioria das vezes, você irá para uma página com informações do pacote. No gerenciador de pacotes do Emacs, **Options/Manage Emacs Packages**, também existem informações sobre a finalidade.

```
(mapc (lambda (p)
        (package-install p))
      gbm-required-packages)
```

Configurações da aparência e dos pacotes

Tamanho da janela

```
(setq initial-frame-alist
      '(
        (width . 130)
        (height . 40)
        ))
```

Especifica uma altura/largura maior do que os valores padrões. Em muitos casos, é melhor maximizar a janela para poder trabalhar com mais de um frame e uma boa visibilidade de cada um.

Which key

```
(which-key-mode)
(which-key-setup-minibuffer)
(setq max-mini-window-height 10)
(setq which-key-idle-delay 0.5)
(set-face-attribute
  'which-key-local-map-description-face nil
  :weight 'bold)
```

Quando o usuário utilizar um atalho como *Ctrl+c* ou *Ctrl+h*, por exemplo, e não digitar o complemento dentro de 0.5 segundos, o minibuffer irá mostrar as possibilidades existentes para completar o comando. Foi configurado que o minibuffer terá 10 linhas de altura, o tempo de espera é de 0,5 segundos e as combinações válidas para o buffer onde o usuário está trabalhando estarão em negrito.

Numeração das linhas

```
(global-linum-mode t)
```

Indica para numerar as linhas em todos os buffers. A qualquer momento o usuário poderá alterar pressionando *Alt+x linum-mode*.

Realça linha do cursor

```
(hl-line-mode t)
(toggle-hl-line-when-idle)
(set-face-attribute hl-line-face nil :background "Grey25")
(set-cursor-color "yellow")
```

Irá realçar a linha onde encontra-se o cursor apenas quando o usuário não estiver fazendo nada. Escolhi *Grey25* como cor de fundo e *yellow* para a cor do cursor. Para ver as cores, suas combinações bem como o nome, basta entrar com *Alt+x list-colors-display*

Realça numeração da linha do cursor

```
(require 'hlinum)
(hlinum-activate)
```

O realce de linha não realça a numeração da linha. A função do *hlinum* é para realçar o número da linha. Sempre será realçada, independente do programa estar em espera.

Realçar parêntesis

```
(show-paren-mode)
```

Realça os respectivos pares de parêntesis, chaves ou colchetes.

Esconde barra de rolamento

```
(scroll-bar-mode -1)
```

Esconde a barra de rolamento do frame. A barra de status já possui informações sobre inicio ou final de arquivo ou percentual que já foi rolado. Também possui um pequeno ícone mostrando a posição relativa (como uma mini barra de rolamento). Ganhamos mais um pouco de espaço na horizontal e menos um elemento para distrair.

Salva estado atual ao sair

```
(require 'saveplace)
(setq-default save-place t)
(setq save-place-file (expand-file-name
  ".places" user-emacs-directory))
```

Salva a posição atual do cursor no arquivo. Na próxima vez que for aberto, será posicionado na posição que estava antes de encerra.

Desabilita buffer de mensagem inicial

```
(setq initial-buffer-choice
    t)
```

Desabilita a tela de abertura que contém diversas informações desnecessárias.

Troca mensagem do buffer de rascunho

```
(setq initial-scratch-message
    ";; Nada neste buffer será salvo. Use:\n;;
    Ctrl+x Ctrl+r / Ctrl+x Ctrl+f para ler um arquivo.\n")
```

Alterei a mensagem do buffer de rascunho. Nada do que for escrito nele será salvo automaticamente ao sair. Buffers contendo arquivos, se forem alterados e ainda não foram salvos ao encerrar o programa, será mostrada uma mensagem informando que os dados não foram salvos e se o usuário deseja sair, salvar ou cancelar.

Configura powerline

```
(powerline-center-theme)
(setq powerline-default-separator
      'wave)
```

Confere uma apresentação melhor para a linha de status.

ido no modo grade

```
(setq ido-enable-flex-matching t)
(setq ido-everywhere t)
(ido-mode t)
(ido-grid-mode t)
(global-set-key (kbd "C-x o") 'ido-select-window)
(global-set-key (kbd "<f4>") 'ido-select-window)
```

IDO (InteractivelyDoThings) mostra as opções disponíveis no minibuffer facilitando a escolha pelo usuário. Se for informado o comando para abrir um arquivo (*Ctrl+x Ctrl+f*) por exemplo, será aberto um frame com a relação dos arquivos e diretórios para que seja feita a escolha. A última escolha sempre aparecerá em primeiro lugar. O usuário poderá usar as setas e enter para selecionar o arquivo ou poderá ir digitando o nome do arquivo ficando visíveis apenas os que coincidirem com o digitado. Se o diretório tiver diversos arquivos com o nome *temp* e extensões diferentes (supondo-se que nenhum inicie com o caractere *t*), basta digitar *t* e parte da extensão: *ttex* selecionará todos os arquivos que possuam a extensão iniciando com /tex'.

Configura atalho Ctrl+. para imenu-anywhere

```
(global-set-key (kbd "C-.") 'imenu-anywhere)
```

Mostra no minibuffer, via IDO, o que o programa acha que é interessante para que o usuário possa movimentar-se com mais rapidez no arquivo. Nome de funções e procedimentos no caso de programas, o que for considerado título em arquivos texto, etc.

Configura atalhos Alt+x e Alt+X para smex

```
(global-set-key (kbd "M-x") 'smex)
(global-set-key (kbd "M-X") 'smex-major-mode-commands)
```

Se o usuário digitar *Alt+x*, será apresentado no minibuffer via IDO, uma seleção das possíveis complementações.

Configura o autocomplete

```
(ac-config-default)
(ac-linum-workaround)
```

Apresenta um menu para completar automaticamente a digitação de funções e procedimentos em programas. Quando existente, apresenta uma janela de auxílio sobre a função//procedimento atual.

Indent guide

```
(indent-guide-global-mode)
```

Mostra barras verticais para mostras a endentação em programas.

Configura theme-looper

```
(theme-looper-set-customizations 'powerline-reset)
(global-set-key (kbd "C-\"") 'theme-looper-enable-next-theme)
```

Permite que o usuário passeie pelos temas especificado para verificar algum que lhe agrade mais. Para alterar definitivamente, uma das opções é ir no menu **Options/Customize Emacs/Custom Themes**.

Configura goto last change

```
(global-set-key (kbd "C-x .") 'goto-last-change)
(global-set-key (kbd "C-x ,") 'goto-last-change-reverse)
```

Permite que o usuário pule nas últimas alterações Pressionando a combinmação /Ctrl+,/ e *Ctrl+.*.

Ctrl+x Ctrl+r abre lista de arquivos recentes

```
(require 'recentf)
(recentf-mode t)
```

```
(setq recentf-max-menu-items 25)
(defun recentf-ido-find-file ()
  "Find a recent file using Ido."
  (interactive)
  (let ((file (ido-completing-read "Choose recent file: "
    recentf-list nil t)))
    (when file
      (find-file file))))
(global-set-key (kbd "C-x C-r") 'recentf-ido-find-file)
```

Utilizando *Ctrl+x Ctrl+r* permite que o usuário abra um minibuffer para escolher entre os últimos arquivo editados.

Carrega arquivo .myemacs

```
(setq myconfig "~/.myemacs")
(if (file-exists-p myconfig)
    (load-file myconfig))
```

Informa para ler o conteúdo do arquivo **.myemacs** se ele existir. Deverá conter outras configurações desejadas pelo usuário. Não colocá-las no arquivo **.emacs**.

Define F3 para pesquisar e Shift+F3 para pesquisar próxima

```
(global-set-key (kbd "C-f") 'isearch-forward)
(define-key isearch-mode-map (kbd "<f3>")
  'isearch-repeat-forward)
(define-key isearch-mode-map (kbd "S-<f3>")
  'isearch-repeat-backward)
```

Permite que o usuário digite *Ctrl+f* para efetuar uma pesquisa ou invés de *Ctrl+s* que é o padrão do Emacs. Pressionando *F3* vai para a próxima ocorrência e *Shift+F3* para a ocorrência anterior.

Configurar o org-mode

```
(setq org-CUA-compatible t)
(setq org-support-shift-select t)
```

```
(setq org-src-fontify-natively t)
(setq org-startup-truncated nil)
(setq org-use-speed-commands t)
```

Apenas algumas configurações (existem muitas outras disponíveis). Uma melhor compatibilização do org-mode com o CUA-mode (shift setas para selecionar, por exemplo). Quebra de linhas no final da janela para não ser necessário rolar para ver a continuação. Colorizar fontes (utiliza htmlize). Speed commands para facilitar o trabalho com o org-mode. Estando no inicio de um título, por exemplo, é possível pressionar apenas **j** para saltar para os diverso títulos, **n** ou **p** para saltar para o tópico seguinte ou anterior e mais diversas facilidades.

Arquivo .myemacs

É aconselhável que toda as alterações efetuadas pelo usuário estejam neste arquivo e não no **.emacs**. Facilita a vida do usuário em caso de atualização do arquivo **.emacs**. Abaixo um exemplo de conteúdo.

Instalação de outros pacotes

```
(package-refresh-contents)
(mapc (lambda (p)
    (package-install p))
      '(magit
        heroku-theme
    gruvbox-theme
        material-theme
    spacemacs-theme
    subatomic-theme
    tangotango-theme
    paradox
    ))
(load-theme 'spacemacs-dark t)
(global-set-key (kbd "C-x g") 'magit-status)
```

Exemplo de instalação de outros pacotes que o usuário deseja. Aqui instalamos o *magit* para facilitar o trabalho com o git e o *paradox* que é um gerenciador de pacotes melhorado. O tema pare ser utilizado é o *spacemacs*.

Funções úteis

Ou, pelo menos, que o usuário considere úteis para o seu trabalho.

Rotacionar janelas

```
(defun rotate-windows ()
  "Rotate your windows"
  (interactive)
  (other-window -1)
  (cond ((not (> (count-windows)1))
         (message "You can't rotate a single window!"))
        (t
         (setq i 1)
         (setq numWindows (count-windows))
         (while  (< i numWindows)
           (let* (
                  (w1 (elt (window-list) i))
                  (w2 (elt (window-list) (+ (% i numWindows) 1)))

                  (b1 (window-buffer w1))
                  (b2 (window-buffer w2))

                  (s1 (window-start w1))
                  (s2 (window-start w2))
                  )
             (set-window-buffer w1  b2)
             (set-window-buffer w2 b1)
             (set-window-start w1 s2)
             (set-window-start w2 s1)
             (setq i (1+ i)))))))
(global-set-key (kbd "<f6>") 'rotate-windows)
```

Rotaciona os frames no sentido anti-horário. Mantém o foco no frame onde o usuário está trabalhando.

Renomeia buffer e arquivo

```
(defun rename-current-buffer-file ()
  "Renames current buffer and file it is visiting."
  (interactive)
```

```
  (let ((name (buffer-name))
        (filename (buffer-file-name)))
    (if (not (and filename (file-exists-p filename)))
        (error "Buffer '%s' is not visiting a file!" name)
      (let ((new-name (read-file-name "New name: " filename)))
        (if (get-buffer new-name)
            (error "A buffer named '%s' already exists!" new-name)
          (rename-file filename new-name 1)
          (rename-buffer new-name)
          (set-visited-file-name new-name)
          (set-buffer-modified-p nil)
          (message "File '%s' successfully renamed to '%s'"
                   name (file-name-nondirectory new-name)))))))
(global-set-key (kbd "C-x r C-f") 'rename-current-buffer-file)
```

Altera o nome do buffer e do arquivo em disco. Como se o usuário gravasse o arquivo, renomeasse no disco e abrisse novamente.

insere linha em branco

```
(defun open-line-below ()
  (interactive)
  (end-of-line)
  (newline)
  (indent-for-tab-command))

(defun open-line-above ()
  (interactive)
  (beginning-of-line)
  (newline)
  (forward-line -1)
  (indent-for-tab-command))
(global-set-key (kbd "C-x C-<down>") 'open-line-below)
(global-set-key (kbd "C-x C-<up>") 'open-line-above)
```

Insere uma linha em branco acima ou abaixo da linha onde está o cursor. O cursor pode estar em qualquer posição na linha.

Conclusão

O presente trabalho encontra-se em fase de testes. Espero que seja útil para quem deseja iniciar com o Emacs ou para quem deseja incrementar e automatizar a sua instalação.

makedoc

A documentação de ./docs foi gerada pelo script **makedoc**. Antes da geração é necessário que o usuário gere um arquivo *.html* pelo org-mode *Ctrl+c Ctrl+e h h* (exporta em formato html para o disco). Utiliza o **pandoc** para gerar os formatos *docx epub odt*. Para a geração do pdf, primeiro é gerado um arquivo .tex, depois o *sed* efetual algumas alterações que achei interessantes e, finalmente, gera o .pdf pelo .tex via *pdflatex*. Roda no Linux. Não sei os equivalentes para Windows.