Guia Rápido para o uso do `org-mode`

Marco Dimas Gubitoso

March 25, 2020

Contents

1 Introdução

Este é um guia rápido para começara usar p `org-mode` do *Emacs*. Não é completo, nem daria para ser, pois há um número enorme de opções e

modos de configuração. No entanto, deve bastar para a maior partes das tarefas do dia a dia.

Vou me focar na produção de *slides* e de páginas *web*. Neste tutorial, o objetivo são páginas e documentos LaTeX. Farei um outro tutorial específico para *slides* usando *Beamer*.

Uma vez domindo o básico, será fácil estender para outras atividades. Entre os recursos adicionais estão:

- Lista de tarefas (*Todo lists*, *checklists* etc)
- Agenda
- Planilhas (é bom conhecer *lisp*)
- Geração de *sites*
- Anotações de ideias para planejamento

Talvez a melhor forma de começar a aprender seja olhando o fonte deste prórpio documento.

2 Organização do documento

As seções são definidas por uma série de *s no começo da linha. O número de *s indica o nível, da forma natural:

`*` Seção

`**` Subseção

`***` Subsubseção

e assim por diante.

O início do documento pode conter um cabeçalho de configurações. Eu pessoalmente gosto de colocar esta linha

```
#+STARTUP: hidestars
```

para ocultar asteriscos repetidos.

3 Opções gerais de configuração

3.1 Por documento

Cada documento pode conter suas próprias configurações, que aparecem como comentários especiais.

Os comentários seguem o formato das linguagens de *script*, isto é, tudo que segue um `#` é ignorado. Para as opções, o início da linha deve conter `#+` seguido do nome da configuração, seguido pr `:`, sem espaços. Depois disso, na mesma linha, seguem os valores. Veja o exemplo acima.

São muitas opções que podem ser vistas na documentação. As mais comuns eu citarei ao longo do documento. Farei uma outra página com um resumo geral.

3.2 Universais

Há configurações que afetam o comportamento do `org-mode` como um todo. Normalmente não é preciso mexer nelas, a menos que você queira adaptações muito particulares.

No entanto, existe pelo menos uma que vale a pena olhar: `org-babel-load-languages`. Esta variável é uma lista das linguagens que recebem formatação especial e que podem ser executadas dentro de um documento. As opções disponíveis dependem dos modos instalados, eis algumas:

Linguagem	Módulo necessário	Comentário
Awk	awk	
C	nenhum	
C++	nenhum	
Calc	nenhum	Calculadora do Emacs
CSS	nenhum	Estilos para HTML
Ditaa	ditaa	Desenhos simples (veja abaixo)
Dot (Graphviz)	dot	
Emacs Lisp	nenhum	
Fortran	fortran	
Gnuplot	gnuplot, gnuplot-mode	
Java	java	
Javascript	node.js	
LaTeX	latex, auctex, reftex	
Lisp	lisp, slime	
Make	nenhum	
Matlab	matlab, matlab.el	
Octave	octave	
Org	nenhum	
Perl	perl, cperl-mode (opcional)	
Python	python, python-mode (opcional)	
R	R, ess-mode, tikzDevice	
Ruby	ruby, irb, ruby-mode, inf-ruby mode	
Sass	sass, sass-mode	
Scala	scala	
Scheme	nenhum	
shell	algum *shell*	
SQLite	SQLite, sqlite3, SQL mode	Banco de dados simples

Há muitas outras. Pessoalmente eu habilitei

- *emacs-lisp*
- *fortran*
- *gnuplot*
- *latex*
- *perl*
- *python*

- *ruby*
- *sh*
- *sqlite*
- *ditaa*
- *C*

4 Escrevendo o documento

4.1 Linguagem de marcação

O texto usa uma linguagem de marcação bem simples e intuitiva.

`/itálico/`	*itálico*
`*negrito*`	**negrito**
`_sublinhado_`	sublinhado
`+cancelado+`	~~cancelado~~
`= verbatim =`	`verbatim`
`~código~`	`código`
`expoente^2`	expoente^2
`índice_ijk`	índice$_{ijk}$

Além disso, símbolos comuns do Latex podem ser usados diretamente como α, β, γ ou Γ — veja `org-entities-help`

```
Além disso, símbolos comuns do Latex podem ser usados diretamente
como \alpha, \beta, \gamma ou \Gamma  --- veja =org-entities-help=
```

Para textos mais complexos, *environments* do Latex podem ser usados normalmente, se necessários

$$x = \sqrt{x^2} \tag{1}$$

Os trechos em Latex podem ser visualizados no próprio Emacs com `C-c C-x C-l`. `C-c C-c` faz retornar ao normal.

4.2 Listas

São três tipos, seguindo o padrão do LaTeX.

1. Enumeradas ou ordenadas, como essa aqui.

   (a) Começam com um número seguido de `.` ou `)`

   (c) A numeração pode ser alterada se colocar um `[@` *número* `]` antes do **texto**.

   (d) Para letras, é preciso mudar a variável `org-list-allow-alphabetical`. Há algumas formas de fazer isso.

      i. Use `C-h v`. para mudar

      ii. Use um bloco de código (veja mais à frente).

      iii. Mude seu `.emacs`

2. Normais (não ordenadas), podem ser marcadas com `*`, `+` ou `-`.

   - O uso de `*` não é recomendado pois pode ser confundido com marcador de seção, embora não haja ambiguidade.
   - Fora isso, não há segredo.

3. Descritivas. São como as não ordenadas, mas cada item recebe um nome seguido de `::`.

   simples Basta usar o `::` como separador

   fácil é edição normal de texto. Em todas as listas `M-<enter>` inicia o próximo item.

   rápido não precisa de muita formatação manual.

   Este é o código:

```
1. Enumeradas ou ordenadas, como essa aqui.
   1. Começam com um número seguido de =.= ou =)=
   3. [@3] A numeração pode ser alterada se colocar um =[@= /número/
    =]= antes do *texto*.
   4. Para letras, é preciso mudar a variável
    =org-list-allow-alphabetical=. Há algumas formas de fazer isso.

      a) Use ~C-h v~. para mudar
```

```
        b) Use um bloco de código (veja mais à frente).
        c) Mude seu =.emacs=
2. Normais (não ordenadas), podem ser marcadas com =*=, =+= ou =-=.
     + O uso de =*= não é recomendado pois pode ser confundido com
     marcador de seção, embora não haja ambiguidade.
     + Fora isso, não há segredo.
3. Descritivas.  São como as não ordenadas, mas cada item recebe um
  nome seguido de =::=.
     + simples :: Basta usar o =::= como separador

     + fácil :: é edição normal de texto. Em todas as listas
              =M-<enter>= inicia o próximo item.

     + rápido :: não precisa de muita formatação manual.
```

4.3 Tabelas

Este é um dos pontos altos do `org-mode` e merece um tutorial à parte. Não pela dificuldade, mas pela enorme quantidade de recursos oferecidos. Aqui eu novamente vou mostar o básico e, se houver interesse, monto uma explicação mais profunda depois.

As tabelas são construídas simplesmente separando as colunas por `|`; A tecla `TAB` faz todo o alinhamento automaticamente. Para colocar uma linha horizontal, basta começá-la com `|---` e pressionar `TAB`. Há exemplos acima, vou destacar um deles:

```
|----------------+--------------|
| =/itálico/=    | /itálico/    |
| =*negrito*=    | *negrito*    |
| =_sublinhado_= | _sublinhado_ |
| =+cancelado+=  | +cancelado+  |
| = =verbatim= = | =verbatim=   |
| =~código~=     | ~código~     |
| =expoente^2=   | expoente^2   |
| =índice_ijk=   | índice_ijk   |
|----------------+--------------|
```

Para inserir ou remover uma coluna, pode-se usar `M-S-<right>` ou `M-S-<left>`, respectivamente.

4.3.1 Conversão automática

A função `org-table-convert-region` transforma uma região de texto em uma tabela. Para definir as colunas, a função usa TABs, vírgulas ou sequências de espaços, dependendo da estrutura das linhas. Um prefixo 4 (`C-u`) força vírgulas, um prefixo 16 (`C-u C-u`) força TAB e outro prefixo numérico força uma quantidade fixa de espaços.

Esta possibilidade é muito útil para imprtar arquivos CSV facilmente.

4.4 Imagens

As imagens podem ser inseridas diretamente, colocando o nome do arquivo entre colchetes duplos:

```
[[file:algo.jpg]] ou [[./algo.jpg]]
```

É possível ainda colocar uma legenda com um comentário *antes* da imagem (ou tabela):

```
#+CAPTION: Legenda da imagem ou tabela
```

Para dar um nome de referência, para *links*, por exemplo, usa-se o mesmo método:

```
#+NAME: fig:exemplo
```

4.5 *Links*

Funcionam do mesmo modo que as imagens, entre colchetes duplos, com uma URL (ou URI) no meio. No caso dos links, é possível colocar uma descrição. Por exemplo, para voltar para o começo

```
Por exemplo, [[começo][para voltar para o começo]]
```

Neste exemplo, temos uma âncora local, definida assim:

```
<<começo>>
```

bem no início, veja o fonte.

Figure 1: Tudo em paz

4.5.1 Notas de rodapé

São casos particulares, marcadas entre colchetes simples e começando com `fn:`. Se estão no meio do texto, formam uma referência, se a marcação estiver no início da linha, é a nota [1] propriamente dita.

```
São casos particulares, marcadas entre colchetes simples e começando
com =fn:=.  Se estão no meio do texto, formam uma referência, se a
marcação estiver no início da linha, é a nota [fn:simples]
propriamente dita.
```

[1] Fácil, não é mesmo?

No final do documento coloquei

```
[fn:simples] Fácil, não é mesmo
```

5 Blocos

Embora tecnicamente seja parte do documento, resolvi deixar em uma seção à parte, pois é outro ponto bem versátil do `org-mode`.

Os blocos são delimitados com duas diretivas especiais:

```
#+BEGIN_...
...
#END
```

A sequência `...` deve ser trocada pelo tipo de bloco desejado. Existem vários, em particular os das linguagens especificadas em `org-babel-load-languages` e os descritos abaixo.

Na versão que uso, os blocos podem ser rapidamente inseridos com `<` seguido de uma letra e um TAB no início da linha, a letra sendo a primeira de cada tipo (`<c` insere `CENTER`, `<s` insere `SRC` etc).

5.1 Gerais

- `CENTER`

  Texto centralizado, *com marcações*. **Bem** conveniente. $\alpha^2 = \beta+\zeta^3$

```
#+BEGIN_CENTER
Texto centralizado, /com marcações/.
*Bem* conveniente.  \alpha^2 = \beta+\zeta^3
#+END_CENTER
```

- `EXAMPLE` Também pode ser gerado iniciando a linha com `:`.

```
Para exemplos, isto é, verbatim.
O está dentro do bloco é copiado literalmente para a saída.
```

```
#+BEGIN_EXAMPLE
Para exemplos, isto é, verbatim.
O está dentro do bloco é copiado literalmente para a saída.
#+END_EXAMPLE
```

- `QUOTE` Para citações.

  I love quotations because it is a joy to find thoughts one might have, beautifull expressed with much authority by someone recognized wiser than oneself.

  ~Marlene Dietrich

```
#+BEGIN_QUOTE
I love quotations because it is a joy to find thoughts one might have,
beautifull  expressed with much authority by someone recognized wiser
than oneself.

~Marlene Dietrich
#+END_QUOTE
```

- `VERSE` Para versos

  Por vezes à noite há um rosto
  Que nos olha do fundo de um espelho
  E a arte deve ser como esse espelho
  Que nos mostra o nosso próprio rosto.

  Jorge Luis Borges

```
#+BEGIN_VERSE
Por vezes à noite há um rosto
Que nos olha do fundo de um espelho
E a arte deve ser como esse espelho
Que nos mostra o nosso próprio rosto.

Jorge Luis Borges
#+END_VERSE
```

5.2 Saídas específicas

São blocos que são usados apenas em formatos específicos e ignorados nos outros. Os formatos possíveis são vários, incluindo *LibreOffice* (odt). Os blocos pré-definidos são estes

- `ASCII` Para saída em texto normal

```
#+BEGIN_ASCII
Este conteúdo  só aparecerá se for selecionado o formato ASCII
#+END_ASCII
```

- HTML

```
#+BEGIN_HTML
Este conteúdo  só aparecerá se for selecionado o formato <i>HTML</i>.
#+END_HTML
```

- LATEX

Este conteúdo só aparecerá se for selecionado o formato LaTeX.

```
#+BEGIN_LaTeX
Este conteúdo  só aparecerá se for selecionado o formato \LaTeX.
#+END_LaTeX
```

5.3 Código fonte

Os blocos com `SRC` recebem um parâmetro adicional com a linguagem desejada. Os blocos são formatados de acordo com a sintaxe, além de permitir que os programas sejam executados.

Tanto o código fonte como o resultado podem ser incluídos no documento final, para isso coloque `:exports` na linha inicial, depois da linguagem. `:exports` recebe um destes valores:

- `code` : apenas o código é exportado, esse é o *default*
- `results` : apenas os resultados
- `both` : código e resultado são exportados
- `none` : nenhum dos dois

Aqui colocarei apenas alguns exemplos, se houver interesse eu faço um outro tutorial específico, com mais informação e detalhes.

Dentro de um bloco, pode-se editar no modo específico da linguagem com `M-x org-edit-special`, normalmente associada a `C-c '`. Seguem alguns exemplos:

C Note o uso de C maiúsculo

```
#include <stdio.h>

int main()
{
  puts("Hello world!");
  return 0;
}
```

```
Hello world!
```

No fonte está escrito assim:

```
 #+BEGIN_SRC C :exports both
#include <stdio.h>

int main()
{
  puts("Hello world!");
  return 0;
}
 #+END_SRC
```

elisp útil para configurações locais no emacs

Neste exemplo, eu simplesmente autorizei a execução de blocos sem perguntas.

```
(setq org-confirm-babel-evaluate nil)
```

ruby

```
require 'date'

$av = "avaliado em #{Date.today}"
```

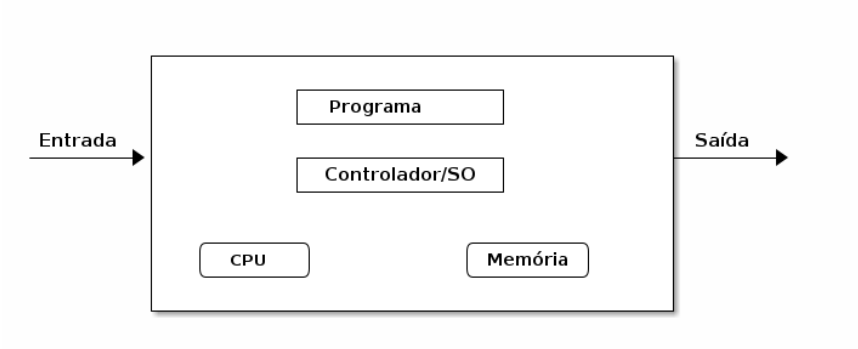
```
avaliado em 2020-03-25
```

python

```
return 6*7
```

```
42
```

ditaa Para desenhos simples. Precisa ter `ditaa.jar` instalado e atualizar a variável `org-ditaa-jar-path` com sua atualização. O pacote `ditaa` está incluído no Ubuntu.



Este é o pedaço do documento que gera esta imagem:

```
#+BEGIN_SRC elisp :exports none
(setq org-ditaa-jar-path "/usr/share/ditaa/ditaa.jar")
#+END_SRC

#+RESULTS:
: /usr/share/ditaa/ditaa.jar


#+BEGIN_SRC ditaa :file eniac.png  :exports results

            +-----------------------------------------+
            |                                         |
            |           +----------------+            |
            |           |  Programa      |            |
            |           +----------------+            |
   Entrada  |                                         | Saída
  --------->|           +----------------+            +-------->
            |           | Controlador/SO |            |
            |           +----------------+            |
            |                                         |
            |                                         |
            |   /--------\            /---------\     |
            |   |  CPU   |            | Memória |     |
            |   \--------/            \---------/     |
            |                                         |
            +-----------------------------------------+

#+END_SRC

#+RESULTS:

:  file:eniac.png
```

6 Geração das saídas

A geração das saídas é muito simples. Basta usar `C-c e` e um menu aparecerá com as opções de geração. O menu é auto-explicatório, mas coloco aqui as opções mais comuns:

`C-c e l p` Gera LaTeX e o PDF correspondente.

`C-c h o` Gera HTML e abre o *browser* padrão.

`C-c t u` Grava um arquivo texto formatado e em UTF-8

O processamento é muito rápido, a possibilidade de abrir imediatamente o documento final ajuda nos ajustes.

Para o LaTeX formatar corretamente código fonte é preciso incluir um `package` adequado. O mais direto é o `listings`. A linha abaixo pode ser colocada no início do documento e será incluída no cabeçalho do arquivo `.tex`, outras inclusões podem ser feitas do mesmo modo:

```
#+LATEX_HEADER: \usepackage{listings}
```

O bloco com `elisp` gera um erro inofensivo, pois `listings` não conhece a linguagem. Há outras opções, como `minted`, mas é preciso configurar mais coisas e ficará para um eventual tutorial no futuro.

Ainda para saídas específicas, pode-se incluir linhas como estas:

```
#+latex: \newline
#+html: <br>
```

Isso é tudo por enquanto. A ideia foi mostrar o suficiente para produzir páginas e PDFs. Se houver interesse eu mostro como preparar outros tipos de material.