Org-mode 简明手册

Table of Contents

1 简介 [1]

1.1 序

Org 是一个用文本方式来快速高效地做笔记、维持待办事项和做项目计划的模式。它是一个创作发布系统。

这个文档是 Org-mode 手册的一个压缩版本。包含了所有的基本功能和命令，以及一些重要的定制提示。本文档写给不想阅读 200 多页手册的新手。

1.2 安装

重要： *如果你用的是 Emacs 或者 XEmacs 包里的 Org 版本，请跳过本节，直接阅读 1.3 节。*

如果你是从网站上下载的 Org 版本，无论是 zip 包还是 tar 文件或者是 Git 文件，最好在分发包目录里直接来设置它。把 lisp 子目录加到 Emacs 的加载路径里，可以把下面两句加在 “.emacs” 文件里：

```
(setq load-path (cons "~/path/to/orgdir/lisp" load-path))
(setq load-path (cons "~/path/to/orgdir/contrib/lisp" load-path))
```

为了提高速度可以用下面的 shell 命令将 Lisp 文件编译一下：

```
make
```

再把下面一行加到 .emacs 文件里。它可以使文件中的函数自动加载，而不是启动 Org 模式时立即加载。

```
(require 'org-install)
```

1.3 激活

把下面几行加到 .emacs 文件里。后三行是为命令定义全局快捷键——请改成适合你自己的。

```
;; The following lines are always needed. Choose your own keys.
(add-to-list 'auto-mode-alist '("\\.org\\'" . org-mode))
(add-hook 'org-mode-hook 'turn-on-font-lock) ; not needed when global-font-lock-
mode is on
(global-set-key "\C-cl" 'org-store-link)
(global-set-key "\C-ca" 'org-agenda)
(global-set-key "\C-cb" 'org-iswitchb)
```

设置之后，打开 .org 扩展的文件会自动进入 org 模式。

1.4 反馈

如果你发现了问题，或者有问题评论或新想法，可以给 Org 的邮件列表 emacs-orgmode@gnu.org 发邮件。了解更多信息或者提交 bug，参见手册。

2 文档结构

Org 是基于 Outline 模式的，它提供了更灵活的编辑结构文件的命令。

2.1 大纲

Org 是在大纲模式之上实现的。大纲模式可以让我们用层次结构来组织文档，这（至少对我来说）是笔记和想法的最好实现方式。这种结构可以折叠（隐藏）文档的一部分而只显示文档的大概结构或者只显示我们正在处理的部分。Org 大大简化了大纲模式的使用，它把大纲模式的整个显示/隐藏功能整合到了一个命令中：org-cycle，这个命令绑定到了 TAB 键上。

2.2 标题

标题定义了大纲树的结构。它以处于一行左边缘的一个或多个星号开头。例如：

```
* Top level headline
** Second level
*** 3rd level
    some text
*** 3rd level
    more text

* Another top level headline
```

如果你不喜欢太多的星号，可以以空格后加一个星号作为标题的开头。查看 15.2 节 [Clean view]，那里有设置方法。

2.3 视图循环

大纲模式可以隐藏缓冲区里的部分正文。Org 用绑定到 TAB 和 S-TAB 上的两个全命令来改变视图。

TAB	子树循环：当加上一个前缀参数时（C-u TAB），在下面的状态中改变当前子树的视图 FOLDED -> CHILDREN -> SUBTREE 当加上 shift 键时会触发全局的视图循环。
S-TAB 和 C-u TAB	全局循环：使整个缓冲区在下列状态中循环 OVERVIEW -> CONTENTS -> SHOWALL
C-u C-u C-u TAB	显示全部，包括 drawers。

当 Emacs 刚打开文件时，全局的状态是 OVERVIEW，也即只有顶层的标题可见。这可以通过变量 org-startup-folded 来设置。也可以通过 startup 关键字设置只对单个文件有效：

```
#+STARTUP: content
```

2.4 移动

下面的命令可以跳转到缓冲区其他的标题。

C-c C-n	下个标题
C-c C-p	上个标题
C-c C-f	下个同级的标题
C-c C-b	上个同级的标题
C-c C-u	回到上层标题

2.5 结构编辑

M-RET	插入一个同级标题。如果光标在文本列表中，创建一个新的项（见 2.7 节 [Plain lists]）。如果处于一行的中间，这一行会被分开，后面的一部分成为新的标题。
M-S-RET	插入一个和当前标题同级的 TODO 项
TAB（新的空的条目中）	如果新的条目中还没有文字，TAB 会调整到合适的级别。
M-LEFT/RIGHT	将当前的标题提升/降低一个等级。
M-S-LEFT/RIGHT	将当前子提升/降低一个等级。
M-S-UP/DOWN	将子树上/下移（和前/后个子树交换）。
C-c C-w	将条目或区域传送到另一个文件中。见 9.2 节 [Refiling notes]。
C-x n s/w	将缓冲区视图局限到当前子树中/再次放宽视图

如果有活动区域（暂时标记状态），提升和降低功能将会对区域中的所有标题起作用。

2.6 稀疏树

Org 模式的一个重要的功能是能根据大纲树中选择的信息构造出稀疏的树，这样文档就可以尽可能地折叠，但是选择的信息和它对应的标题会显示出来。试下就知道它是怎样工作的了。

Org 模式有几个命令可以创建这种树，这些命令都可以通过调度器来使用：

C-c /	它会提示再输入一个字符来选择稀疏树的创建命令。
C-c / r	触发后，会提示输入匹配串，并且将所有匹配的项显示成稀疏树。所有的匹配项都会高亮显示；按 C-c C-c 取消高亮。

其他的基于 TODO 关键字、标签或属性来选择标题的稀疏树命令，我们会在本手册的后面讨论。

2.7 文本列表

在大纲树的一项中，自定义格式的列表可以提供更多的组织结构，也使我们可以得到一个复先框列表（见 5.6 节［复先框］）。Org 模式可以处理这种列表，并且 HTML 导出器（见 12 章）也支持这种格式。

Org 能够识别有序列表、无序列表和描述列表。

- 无序列表项以‘-’、‘+’或者‘*‘开头。
- 有序列表项以‘1.’、‘1)’或者开头。
- 描述列表用‘::’将项和描述分开。

同一列表中的项的第一行必须缩进相同程度。当下一行的缩进与列表项的的开头的符号或者数字相同或者更小时，这一项就结束了。当所有的项都关上时，或者后面有两个空行时，列表就结束了。例如：

```
** Lord of the Rings
   My favorite scenes are (in this order)
   1. The attack of the Rohirrim
   2. Eowyn's fight with the witch king
       + this was already my favorite scene in the book
       + I really like Miranda Otto.
   Important actors in this film are:
   - Elijah Wood :: He plays Frodo
   - Sean Austin :: He plays Sam, Frodo's friend.
```

当光标位于一项的第一行时（带有项标志符号的行），下面的命令将会作用于该项：

TAB	折叠项
M-RET	在当前级别插入一个项，有前缀时是强制新建一个标题
M-S-RET	插入一个带有复先框的项（见 2.5 节［复先框］）
M-S-UP/DOWN	将当前项和它的子项向上/下移动（和相同的缩进的前/后一个项交换位置）。如果列表是有序的，数字会自动改变
M-LEFT/M-RIGHT	提升/降低项的缩进，不包含子项
M-S-LEFT/RIGHT	提升/降低项的缩进，包含子项
C-c C-c	如果项中有复先框，就触发改变其状态。并且自动保持本项的符号与缩进在列表中的一致性
C-c -	循环改变将当前列表的项标志符号

2.8 脚注

脚注就是以脚注定义符号开头的一段话，脚注定义符号是将脚注名称放在一个方括号里形成的，要求放在第 0 列，不能有缩进。而引用就是在正文中将脚注名称用方括号括起来。例如：

```
The Org homepage[fn:1] now looks a lot better than it used to.
...
[fn:1] The link is: http://orgmode.org
```

用下面的命令来处理脚注：

C-c C-x f	这是一个移动命令。当光标处理引用处时，跳转到它的定义；当光标处理定义处时，跳转到第一个引用处。其他情况下，新建一个脚注。当有前缀时，会提供一个菜单供选择操作，其中包括重新给脚注编号。
C-c C-c	在定义和引用之间跳转

扩展阅读：

手册第 2 章

Sacha Chua's tutorial

3 表格

Org 提供了快速易用的表格编辑功能。通过调用 Emacs 内嵌的 ‘calc’包（对于 Emacs 的计算器可以查看 Emacs Calculator 手册）它支持类似于制表软件的计算操作。

Org 能够很容易地处理 ASCII 文本表格。任何以‘|’为首个非空字符的行都会被认为是表格的一部分。’ |‘也是列分隔符。一个表格是下面的样子：

```
| Name  | Phone | Age |
|-------+------+-----|
| Peter | 1234 | 17  |
| Anna  | 4321 | 25  |
```

当你在表格内部输入 TAB、RET 或者 C-c C-c 时表格都会自动调整。TAB 会进入下一个区域（RET 进入下一行）并且创建一个新的行。表格的缩进程度可以在第一行设定。以’ |-’开头的一行会作为一个水平分隔行，当它下次调整排列时会将 ‘-’扩展至填充整行。所以想要建上面的那个表格，只需键入：

```
|Name|Phone|Age|
|-
```

然后 TAB 排列表格。还有一个更快的方法就是键入|Name|Phone|Age，再 C-c RET。

在表格区域中输入文本时，DEL、BACKSPACE 和所有其他的字符会以特殊的方式处理，防止影响到其他的区域。当按 TAB、S-TAB 或者 RET 将光标移动到其他区域时，区域中会自动填充一些空格。

创建和转换

C-c | 将活动区域（选中区域）转换成一个表。如果第一行至少有一个 TAB 字符，就用 TAB 划分内容；如果第一行都有逗号，就分逗号划分内容；否则就用空白符来划分区域。如果当前没有活动区域就会建立一个空的 Org 表格。其实用|Name|Phone|Age C-c RET 来建表会更简单一点。

调整和区域移动

C-c C-c	调整表格，不移动光标
TAB	调整表格，将光标移到下一个区域，必要时新建一行
S-TAB	调整表格，将光标移到上一个区域
RET	调整表格，将光标移到下一行，必要时会新建一行

编辑行和列

M-LEFT/RIGHT	左/右移当前列
M-S-LEFT	删除当前行
M-S-RIGHT	在光标位置左边添加一列
M-UP/DOWN	上/下移当前行
M-S-UP	删除当前行
M-S-DOWN	在当前行上面添加一行。如果有前缀，则在下面添加一行
C-c -	在当前行下面添加一个水平线。如果带前缀，则在上面添加一行水平线
C-c RET	在当前行下面添加一个水平线。并将光标移动到下一行
C-c ^	将表排序。当前位置所在的列作为排序的依据。排序在距当前位置最近的两个水平线之间的行（或者整个表）中进行

扩展阅读：

手册第 3 章
Bastien’s table tutorial
Bastien’s spreadsheet tutorial
Eric’s plotting tutoria

4 超链接

就像 HTML 一样，Org 也提供了文件的内部链接，以及到其他文件、新闻组、电子邮件的外部链接等链接格式。

4.1 链接格式

Org 能够识别 URL 格式的文本并将它们处理成可点击的链接。通常链接格式是这样的：

`[[link][description]]` 或者 `[[link]]`

链接输入一旦完成（所有的括号都匹配），Org 就会改变它的视图。这里会看到 ”description“ 和 ”link“，而不是

`[[link][descriptoin]]`

和

`[[link]]`。

要想编辑链接，可以光标置于链接上并键入 C-c C-l。

4.2 内部链接

如果一个链接地址并不是 URL 的形式，就会作为当前文件内部链接来处理。最重要的一个例子是

```
[[#my-custom-id]]
```

它会链接到 CUSTOM_ID 属性是 “my-custom-id” 的项。

类似

```
[[My Target]]
```

和

```
[[My Target][Find my target]]
```

的链接，点击后本文件中查找对应的目标 “<<My Target>>”。

4.3 外部链接

Org 支持的链接格式包括文件、网页、新闻组、BBDB 数据库项、IRC 会话和记录。外部链接是 URL 格式的定位器。以识别符开头，后面跟着一个冒号，冒号后面不能有空格。下面是一些例子：

```
http://www.astro.uva.nl/~dominik             on the web
file:/home/dominik/images/jupiter.jpg        file, absolute path
/home/dominik/images/jupiter.jpg             same as above
file:papers/last.pdf                         file, relative path
file:projects.org                            another Org file
docview:papers/last.pdf::NNN                 open file in doc-view mode at page
NNN
id:B7423F4D-2E8A-471B-8810-C40F074717E9      Link to heading by ID
news:comp.emacs                              Usenet link
mailto:adent@galaxy.net                      Mail link
vm:folder                                    VM folder link
vm:folder#id                                 VM message link
wl:folder#id                                 WANDERLUST message link
mhe:folder#id                                MH-E message link
rmail:folder#id                              RMAIL message link
gnus:group#id                                Gnus article link
bbdb:R.*Stallman                             BBDB link (with regexp)
irc:/irc.com/#emacs/bob                      IRC link
info:org:External%20links                    Info node link (with encoded space)
```

链接的括号应当是闭合的。当链接含有描述文字是显示描述文字而不是链接地址（见 4.1 节［链接格式］），例如：

```
[[http:www.gnu.org/software/emacs/][GNU Emacs]]
```

如果描述信息是一个文件名或者是指向图片的 URL。HTML 导出（见 12.4 节[HTML 导出]）时会将图片内联成一个可以点击的按钮。如果没有描述信息且链接指向一个图片，那么图片就会嵌入到导出的 HTML 文件中。

4.4 使用链接

Org 提供了以下方法来创建和使用链接。

C-c l	在当前位置保存一个链接。这是一个全局命令（你可以设置自己的快捷键），可以在任何类型的缓冲区中使用。链接保存下来以便以后插入 Org 文件中（见

	下面）
C-c C-l	插入一个链接。它会让你输入，你可以输入一个链接，也可心用上/下键来获取保存的链接。它还会让你输入描述信息。
C-c C-l（光标在链接上）	当光标处于链接上时，你可以修改链接
C-c C-o 或者 mouse-1 或者 mouse-2	打开链接
C-c &	跳回到一个已记录的地址。用 C-c % 可以将地址记录下来，内部链接后面的命令也会自动将地址记录下来。使用这个命令多次可以一直往前定位。

4.5 目标链接

文件链接可以包含一些其他信息使得进入链接时可以到达特定的位置。比如双冒号之后的一个行号或者搜索选项。

下面是一些包含搜索定位功能的链接例子以及其说明：

```
file:~/code/main.c::255            进入到 255 行
file:~/xx.org::My Target           找到目标‘<<My Target>>’
file:~/xx.org/::#my-custom-id      查找自定义 id 的项
```

扩展阅读：

手册第四章

5 待办事项

Org 模式并不用一个单独的文件来维持 TODO 列表[2]。它是一些笔记的集合体，因为 TODO 列表是在你记录笔记的过程中逐渐形成的。你 Org 模式下可以很容易地将树中的一项标记为一个 TODO 的项。用这种方式，信息内容不会冗余加倍，而且可以显示 TODO 项的上下文环境。

当然，这种处理待办事项的方式会将它们分散于各个笔记文件中。Org 模式提供了一些方法使我们可以把它们看作一个整体来处理。

5.1 使用 TODO 状态

当标题以 TODO 开关时它就成为了一个 TODO 项，例如：

```
***TODO Write letter to Sam Fortune
```

下面是一些使用 TODO 项的常用命令：

C-c C-t	将当前项的状态在（unmarked）->TODO->DONE 之间循环切换，同样的切换也可以在时间轴（timeline） 和议程（ agenda） 的缓冲区（buffer）中用 t 键“远程”进行。（见 2.6 节[稀疏树]）
S-	选择下一个/上一个 TODO 状态，与上面的循环方式相同。

RIGHT/LEFT	
C-c / t	在稀疏树中显示 TODO 项。将 buffer 折叠，但是会显示 TODO 项和它们所在的层次的标题。
C-c a t	显示全局 TODO 列表。从所有的议程文件中收集 TODO 项到一个缓冲区中。详见 10.3.2 节。
S-M-RET	在当前项下插入一个新的 TODO 项。

改变 TODO 的状态会触发标签改变。查看选项 org-todo-state-tags-triggers 的描述获得更多信息。

5.2 多状态工作流程

你可以用 TODO 关键字来定义不同的状态，用以处理项，比如：

```
(setq org-todo-keywords
      '((sequence "TODO" "FEEDBACK" "VERIFY" "|" "DONE" "DELEGATED")))
```

竖直线将 TODO 关键字（还需要进一步的动作）和 DONE 状态（不需要进一步的动作）分隔开。如果你不给出竖直线，最后一个状态会作为 DONE 状态。设置之后，C-c C-t 就会将状态从 TODO 转换到 FEEDBACK，再转换到 VERIFY，最后到 DONE 和 DELEGATED。

有时你可能希望同时使用几个不同的 TODO 状态集合。例如，你可能想要一个基本的 TODO/DONE，以及一个修改 bug 的工作流程和一个隔开的状态来表示取消的项目（既还是 DONE，也不需要进一步的动作），你可以这样设置：

```
(setq org-todo-keywords
      '((sequence "TODO(t)" "|" "DONE(d)")
        (sequence "REPORT(r)" "BUG(b)" "KNOWNCAUSE(k)" "|" "FIXED(f)")
        (sequence "|" "CANCELED(c)")))
```

关键字应该各不相同，这样对于一个选项 Org 才知道该用哪个状态序列（集合）。例子中也给出了快速使用一个关键字的方法，就是在关键字后面括号中给出快捷字母——当用 C-c C-t 时，会询问，让你输入一个字母。

要定义只在一个文件中有效的 TODO 关键字，可以在文件中任意地方给出下面的文本：

```
#+TODO: TODO(t) | DONE(d)
#+TODO: REPORT(r) BUG(b) KNOWNCAUSE(k) | FIXED(f)
#+TODO: | CANCELED(c)
```

当改变这些行中的一行后，光标停留在改变行上，用 C-c C-c 让改变生效。

5.3 进度日志

当你改变一个 TODO 状态为 DONE 时，或者当你每次改变一个 TODO 项的状态时，Org 都会自动记录时间戳或者作一个记录。这是高度可配置的。可以基于每一个关键字进入设置，并且可以定位到一个文件甚至子树。怎样记录一个任务的工作时间，见 8.4 节。

完成的项目

最基本的日志功能是跟踪一个特定项目的完成。这可以这样实现：[3]

```
(setq org-log-done 'time)
```

这时当你将一个项目从一个 TODO（未完成）状态改变为一个完成状态时，标题下面就会插入一行“CLOSED:[timestamp]”。如果你想和时间戳一起作一个记录，用：[4]

```
(setq org-log-done 'note)
```

这时会提示你输入一个记录（note），并将它保存在标题为“Closing Note”项目之下。

跟踪 TODO 状态变化

你可能想跟踪 TODO 状态的变化。可以只记录一个时间戳，也可以为变化作一个带时间戳的记录。记录会被插入到标题之后形成列表。当有很多记录之后，你可能希望将记录取出放到抽屉里。通过定制变量 org-log-into-drawer 可以实现这个功能。 对于状态记录，Org 可以实现基于每个状态关键字的设置。实现方法是在每个后的括号中指定“！”（记录时间戳）或“@”（作一个记录）。例如：

```
#+TODO: TODO(t) WAIT(w@/!) | DONE(d!) CANCELED(c@)
```

将会设置 TODO 关键字和快速访问字母，以及当一个项目设为 DONE 时，会记录时间戳，当状态变为 WAIT 或 CANCELED 时，会作一个记录。这个语法也适用于变量 org-todo-keywords。

5.4 优先级

如果你广泛地使用 Org 模式，这样你就会有大量的 TODO 项。给它们设定优先级就很有必要。可以在 TODO 项的标题中加入一些标记（cookie）来设置它们的优先级，像这样：

```
*** TODO [#A] Write letter to Sam Fortune
```

Org 模式支持三个优先级别：’ A‘、’ B‘和’ C‘。’ A‘是最高级别，如不指定，’ B‘是默认的。优先级只在议程中有用。

C-c ,	设置当前标题的优先级。按’ “ “ ‘选择一个级别，或者 SPC 删除标记（cookie）。
S-UP	
S-Down	增加/减少当前标题的优先级。

5.5 任务细分

很多时候将一个大的任务分成几个的易于完成的小任务是明智的。你可以通过在 TODO 项目下新建一个大纲树，并在子树上标记子任务来实现这个功能。为了能对已经完成的任务有个大致的了解，你可以在标题的任何地方插入‘[/]’或者‘[%]’。当每个子任务的状态变化时，或者当你在标记上按 C-c C-c 时，这些标记状态也会随之更新。例如：

```
* Organize Party [33%]
** TODO Call people [1/2]
*** TODO Peter
*** DONE Sarah
** TODO Buy food
** DONE Talk to neighbor
```

5.6 复选框

当纯文本中的项以‘[]’开头时，就会变成一个复选框。复选框不会包含在全局 TODO 列表中，所以它们很适合地将一个任务划分成几个简单的步骤。下面是一个复选框的例子：

```
* TODO Organize party [1/3]
  - [-] call people [1/2]
    - [ ] Peter
    - [X] Sarah
  - [X] order food
  - [ ] think about what music to play
```

复选框是分层工作的。所以如果一个复选框项目如果还有子复选框，触发子复选框将会使该复选框变化以反映出一个、多个还是没有子复选框被选中。

下面是处理复选框的命令：

C-c C-c	触发复选框的状态或者（加上前缀）触发复选框的的存在状态。
M-S-RET	增加一个带有复选框的项。这只在光标处于纯文本列表项（见 2.7 节）中才起使用。

扩展阅读：

手册第 5 章
David O’Toole’s introductory tutorial
Charles Cave’s GTD setup

6 标签

要为交叉相关的信息提供标签和上下文，一个不错的方法是给标题分配标签。Org 模式能够广泛地支持标签。

每一个标题都能包含多个标签，它们位于标题的后面。标签可以包含字母，数字，‘_’和‘@’。标签的前面和后面都应该有一个冒号，例如，“:work:”。可以指定多个标签，就像“:work:urgent:”。标签默认是粗体，并和标题具有相同的颜色。

6.1 标签继承

标签具有大纲树的继承结构。如果一个标题具有某个标签，它的所有子标题也会继承这个标签。例如，在列表

```
* Meeting with the French group      :work:
** Summary by Frank                  :boss:notes:
*** TODO Prepare slides for him      :action:
```

中,尽管没有明确标出,最后一个标题会有标签“:work:”，“:boss:”，“:note:”，和“:action”。你也可以设定一个标签让所有的标题都继承，就好像标签在包含整个文件的第零级标题中指定了一样。用下面的方法[5]:

```
#+FILETAGS: :Peter:Boss:Secret:
```

6.2 设置标签

在标题后可以很容易地输入标签。在冒号之后，M-TAB 可以补全标签。也有一些专门的命令用于输入标签：

C-c C-q	为当前标题输入标签。Org 模式既支持补全，也支持单键接口来设置标签，见下文。回车之后，标签会被插入，并放到第 org-tags-column 列。如果用前缀 C-u，会把当前缓冲区中的所有标签都对齐到那一列，这看起来很酷。
C-c C-c	当光标处于标题上时，这个命令同 C-c C-q。

Org 支持基于一个标签列表来插入标签。默认情况这个列表是动态构建的，包含了当前缓冲区中使用过的所有标签。你也可以通过变量 org-tag-alist 在全局设定一个标签的硬列表（hard list）。另外，对于某个特定文件你也可以用下面这几行设置一个默认列表：

```
#+TAGS: @work @home @tennisclub
#+TAGS: laptop car pc sailboat
```

默认 Org 模式用一个迷你缓冲区补全设施来输入标签。另外，它也实现了一个更快速，称为 *快速标签选择*（*fast tag selection*）的标签选择方法。这使得你只用按一次键就可以选择或者取消一个标签。为了使它能很好地工作，需要为常用的标签赋唯一的值。你可以在你的“.emacs”文件中通过设置变量 org-tag-alist 作全局设定。例如，如果你需要在不同的文件中经常要给条目添加标签“:@home:”，这时你就可以像这样设置：

```
(setq org-tag-alist '(("@work" . ?w) ("@home" . ?h) ("laptop" . ?l)))
```

如果标签只用于当前正在处理的文件，那么你可以这样设置标签选项行：

```
#+TAGS: @work(w) @home(h) @tennisclub(t) laptop(l) pc(p)
```

6.3 标签查找

一旦标签体系设置好，就可以用来收集相关联的信息到指定列表中。

C-c \	
C-c / m	用匹配标签搜索的所有标题构造一个稀疏树。带前缀参数 C-u 时，忽略所有还是 TODO 行的标题。
C-c a m	用所有议程文件匹配的标签构造一个全局列表。见第 10.3.3 节。
C-c a M	用所有议程文件匹配的标签构造一个全局列表，但只搜索 TODO 项，并强制搜索所有子项（见变量 org-tags-match-listsublevels）。

这些命令都会提示输入字符串，字符串支持基本的逻辑去处。像“+boss+urgent-project1”，是搜索所有的包含标签“boss”和“urgent”但不含“project1”的项；而 “Kathy|Sally”，搜索标签包含“Kathy”或者“Sally”和项。搜索字符串的语法很丰富，支持查找 TODO 关键字、条目级别和属性。更详细的介绍和例子，见第 10.3.3 节。

扩展阅读

手册第 6 章

Sacha Chua’s article about tagging in Org-mode

7 属性

属性是一些与条目关联的键值对。它们位于一个名为 **PROPERTIES** 的特殊抽屉中。第一个属性都单独一行，键在前（被冒号包围），值在后：

```
* CD collection
** Classic
*** Goldberg Variations
    :PROPERTIES:
    :Title:     Goldberg Variations
    :Composer:  J.S. Bach
    :Publisher: Deutsche Grammophon
    :NDisks:    1
    :END:
```

通过设置属性“`:Xyz_ALL:`”，你可以为属性“`:Xyz:`”设置所有合法的值。这个特定的属性是有 *继承性* 的，即，如果你是在第 1 级别设置的，那么会被应用于整个树。当合法的值设定之后，设置对应的属性就很容易了，并且不容易出现打字错误。用 CD 唱片集为例，我们可以预定义发行商和盒中的光盘数目：

```
* CD collection
  :PROPERTIES:
  :NDisks_ALL: 1 2 3 4
  :Publisher_ALL: "Deutsche Grammophon" Philips EMI
  :END:
```

也可以在全局设置 org-global-properties ，或者在文件级别设置：

```
#+PROPERTY: NDisks_ALL 1 2 3 4
```

C-c C-x p　设置一个属性。会询问属性名和属性值。

C-c C-c d　从当前项中删除一个属性。

要基于选择的属性创建稀疏树或者特殊列表，跟标签搜索的命令相同（见第 6.3 节）。搜索字符串的语法在第 10.3.3 节中详述。

7.1 扩展阅读

手册第 7 章

Bastien Guerry’s column view tutorial

8 日期和时间

为了支持工程的计划，TODO 项可以标记上日期和/或时间。带有日期和时间信息的特定格式的字符串在 Org 模式中称为时间戳。

8.1 时间戳

时间戳是一个具有特定格式的日期（可能带有时间和时间段）说明，例如 ~ 2005-10-01~ Tue ， ~ 2003-09-16~ Tue 09:39 ，或者 ~ 2003-09-16~ Tue 12:00-12:30 。 时间戳可以出现在树条目的标题和正文的任何地方。它能使条目只在特定的日期才出现在议程列表中。（见第 10.3.1 节）我们区分为：

普通时间戳；事件；约会

一个简单的时间戳只是给一个条目加上时间和日期。这跟在纸质的议程上写下约会和事件是一样的。

```
* Meet Peter at the movies <2006-11-01 Wed 19:15>
* Discussion on climate change <2006-11-02 Thu 20:00-22:00>
```

具有时间间隔的时间戳

一个时间戳可以包含一个时间间隔，表示事件不只在指定的时间发生，还在每隔一个特定的时间如 N 天（d）、周（w）、月（m）或者年（y）之后重复发生。下面的事件每周二在议程中显示：

```
* Pick up Sam at school <2007-05-16 Wed 12:30 +1w>
```

日记样式的 sexp 条目

为了能定义更复杂的时间，Org 模式支持 Emacs 日历/日记包（calendar/diary package）中的日记条目。例如：

```
* The nerd meeting on every 2nd Thursday of the month
  <%%(diary-float t 4 2)>
```

时间/日期段

两个时间戳用‘–’连接起来就定义了一个时间段：

```
** Meeting in Amsterdam
   <2004-08-23 Mon>--<2004-08-26 Thu>
```

非激活的时间戳

跟普通时间戳一样，但是这里是方括号而不是尖括号。这种时间戳是未激活的，它 *不* 会让一个条目显示在议程中。

```
* Gillian comes late for the fifth time [2006-11-01 Wed]
```

8.2 创建时间戳

时间戳要有特定的格式，这样才能被 Org 模式识别。下面的命令可以用来正确地处理时间戳的格式。

C-c .	询问日期并输入正确的时间戳。当光标处理一个时间戳之上时，是修改这个时间戳，而不是插入一个新的。如果这个命令连用再次，就会插入一个时间段。加上前缀会附带当前时间。
C-c !	功能同 C-c .，但是插入的是一个未激活的时间戳。

S-LEFT/RIGHT	将光标处理的时间戳改变一天。
S-UP/DOWN	改变时间戳中光标下的项。光标可以处在年、月、日、时或者分之上。当时间戳包含一个时间段时，如 “15:30-16:30”，修改第一个时间，会自动同时修改第二个时间，以保持时间段长度不变。想修改时间段长度，可以修改第二个时间。

当 Org 模式询问时间/日期时，能接收任何包含时间和/或日期的字符串，它能根据当前的时间日期智能地分析字符串，从而得到没有指明的信息。你也可以用弹出的日历中选择日期。想完整地了解时间/日期询问的工作方式，可以参考手册。

8.3 截止期限和计划安排

时间戳前面可以加一些关键字来协助计划安排。

截止期限

意义：任务（大多数情况都会是一个 TODO 项，当然也可以不是）应该完成的日期。

C-c C-d　在标题下面一行插入一个带有 “DEADLINE”关键字的时间戳。

在 *截止日期*， *任务* 会列在 *议程* 中。另外， *今天的议程* 会在任务到期 orgdeadline-warning-days 天前对即将即将到期以及已经过期的任务给出提醒，直到任务被标记为 DONE。例如：

```
*** TODO write article about the Earth for the Guide
    The editor in charge is bbdb:Ford Prefect
    DEADLINE: <2004-02-29 Sun>
```

日程安排

意义：你计划在给定的那个日期开始进行那项任务。[6]

C-c C-s　在标题下面插入一个带有 “SCHEDULED”关键字的时间戳。

在给定的日期标题会列在议程中。[7]另外，对于过期的日程安排会在编辑为 *今天* 并给出提醒，直到被标记为 DONE。也就是说，任务会自动推迟日期直到它被完成。

```
*** TODO Call Trillian for a date on New Years Eve.
    SCHEDULED: <2004-12-25 Sat>
```

有些任务需要一再重复出现。Org 模式在截止期限、计划安排和普通时间戳中用所谓的中继器来管理这种任务。在下面的例子中：

```
** TODO Pay the rent
   DEADLINE: <2005-10-01 Sat +1m>
```

+1m 是一个中继器；上面的意思是任务有一个截止期限 ~ 2005-10-01~ ，并从这个日期开始每月都重复出现。

8.4 记录工作时间

使用 Org 可以记录在一个工程中花在某些特定任务上的时间。

C-c C-x C-i	开始当前条目的计时（clock-in）。这会插入一个 CLOCK 关键字和一个时间戳。加上 C-u 前缀，从当前已经计时的任务中选择任务。
C-c C-x C-o	停止计时（clock-out）。这会在开始计时的地方插入另一个时间戳。它会直接计算使用时间并插入到时间段的后面如 “=> HH:MM”。
C-c C-x C-e	为当前的计时任务更新进度。
C-c C-x C-x	取消当前的计时。当你误操作打开一个计时时，或者转而去做其他事情时，这个命令就很有用。
C-c C-x C-j	跳转到包含当前正在运行的计时的任务条目。用 C-uf 前缀从当前计时的任务中选择。

C-c C-x C-r	在当前文件插入一个包含像 Org 表格一样的计时报告的动态块。当光标正处于一个存在的块上时，更新它。
	`#+BEGIN: clocktable :maxlevel 2 :emphasize nil :scope file`
	`#+END: clocktable`
	如何定制视图，见手册。

C-c C-c	在一个已经存在的计时表格之上时，更新它。 更新动态块。光标需要置于动态块 `#+BEGIN` 这行。

l 键可能会在时间轴（见第 10.3.4 节）和议程（见第 10.3.1 节）中使用来查看一天中处理和关闭了哪些任务。

扩展阅读

手册第 8 章
Charles Cave’s Date and Time tutorial
Bernt Hansen’s clocking workflow

9 捕获——转发——存档

任何组织系统都有一个重要功能，就是能捕获新的灵感或者任务，并将相关的引用材料与之联系起来。Org 提供了一个捕获过程来创建任务。它将与一个任务相关的文件（附件）保存在一个特定的目录下。在系统中，任务和项目经常移动。将整个项目树保存到一个归档文件中可以保持系统简洁快速。

9.1 捕获

Org 的获取一个新条目的方法很大程序上受 John Wiegley 的 excellent remem-ber package 的影响。它使得你可以在工作流程中中断一小会儿来存贮一个简短的笔记。Org 可以为新条目定义模板，并将它们与不同的目标文件关联起来以保存笔记。

设定截取位置 [8]

下面的定制为笔记设置了一个默认的目标 [9] 文件，并为捕获新的任务定义了一个全局快捷键 [10]。

```
(setq org-default-notes-file (concat org-directory "/notes.org"))
(define-key global-map "\C-cc" 'org-capture)
```

截取的使用

C-c c	启动一个捕获过程。进入一个窄的间接缓冲区来编辑条目。
C-c C-c	一旦完成捕获信息的输入，可以用 C-c C-c 返回之前的窗口，继续中断的工作。
C-c C-w	将条目保存到一个接收地址（见第 2 节）并结束。
C-c C-k	取消捕获过程，返回之前的状态。

捕获模板

用可以用不同的模板来做不同的捕获笔记，并将它们保存到不同的地方。例如，你想将新任务保存到文件“TODO.org”的“Tasks”标题下，而将日记项目保存到“journal.org”中一个时间树中。你可以：

```
(setq org-capture-templates
      '(("t" "Todo" entry (file+headline "~/org/gtd.org" "Tasks")
         "* TODO %?\n %i\n %a")
        ("j" "Journal" entry (file+datetree "~/org/journal.org")
         "* %?\nEntered on %U\n %i\n %a")))
```

其中，第一个字符串是模式的关键字，第二个字符串是简短的描述信息。接着是条目的类型和保存笔记的目标地址。最后是模板本身，它利用%作转义符基于时间和上下文来填充一些信息。

当你调用 M-x org-capture 时，Org 提示输入一个键来选择模板（如果你有多个模板），然后就会给出像这样的内容：

```
* TODO
 [[file:link to where you were when initiating capture]]
```

在扩展模板时，可以用%转义符进行动态地插入内容。下面是一些可以使用的项，查看手册获得更多的选项。[11]

%a	注解，通常是由 org-store-link 创建的链接
%i	初始化内容，当记忆时区域被 C-u 调用
%t	时间戳，只是日期
%T	带有日期和时间的时间戳
%u，%U	同上，但是时间戳不激活

9.2 转送笔记

当你回顾捕获的数据时，可以想把其中的一些条目转送到另一列表中，比如说到一工程项目。剪切，查找正确的地址，然后再粘贴笔记，这就似乎有些麻烦。为了简化这个过程，可以用专门的命令：

C-c C-w	转送光标处的条目或者区域。这个命令会提供一些目标地址供选择,你可以通过补全功能选择一个。条目（或者区域中的所有条目）就会作为一个子项填充到目标标题下。

	默认情况下，当前缓冲区的一级标题会被作为转送的目标，你可以通过设置给出跨多个文件的复杂的定义。详见变量 org-refile-targets 的描述。
C-u C-c C-w	借助于转送功能的接口来跳转到一个标题。
C-u C-u C-c C-w	跳转到 org-refile 最后转送子树所到的地方。

9.3 归档

当一个用（子）树表示的工程完成后，你可能想把它移走，不让它再在议程里显示。归档能使你的工作文件变得简洁，并能使议程视图构造等全局搜索保持高效。最常用的归档命令是将工程树移到另一个文件——归档文件。

C-c C-x C-a	用变量 orgarchive-default-command 指定的命令归档当前的项。
C-c C-x C-s 或者简化为 C-c $	将光标处的子树归档至 org-archive-location 指定的位置。

默认的归档位置是当前文件同目录下，名为当前文件名后加“_archive”的文件。例子和设置位置的方法见变量 org-archivelocation 的帮助信息。下面是一个在缓冲区内设置该变量的方法：

```
#+ARCHIVE: %s_done::
```

扩展阅读

手册第 9 章
Charles Cave’s remember tutorial
Sebastian Rose’s tutorial for capturing from a web browser

10 议程视图

根据 Org 的工作方式，TODO 项、时间戳和带标签的标题分散在一个或者多个文件中。为了能够查看某一天的项目或者事件，信息必须收集在一起，以一种的有条理方式排序、显示。有几种不同的视图，见下文。

收集的信息在一个专门的议程缓冲区中显示。这个缓冲区是只读的，但是提供了一些命令可以访问原 Org 文件中对应的条目，并且可以远程地编辑这些文件。从议程缓冲区中远程编辑是说，比如，你可以在议程缓冲区中改变标题和约会的日期。议程缓冲区中使用的命令在第 10.4 节列出。

10.1 议程文件

显示的信息通常是从各个议程文件中收集来的，这样文件在变量 org-agenda-files 中列出。

C-c [	将当前文件加入到议程文件列表中。当前文件会被加到列表的前面。如果文件已经在列表中，会被移到前面。带有前缀时，文件添加/移到到后面。
C-c]	将当前文件从议程文件列表中删除。
C-,	遍历议程文件列表，依次访问其中的每一个文件。

10.2 议程调度器

视图是通过议程调试器创建的，通常我们会给它设置一个全局快捷键——比如 C-c a （见第 1.2 节）。按 C-c a 之后，就会提示再输入一个字母来执行对应的命令：

a	日历式的议程。（见 10.3.1）
t/T	TODO 项的列表。（见 10.3.2 节）
m/M	匹配某个标签表达式的标题的列表。（见 10.3.3）
L	当前文件的时间轴视图。（见 10.3.4）
s	通过关键字和/或正则表达式选中的条目的列表。

10.3 内建议程视图

10.3.1 周/日议程

周/日议程就像纸质的议程一样，用以显示本周或当天的所有任务。

C-c a a	从一列 Org 文件中为本周收集出一个议程。议程显示出每天的条目。

Emacs 包含了 Edward M. Reingold 的日历和日记功能。Org 模式能识别日记的语法并允许在 Org 文件中直接使用日记的 sexp 条目：[12]

```
* Birthdays and similar stuff
#+CATEGORY: Holiday
  %%(org-calendar-holiday) ; special function for holiday names
#+CATEGORY: Ann
  %%(diary-anniversary 5 14 1956) Arthur Dent is %d years old
  %%(diary-anniversary 10 2 1869) Mahatma Gandhi would be %d years old
```

Org 可以跟 Emacs 的约会提醒功能结合。想添加议程文件中的约会提醒，可以使用命令 org-agenda-to-appt。详见帮助文档的描述。

10.3.2 全局 TODO 列表

全局 TODO 列表将所有未完成的 TODO 项格式化并集中到一处。TODO 项的远程编辑使得我们只用按一下键就可以改变 TODO 项的状态。TODO 列表中可以使用的命令在第 10.4 节给出。

C-c a t	显示全局 TODO 列表。这会从所有的议程文件（见第 10 章）中收集 TODO 项到一个缓冲区中。
C-c a T	同上，但可以选择 TODO 关键字

10.3.3 匹配标签和属性

如果议程文件中的标题带有标签（见第 6 章）或者带有属性（见第 7 章），就可以基于这些元数据筛选标题到议程缓冲区中。这里描述的匹配语法在用 C-c / m 创建稀疏树时也同样适用。在标签列表中可以使用的命令在第 10.4 节描述。

C-c a m　将匹配指定的标签集的所有标题生成一个列表。这个命令询问筛选规则，可以是标签的逻辑表达式，如 “+work+urgent-withboss” 或 “work|home” （见第 6 章）。如果你经常使用某个搜索，可以将它定义成一个命令。（见第 10.2 节）
C-c a M　同 C-c a m，但只复选同时也是 TODO 项的标题。

匹配语法

搜索字符串可以使用 ‘&’ 作与运算， ‘|’ 作或运算。 ‘&’ 的约束力比 ‘|’ 的强。括号功能现在还没实现。用以搜索的元素可以是标签 、匹配标签的正则表达式、或者像 PROPERTY OPERATOR VALUE 这样带有比较操作符的用来比较属性值的表达式。第一个元素前面加 ‘-’ 表示不选匹配的项，加 ‘+‘ 表示选择匹配的项。使用 ’ +‘ 和 ’ -“ 时，与操作符 ‘&’ 就是可选的了。这里有一些只使用标签的例子。

”+work-boss“	选择标有”:work:“的标题，但去掉同时也标有”:boss:“的标题。。
”work|laptop“	选择标有”:work:“或者”:laptop:“的行。
”work|laptop+night“	跟前面相同，但要求标有”:laptop:“和行也要标有”:night:“。

匹配标签时你也可以尝试同时匹配属性，详细内容见手册。

10.3.4 单文件时间轴

时间轴用时间排序视图概述单个文件中的所有带有时间戳的条目。这个命令的目的是用来给出一个工程中事件的鸟瞰图。

C-c a L	给出 Org 文件中所有带时间戳条目的排序视图。带有 C-u 前缀时，没有完成的 TODO 项（作了安排的以及没作安排的）也列在当前日期下。

10.3.5 查找视图

这个议程视图用来对 Org 模式下的条目进行普通的文本查找。对于查找笔记很有用。

C-c a s	这个查找方式可以让你通过匹配子串或者用逻辑表达式指定关键字来选择条目。

例如，查找字符串 ” **computer equipment“** 将会查找包含子串 ” **computer equipment“** 的条目。查找视图也可以用布尔逻辑查找条目中的关键字。查找字符串 ” **+computer +wifi -ethernet-{8\.11[bg]}“** 将会搜索包含关键字 **computer** 和 **wifi** 但不含 **ethernet** ，并且不被正则表达式 **8\.11[bg]** （排除 8.11b 和 8.11g） 匹配的笔记条目。

注意，除了议程文件，这条命令也会搜索 org-agenda-text-search-extra-files 中列出的文件。

10.4 议程缓冲区的命令

议程缓冲区中的条目链接到了它们的源 Org 或者日记文件。有一些命令可以用来显示和跳转到条目的源位置，也可以从视图缓冲区中” 远程 “编辑源文件。下面只是所有命令的一个选集，浏览 **Agenda** 菜单和手册获得完整的列表。

动作

n	下一行（同 DOWN 和 C-n）。[13]
p	上一行（同 UP 和 C-p）。[14]

查看/转到 Org 文件

mouse-3	
SPC	在另一个窗口中显示条目的源位置。带前缀使得整个条目在大纲中可见，而不只是标题。
TAB	在另一个窗口中条目的源位置。在 Emacs 22 之前的版本，mouse-1 也有这个功能。
RET	转到条目的源位置并删除其它的窗口。[15]

改变显示方式

o	删除其他的窗口。
d / w	切换到日/周视图。
f 和 b	时间前移或者后移来显示随后的 org-agenda-current-span 天。例如，如果显示了一周的内容，切换到下/上一周。
.	转到今天。
j	询问日期并转到那天。
v l 或简化为 l	触发日志模式（Logbook mode）。在日志模式中，当记录功能打开（变量 org-log-done）时标记为 DONE 的条目，以及在那天计时的条目，都会显示在议程中。
r 或 g	重新构造议程，以反映最新的状态。
s	保存当前 Emacs 会话的所有 Org 缓冲区和 ID 的地址。

二级筛选和查询编辑

/	根据标签过滤当前的缓冲区。提示你输入一个字母选择一个标签。先按 ‘-’排除一个标签。
\	通过增加条件缩小当前议程的视图。[16]

远程编辑（参考手册获得更多命令）

0-9	数字参数。
t	修改议程和 org 文件中的条目的 TODO 状态。
C-k	删除当前的议程条目以及源文件中它的整个子树。
C-c C-w	传送当前的条目。
C-c C-x C-a 或简作 a	用在 org-archive-default-command 中设置的默认归档命令对当前的条目对应的整个树进行归档。
C-c C-x C-s 或简作 $	归档当前标题对应的树。
C-c C-s	规划（Schedule）一个条目，带有前缀参数时删除规划时间戳。
C-c C-d	为条目设置截止期限，带前缀时删除截止期限。
S-RIGHT 和 S-LEFT	将与当前行相关的时间戳改变一天。

I	对当前条目开始计时。
O / X	暂停/取消最近开始的计时。
J	在另一个窗口中跳转到正在进行的计时。

10.5 定制议程视图

自定义搜索的主要用途是对于频繁使用的搜索进行快捷键绑定，从而快捷地创建议程缓冲区或者稀疏树（当然后者只涵盖当前缓冲区的内容）。自定义的命令是用变量 org-agenda-custom-commands 来配置的。你可以用 C-c a C 来定制这个变量。也可以直接在 ”.emacs“ 中用 Emacs lisp 来设置。下面的例子包含了所有合法的搜索类型：

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING")
        ("u" tags "+boss-urgent")
        ("v" tags-todo "+boss-urgent")))
```

每个项的首字符串是使用调度器命令 C-c a 之后要给出的键以使用相应的命令。通常都是单个字符。第二个参数是搜索类型，接着是用来进行匹配的字符串或者正则表达式。上面的例子定义了：

C-c a w	对于包含关键字 ” “ 的 TODO 项的全局搜索。
C-c a u	对于带有标签 ”:boss:“ 而不含标签 ”:urgent:“ 的标题的全局标签搜索。
C-c a v	同搜索 C-c a u，但搜索范围只限于同时也是 TODO 项的标题。

扩展阅读

手册第 10 章
Mat Lundin’s tutorial about custom agenda commands
John Wiegley’s setup

11 准备导出

当导出 Org 模式的文档时，导出器在后端（backend）尽可能准确地反映出文档的结构。由于所要导出的目标文档像 HTML，LaTeX 和 DocBook 具有丰富的格式，Org 为富导出（rich export）提供了一些规则。这节概述 Org 模式缓冲区中的准备规则。

11.1 结构的组成元素

文档标题

导出文件的标题在特定行给出：

```
#+TITLE: This is the title of the document
```

标题和章节

第二章描述的大纲结构确定了导出文档的结构基础。然而由于大纲结构也用于（比如说）列表和任务，因此只有前三个级别用作标题。更深的级别会被看作项目列表。你可以通过变量 org-export-headline-levels 在全局设置这个开关，或者只是在单个文件中设置：

```
#+OPTIONS: H:4
```

目录表

目录表通常会直接插入在文档第一个标题之前。

```
#+OPTIONS: toc:2 (目录中只显示二级标题)
#+OPTIONS: toc:nil (无目录)
```

段落、分行和引用

段落之间至少要有一空行。如果你想实现段内分行，可以在行后加上“\\”。

要想在一个区域内实现分行，而其他地方使用正常格式，你可以使用下面的构造，它也可以用来实现诗歌的格式：

```
#+BEGIN_VERSE
 Great clouds overhead
 Tiny black birds rise and fall
 Snow covers Emacs

     -- AlexSchroeder
#+END_VERSE
```

当从另外一个文档中引用一段话时通过会让它左右都缩进。在 Org 文档中可以这样作引用：

```
#+BEGIN_QUOTE
Everything should be made as simple as possible,
but not any simpler -- Albert Einstein
#+END_QUOTE
```

如果你想让某些文本居中，可以这样：

```
#+BEGIN_CENTER
Everything should be made as simple as possible, \\
but not any simpler
#+END_CENTER
```

强调和等宽

你可以让文字 **粗体**， *斜体*， 下划线， 代码，以及 逐文本，如果必需，也可以'划掉'。代码和逐文本的字符串不会以 Org 模式的语法格式来处理，会被逐字输出。想要插入一个水平格尺[17]，用一个只含有破折号的行来实现，要求至少有 5 个破折号。

注释行

以‘#‘位于第 0 列的行会被看作注释，不会被导出。如果你想要一个缩进的行也被作为注释，用“#+”开头。另外以关键字 “COMMENT” 开头的子树整个树都不会被导出。最后，被 “#+BEGIN_COMMENT“ … ”#+END_COMMENT” 包围的整个区域也都不会被导出。

C-c ； 在一个项的开头触发 COMMENT 关键字

11.2 图片和表格

对于 Org 模式的表格，以竖直线开头的行会成为表格的首行。你可以在表格前面用下面几行为表格指定标题和标签，以方便交叉引用，在文本中可以用 `\ref{tab:basic-data}` 来引用它：

```
#+CAPTION: This is the caption for the next table (or link)
#+LABEL: tbl:basic-data
   | ... | ... |
   |-----+-----|
```

一些后端（HTML，LaTeX，以及 DocBook）允许直接插入图片到导出的文档中。Org 也可以，只要图片的链接不含有描述部分就行了，例如：

```
[[./img/a.jpg]]
```

如果你希望为图片定义一个标题，或者一个标签方便内部交叉引用，可以让图片单独一行，在前面加上：

```
 #+CAPTION: This is the caption for the next figure link (or table)
 #+LABEL: fig:SED-HR4049
 [[./img/a.jpg]]
```

你也可以为图形指定一些其他的特性。但由于这与后端[18]密切相关，可以参考关于特定后端的章节获得详细信息。

11.3 纯文本的例子

你可以包含进来一些纯文本的例子，这不属于准备的范畴。这些例子会等宽排版，所以适用于代码以及其他类似的情况：

```
#+BEGIN_EXAMPLE
Some example from a text file.
#+END_EXAMPLE
```

为了简单化，一些小型的例子也可以将各行以冒号开头。冒号前面可以有空格：

```
Here is an example
   : Some example from a text file.
```

对于一些程序设计语言的源代码以及一些其他的文本，可以被 Emacs 的字体锁（font-lock）特殊标记，你也可以让它们像在 Emacs 的缓冲区中那样显示：

```
#+BEGIN_SRC emacs-lisp
(defun org-xor (a b)
   "Exclusive or."
   (if a (not b) b))
#+END_SRC
```

为了能在支持这种语言的专门的缓冲区中编辑例子，可以用 C-c , 启动和退出编辑缓冲区。

11.4 包含文件

当导出文档时，你可以包含其他文件中的内容。比如，想包含你的“.emacs”文件，你可以用：

```
#+INCLUDE: "~/.emacs" src emacs-lisp
```

可选的第二个第三个参数是组织方式（例如，“quote”，“example”，或者“src”），如果是 “src”，语言用来格式化内容。组织方式是可选的，如果不给出，文本会被当作 Org 模式的正常处理。用 C-c ,可以访问包含的文件。

11.5 嵌入 LaTex

对于需要包含数学符号和特殊方程的科学笔记，Org 模式支持嵌入 LaTeX 代码到文件中。你可以直接使用类 TeX 的宏来输入特殊符号，输入方程，或者整个 LaTeX 环境。

```
Angles are written as Greek letters \alpha, \beta and \gamma. The mass if
the sun is M_sun = 1.989 x 10^30 kg. The radius of the sun is R_{sun} =
6.96 x 10^8 m. If $a^2=b$ and $b=2$, then the solution must be either
$a=+\sqrt{2}$ or $a=-\sqrt{2}$.

\begin{equation}
x=\sqrt{b}
\end{equation}
```

特殊设置之后，导出 HTML 时 LaTeX 代码片断会生成图片并包含进来。

扩展阅读

手册第 11 章

12 导出

Org 模式文档可以导出成多种格式：ASCII 用于包含在邮件中；HTML 用来发布到网页上；LaTeX/PDF 用来打印出漂亮的文档；DocBook 通过 DocBook 工具转换成其他各种各样的格式。也可以导出成 iCalendar 格式，将计划信息并入到桌面日历中。

12.1 导出选项

导出器能识别缓冲区中提供附加信息的特殊行。这样行可以放在文件中的任何地方。整个集合可以用 C-c C-e t 插入到缓冲区中。

C-c C-c t　插入导出选项模板，见下面的例子

```
#+TITLE:       the title to be shown (default is the buffer name)
#+AUTHOR:      the author (default taken from user-full-name)
#+DATE:        a date, fixed, of a format string for format-time-string
#+EMAIL:       his/her email address (default from user-mail-address)
#+DESCRIPTION: the page description, e.g. for the XHTML meta tag
#+KEYWORDS:    the page keywords, e.g. for the XHTML meta tag
#+LANGUAGE:    language for HTML, e.g. ‘en’ (org-export-default-language)
#+TEXT:        Some descriptive text to be inserted at the beginning.
#+TEXT:        Several lines may be given.
#+OPTIONS:     H:2 num:t toc:t \n:nil @:t ::t |:t ^:t f:t TeX:t ...
#+LINK_UP:     the ``up'' link of an exported page
```

```
#+LINK_HOME:   the ``home'' link of an exported page
#+LATEX_HEADER: extra line(s) for the LaTeX header, like \usepackage{xyz}
```

12.2 导出调度器

所有的导出命令都可以通过导出调度器来使用，调度器是一个前缀快捷键，它会提示输入一个字母来指定命令。通常整个文件都会被导出，但是如果选中区域包含大纲树，就会导出大纲树，并以第一个标题作为文件标题。

C-c C-e	用来导出和发布的调度器

12.3 ASCII/Latin-1/UTF-8 的导出

ASCII 导出功能能给 Org 文件提供的一个简单易读的版本，它只包含纯 ASCII 文本。Latin-1 和 UTF-8 导出用它们能编码的特殊字符扩展了文件的功能。

C-c C-e a	导出 ASCII 文件
C-c C-e n 和 C-c C-e N	和上面的命令一样，但是用 Latin-1 编码 [19]
C-c C-e u 和 C-c C-e U	和上面的命令一样，但是用 UTF-8 编码

12.4 HTML 的导出

C-c C-e h	导出 HTML 文件
C-c C-e b	导出 HTML 文件并用浏览器打开

想要将 HTML 以纯文本方式复制到导出文件，可以：

```
#+HTML: Literal HTML code for export
```

或者

```
#+BEGIN_HTML
All lines between these markers are exported literally
#+END_HTML
```

12.5 LaTeX 和 PDF 的导出

C-c C-e l	导出 LaTeX 文件
C-c C-e p	导出 LaTeX 文件，并处理成 PDF 文件
C-c C-e d	导出 LaTeX 文件，处理成 PDF 文件，并打开

默认，LaTeX 输出是使用 article 类型。但你可以在文件中通过选项 `#+LaTeX_CLASS: myclass` 来改变，但类型必须是 org-export-latex-classes 中列出的。

第 11.5 节撰述的内嵌的 LaTeX 可以正确地插入到 LaTeX 文件中。跟 HTML 导出器相似，也可以通过 `#+LaTeX:` 和 `#+BEGIN_LaTeX ... #+END_LaTeX` 来加入纯文本的 LaTex 代码。

12.6 DocBook 的导出

C-c C-e D	导出 DocBook 文件

跟 HTML 导出器相似，也可以通过 `#+DocBook:` 和 `#+BEGIN_DocBook ...` `#+END_DocBook` 结构来加入纯文本的 DocBook 代码。[20]

12.7 iCalendar 的导出

C-c C-e i	在一个 “.ice” 文件中为当前文件创建 iCalendar 项。
C-c C-e c	从 org-agenda-files 中的所有文件创建一个较大的 iCalendar 文件，并写入到 org-combined-agenda-icalendar-file 指定的文件中。

扩展阅读

手册第 12 章
Sebastian Rose’s image handling tutorial
Thomas Dye’s LaTeX export tutorial Eric Fraga’s BEAMER presentation tutorial

13 发布

Org 包含一个发布管理系统，可以配置一个由相互链接的 Org 文件组成的工程项目的自动向 HTML 转换。你也可以设置 Org，将导出的 HTML 页面和相应的附件如图片，源代码文件等自动上传到服务器。如何设置，详见手册。

下面是一个例子：

```
(setq org-publish-project-alist
      '(("org"
         :base-directory "~/org/"
         :publishing-directory "~/public_html"
         :section-numbers nil
         :table-of-contents nil
         :style "<link rel=\"stylesheet\"
                href=\"../other/mystyle.css\"
                type=\"text/css\"/>")))
```

C-c C-e C	提示指明一个项目，将所有的文件发布。
C-c C-e P	发布包含当前文件的项目。
C-c C-e F	只发布当前文件。
C-c C-e E	发布所有项目。

Org 用时间戳来查看文件是否改变。上面的命令只发布修改过的文件。你可以给它们加上前缀来强制重新发布所有的文件。

扩展阅读

手册第 1 章

Sebastian Rose's publishing tutorial
Ian Barton's Jekyll/blogging setup

14 处理源代码

Org 模式提供了一系列功能来处理源代码，包括源代码块的本地主模式编辑，代码块的运行(evaluation)，代码块的混合，以及以多种方式导出代码块和它们的结果。

代码块的结构

代码块的结构就像下面这样：

```
#+srcname: <name>
#+begin_src <language> <switches> <header arguments>
  <body>
#+end_src
```

其中<name>是代码块的名称，<language>指定代码块的语言（例如，emacs-lisp，shell，R，python，等等），<switches>用以控制代码块的导出，<header arguments>用来从多个方面控制代码块的行为，下面会详述，最后<body>是我们要写的代码。

编辑源代码

使用 C-c ,[21] 来编辑当前代码块。这个命令会新开一个以代码语言为主模式并包含代码的缓冲区（buffer）。保存这个缓冲区，会将新的内容写回 Org 缓冲区。再次使用 C-c , 退出这个缓冲区。

运行代码块

用 C-c C-c 运行当前代码块并将它们的结果插入 Org 缓冲区中。默认情况下，运行功能只对 emacs-lisp 代码块开启，但支持多种语言。所支持语言的完整列表见手册。下面是一个代码块和它的结果。

```
#+begin_src emacs-lisp
  (+ 1 2 3 4)
#+end_src

#+results:
: 10
```

抽取源代码

用 C-c C-v 将代码块从一个 Org 模式的文件中抽取到“杂货库”（Library of Babel）中，这样在所有的 Org 模式的缓冲区中都可以运行该代码。一个常用的代码块集合在 contrib/library-of-babel.org 中随 Org 一直发布。

头参数

运行和导出代码时的很多选项都通过头参数来设置。选项可以指定为全局的，文件级别的，大纲子树级别的，或者只是用于一个代码块。下面解释部分头参数。

:var	:var 头参数用来将参数传递给代码块。能用来传递给参数的值可以是直接量，org 模式表格中的值，文字实例块(literal example blocks)中的值，或者一个已命名代码块的结果。

:results	:result 头参数控制代码块结果的收集、类型和处理。output 和 value（默认）的值指定怎样在运行代码块时收集结果。vector，scalar，file， raw， html， latex 和 code 的值指定代码块结果的类型并以此确定将结果并入 Org 缓冲区的方式。silent， replace， prepend 和 append 指定处理代码块结果的方式，明确是否以及如何将结果插入 Org 缓冲区中。
:session	:session 头参数将会使代码块在 Emacs 的一个持续交互的底层进程（persistent interactive inferior process）中执行。这考虑到了代码运行的持续状态和运行结果的人工检查。
:exports	代码和块结果的任何组合在导出时都可以保持，这可以通过设置:results 头参数为 code results none 或者 both 来指定。
:tangle	头参数:tangle yes 将使代码块的内容到保存到一个以 Org 模式缓冲区命名的文件中。也可以通过:tangle filename 指明文件名。
:cache	头参数:cache yes 将使繁杂的代码块和结果关联，确保输入改变时代码块重运行。
:noweb	头参数:noweb 将扩展运行和混合时的”noweb“样式的引用。
:file	将代码块结果输出到文件时（比如，图形，表格，图表）可以用头参数:file filename，结果会被保存至指定的文件中，在 Org 缓冲区中插入一个到该文件的链接。

扩展阅读

手册第 11.3 节

The Babel site on Worg

15 杂项

15.1 补全

Org 支持用 M-TAB 进行缓冲区内部的补全。这种补全不需要利用 minibuffer。你只需要键入几个字母然后用快捷键在原位补全。例如，这个命令可以在 ‘\’后面补全 TeX 符号，在标题的开头补全 TODO 关键字，在 ‘:’之后补全标签。

15.2 一个更清晰的大纲视图

当 Org 标题含有很多星号并且标题下面的文字不缩进时，就会显得杂乱无章。当写一个图书结构的文件时，大纲标题就是实际章节的标题，基于列表机大纲，上面的问题就不会再有，缩进的结构也会更清晰：

```
* Top level headline
** Second level
*** 3rd level
some text
*** 3rd level
more text
* Another top level headline
```

如果你用的 Emacs 23.1.50.3 和 Org 6.29 的更高版本，这种视图可以用 org-indent-mode 模式动态地实现，它会在每行前面加上一些前导空格。你可以通过设置变量 org-startup-indented 为所有的文件打开 org-indent-mode 模式，或者用

```
#+STARTUP: indent
```

为单个文件打开缩进。 如果你想在 Emacs 或者 Org 的早期版本中实现同样的效果，或者想让缩进用硬空格符号，以使得纯文本文件看起来一样。Org 可以缩进标题下面的文本（用 TAB）;隐藏标题中的星号;只使用一级、三级等标题来为每级实现两个字符的缩进，从而实现这个功能。为了使这个特性在文件中支持，用：

```
#+STARTUP: hidestars odd
```

15.3 MobileOrg

MobileOrg 最初是由 Richard Moreland 为 iPhone/iPod Touch 系列设备开发的应用程序。Matt Jones 也为 Android 设备独立实现了一个版本。详见 Org 手册。

---

note 本文的英文原文是 Org 主页上的一篇简短手册（The compact Org-mode Guide ）。另外：

- 本文的脚注是双向链接的，你可以大胆地查看脚注而不用担心如何再定位到原文。
- 我不知道如何在引用的源代码中加脚注，如果你知道，希望你能告诉我。
- 我不知道如果在表格中正确显示 ‘|’，如果你知道，希望你能告诉我。
- 本文可能有错误，如果发现错误，请在评论中给出。

返回

Footnotes:

[1] 见说明。

[2] 当然你也可以专门用一个文件来记录待办事项，但这不是必需的。

[3] 对应的 buffer 中的设置是：#+STARTUP: logdone

[4] 对应的 buffer 中的设置是：#+STARTUP: lognotedone

[5] 跟所有的缓冲区内设置一样，用 C-c C-c 使行中的改变生效。

[6] 这跟通常意义上的 *安排一个会议* （ *scheduling a meeting* ）不同，后者只要在 Org 模式中插入一个不带关键字的时间戳就行了。

[7] 即使被标记为 DONE，在指定日期它依然会列在议程中。如果你不希望这样可以用变量 org-agenda-skip-scheduled-if-done 来设置。

[8] 截取位置是指保存截取信息的文件地址。

[9] 使用捕获模板，可以定义更细致的捕获地址，见[Capture templates]。

[10] 请设置你自己的快捷键，C-c c 只是一个建议。

[11] 这个表格实在翻译不通，以后再说吧。

[12] 注意，后两行中参数的顺序（月，日，年）依赖于 calendar-date-style 的设置。

[13] 原文是 C-p 有误。

[14] 原文是 C-n，有误。

[15] 在视图缓冲区的位置直接打开源位置，可能是版本的问题，并不删除其他的窗口。

[16] 不知道为什么我的版本不识别这个命令。

[17] 分隔线

[18] （backend，导出目标）

[19] 可能是版本的问题，这个命令和下面的命令在我的机器上没有。

[20] 原文说是 LaTeX 代码，有误。

[21] 前面的逗号是命令的一部分，下同。

Date: 2012-09-26 Wed

Author: Hu Wenbiao

Org version 7.8.11 with Emacs version 24

Validate XHTML 1.0