org 快速指南

JanSky

2024-07-27

目录

1 简介

Org 是一种使用快速有效的纯文本系统记笔记、维护待办事项列表和进行项目规划的模式。它也是一个创作和发布系统，支持使用源代码进行文学编程和可重复的研究。

本文档是 Org 手册的精简版。它包含所有基本功能和命令，以及自定义的重要提示。它适用于那些因为篇幅过大而不愿阅读 200 页手册的初学者。

1.1 安装

重要提示：如果您使用的 Org 版本是 Emacs 发行版的一部分，请跳过此部分并直接转到激活。将从网上下载的 org 模块添加到 Emacs 配置中以激活

```
(add-to-list 'load-path "~/path/to/orgdir/lisp")
```

1.2 激活

将以下几行添加到您的 Emacs 初始化文件中，为三个命令定义全局键，这些键在任何 Emacs 缓冲区（而不仅仅是 Org 缓冲区）中都很有用。可根据自身习惯自行选择合适的键。

```
(global-set-key (kbd "C-c l") #'org-store-link)
(global-set-key (kbd "C-c a") #'org-agenda)
(global-set-key (kbd "C-c c") #'org-capture)
```

1.3 反馈

如果您发现 Org 存在问题，或者您有疑问、评论或想法，请发送邮件至 Org 邮件列表 `mailto:emacs-orgmode@gnu.org`。有关如何提交错误报告的信息，请参阅主手册。

2 文档结构

Org 是一个大纲工具。大纲允许将文档组织成层次结构，对于我们来说，这是对笔记和想法的最佳表示。通过折叠可以快速浏览文档的结构，即隐藏文档的大部分内容以仅显示一般文档结构和当前正在处理的部分。Org 通过将整个显示和隐藏功能通过 TAB 键与 org-cycle 绑定的单个命令，大大简化了整体的使用。

2.1 标题

标题定义了大纲树的结构。Org 中的标题从左边距开始，以一个或多个星号开头，后跟一个空格。例如：

```
* 顶级标题
** 第二级
*** 第三级
    一些文本
*** 第三级
    更多文本
* 另一个顶级标题
```

请注意，命名的标题 org-footnote-section 默认为' 脚注'，被视为特殊。具有此标题的子树将被导出函数默认忽略。

有些人觉得很多星星太嘈杂，他们更喜欢轮廓上有空白，后面跟着一颗星星作为标题开头。请参阅杂项，了解实现此目的的设置。

2.2 可见的循环

TAB 功能可在缓冲区中隐藏文本。Org 模式使用两个命令，分别绑定到 TAB 和 S-TAB（org-cycle 和 org-shifttab），以改变缓冲区中的可见性。

- TAB (org-cycle)
  子树循环：在当前子树的状态之间轮换。

- S-TAB (org-global-cycle) | C-u TAB (org-cycle)
  全局循环：在整个缓冲区的状态之间轮换

当 Emacs 首次访问 Org 文件时，全局状态设置为概览，默认不隐藏文档内容。可以通过变量 org-startup-folded 进行配置，或者通过在文件中添加 STARTUP 关键字来进行文件级别的配置。关键字可以是 overview(隐藏全部内容)、content(仅显示标题包括子标题)、showall(文档内容全部显示)、showeverything 或 show<n>levels（n = 2..5），例如：

```
#+STARTUP: content
```

2.3 移动

以下命令用于跳转到缓冲区中的其他标题。

- C-c C-n (org-next-visible-heading)
  跳转到下一个标题。
- C-c C-p (org-previous-visible-heading)
  跳转到上一个标题。
- C-c C-f (org-backward-heading-same-level)
  跳转到下一个同级标题。
- C-c C-b (outline-backward-same-level)
  跳转到上一个同级标题。
- C-c C-u (outline-up-heading)
  向上跳转到更高层级的标题。

2.4 结构编辑

- M-RET (org-meta-return)
  插入与当前级别相同的新标题。如果光标在普通列表项中，则创建一个新项（参见普通列表）。当在行中间使用此命令时，当前行会被拆分，剩余部分成为新的标题。

- M-S-RET (org-insert-todo-heading)
  在当前标题级别插入新的 TODO 项目。

- TAB (org-cycle)
  在新的空条目中，TAB 循环切换合理的级别。

- M-LEFT (org-metaleft) | M-RIGHT (org-metaright)
  将当前标题提升或降低一级。

- M-UP (org-move-subtree-up) | M-DOWN (org-move-subtree-down)
  将子树上移或下移，即与同级的前一个或下一个子树交换位置。

- C-c C-w (org-refile)
  将条目或区域重新归档到不同的位置。参见“重新归档和复制”。

- C-x n s (org-narrow-to-subtree) | C-x n w (widen)
  将缓冲区限制到当前子树，然后再扩大它。

当有一个活动区域（瞬态标记模式）时，提升和降级操作将作用于该区域内的所有标题。

2.5 稀疏树

Org mode 的一个重要功能是能够为大纲树中的选定信息构建稀疏树，使整个文档尽可能地折叠。同时将选定的信息及其上方的标题结构显示出来。只需尝试一下，你就会立即明白它是如何工作的。

Org mode 包含几个创建这种树的命令，所有这些命令可以通过调度器访问：

- C-c / (org-sparse-tree)
  这将提示你输入一个额外的键来选择一个创建稀疏树的命令。

- C-c / r (org-occur)
  Occur。提示输入一个正则表达式，并显示一个包含所有匹配项的稀疏

树。每个匹配项也会被高亮显示；按下 C-c C-c 可以取消高亮显示。其他稀疏树命令根据 TODO 关键字、标签或属性来选择标题，这些将在本手册的后续部分讨论。

2.6 普通列表

在大纲树的条目中，手工格式化的列表可以提供额外的结构。它们还提供了一种创建复选框列表的方式（参见复选框）。Org 支持编辑这些列表，并且每个导出器（参见导出）都可以解析和格式化它们。

Org 支持有序列表、无序列表和描述列表。
无序列表项以‘-’，‘+’，或‘*’作为项目符号开始。
有序列表项以‘1.’或‘1)’开始。
描述列表使用‘::’来分隔术语和描述。

属于同一列表的项目必须在第一行具有相同的缩进。一个项目在下一个与其项目符号/编号对齐的行或更少的缩进之前结束。一个列表在所有项目都闭合后结束，或在两个空行之前结束。示例如下：

```
* 《魔戒》
  我最喜欢的场景是（按以下顺序）
  1. Rohirrim 的攻击
  2. Eowyn 与巫王的战斗
     + 这已经是我在书中的最喜欢的场景
     + 我非常喜欢 Miranda Otto。
  这部电影中的重要演员有：
  - Elijah Wood :: 他饰演弗罗多
  - Sean Astin :: 他饰演山姆，弗罗多的朋友。
```

当光标位于项目的第一行（带有项目符号或编号的行）时，以下命令会对项目执行操作。

- TAB (org-cycle)

项目可以像标题级别一样折叠。

- M-RET (org-insert-heading)
  在当前级别插入新项目。使用前缀参数时，强制插入新标题（请参见结构编辑）。
- M-S-RET (org-insert-todo-heading)
  插入一个带有复选框的新项目（请参见复选框）。
- M-UP (org-move-item-up) | M-DOWN (org-move-item-down)
  将当前项目（包括子项目）上移/下移（与相同缩进的前一个/下一个项目交换）。如果列表是有序的，自动重新编号。
- M-LEFT (org-do-promote) | M-RIGHT (org-do-demote)
  减少/增加项目的缩进，不影响子项目。
- M-S-LEFT (org-promote-subtree) | M-S-RIGHT (org-demote-subtree)
  减少/增加当前项目及其子项目的缩进。
- C-c C-c (org-toggle-checkbox)
  如果项目行中有复选框（参见复选框），切换复选框的状态。同时验证整个列表中的项目符号和缩进一致性。
- C-c - (org-cycle-list-bullet)
  在整个列表级别之间循环不同的项目符号（‘-’, ‘+’, ‘*’, ‘1.’, ‘1)’）。

3 表格

Org 配备了一个快速直观的表格编辑器。与 Emacs Calc 包（参见 GNU Emacs 计算器手册）结合使用时，支持类似电子表格的计算功能。Org 使得以纯 ASCII 格式化表格变得简单。任何以 ‘|’ 作为第一个非空白字符的行都被视为表格的一部分。‘|’ 也用作列分隔符。一个表格可能看起来像这样：

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

每次在表格内按 TAB、RET 或 C-c C-c 时，表格会自动重新对齐。TAB 还会移动到下一个字段（RET 移动到下一行），并在表格末尾或水平线之前创建新的表格行。表格的缩进由第一行设置。任何以‘|-’开头的行都被视为水平分隔线，并将在下次对齐时扩展以覆盖整个表格宽度。因此，要创建上述表格，你只需输入

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

然后按 TAB 来对齐表格并开始填写字段。更快的方法是输入‘|Name|Phone|Age’，然后按 C-c RET。在字段中输入文本时，Org 以特殊方式处理 DEL、Backspace 和所有字符键，以避免插入和删除操作导致其他字段发生位移。此外，当在使用 TAB、S-TAB 或 RET 将光标移动到新字段后立即开始输入时，该字段会自动变为空白。

- 创建和转换

  - C-c | (org-table-create-or-convert-from-region)
    将活动区域转换为表格。如果每行都包含至少一个 TAB 字符，函数会假定这些内容是以 TAB 分隔的。如果每行都包含逗号，则假定为逗号分隔值（CSV）。如果都不符合，行则会在空白处拆分为字段。如果没有活动区域，此命令将创建一个空的 Org 表格。但更简单的方法是直接开始输入，比如输入 | Name | Phone | Age RET | - TAB。

- 重新对齐和字段移动

  - C-c C-c (org-table-align): 重新对齐表格，但不移动光标。
  - TAB (org-table-next-field): 重新对齐表格，并移动到下一个字段（单元格）。如果在行末，会创建一个新行（如果需要）。
  - S-TAB (org-table-previous-field): 重新对齐表格，并移动到上一个字段。

- RET (org-table-next-row): 重新对齐表格，并移动到下一行。如果需要，会创建一个新行。
- S-UP (org-table-move-cell-up): 将当前单元格向上移动，通过与上方的单元格交换位置。
- S-DOWN (org-table-move-cell-down): 将当前单元格向下移动，通过与下方的单元格交换位置。
- S-LEFT (org-table-move-cell-left): 将当前单元格向左移动，通过与左侧的单元格交换位置。
- S-RIGHT (org-table-move-cell-right): 将当前单元格向右移动，通过与右侧的单元格交换位置。

- 列和行编辑

  - M-LEFT (org-table-move-column-left): 将当前列向左移动。
  - M-RIGHT (org-table-move-column-right): 将当前列向右移动。
  - M-S-LEFT (org-table-delete-column): 删除当前列。
  - M-S-RIGHT (org-table-insert-column): 在光标位置左侧插入一个新列。
  - M-UP (org-table-move-row-up): 将当前行向上移动。
  - M-DOWN (org-table-move-row-down): 将当前行向下移动。
  - M-S-UP (org-table-kill-row): 删除当前行或水平线。
  - M-S-DOWN (org-table-insert-row): 在当前行上方插入一行。带有前缀参数时，将在当前行下方创建一行。
  - C-c - (org-table-insert-hline): 在当前行下方插入一条水平线。带有前缀参数时，将在当前行上方插入一条水平线。
  - C-c RET (org-table-hline-and-move): 在当前行下方插入一条水平线，并将光标移动到该行下方的行中。
  - C-c ^ (org-table-sort-lines): 对表格中的行进行排序。

在指定区域内对表格行进行排序。光标位置指示用于排序的列，排序的范围是最接近的水平分隔线之间的行，或者是整个表格。

4 超链接

类似于 HTML，Org 也支持在文件内部、外部文件、Usenet 文章、电子邮件等地方使用链接。Org 可以识别普通的 URI，通常用尖括号括起来，并将其激活为可点击的链接。不过，一般的链接格式如下：

```
[[LINK][DESCRIPTION]]
```

或者：

```
[[LINK]]
```

- 处理链接
  Org 提供了多种方法来创建链接、将其插入到 Org 文件中，以及跟随链接。主要功能是 org-store-link，可以通过 M-x org-store-link 调用。由于其重要性，我们建议将其绑定到一个广泛使用的快捷键（参见激活）。该功能会存储当前位置的链接，以便稍后插入到 Org 缓冲区中（见下文）。在 Org 缓冲区中，以下命令用于创建、导航或更一般地操作链接：

  - C-c C-l (org-insert-link)：插入一个链接。该命令会提示输入要插入的链接。你可以直接输入一个链接，也可以使用历史记录键 UP 和 DOWN 访问已存储的链接。系统会提示你输入链接的描述部分。
  - 当以 C-u 前缀参数调用时，将使用文件名补全功能来链接到文件。
  - C-c C-l（当光标位于现有链接上）(org-insert-link)：当光标位于现有链接上时，C-c C-l 允许你编辑链接和描述部分。
  - C-c C-o (open-link-at-point)：打开光标所在位置的链接。

– C-c & (org-mark-ring-goto)：跳转回记录的位置。位置由内部链接的命令记录，并由 C-c % 记录。多次连续使用该命令可以在之前记录的位置环中循环移动。

5 TODO 项目

Org 模式并不要求 TODO 列表必须存在于单独的文档中。相反，TODO 项目可以作为笔记文件的一部分，因为 TODO 项目通常是在记录笔记时产生的！在 Org 模式中，只需将树形结构中的任何条目标记为 TODO 项目即可。这样，信息不会重复，TODO 项目也保留在其产生的上下文中。

Org 模式提供了多种方法来概览你需要完成的所有事项，这些事项可以从多个文件中收集。

5.1 基本 TODO 功能

任何标题如果以 TODO 去表示，例如：

```
*** TODO 写信给 Sam Fortune
```

与 TODO 条目一起使用的最重要的命令是：

- C-c C-t（org-todo）
  循环 TODO 状态

- S-RIGHT（org-shiftright）S-LEFT（org-shiftleft）
  选择下一个/前一个 TODO 状态，类似于循环。

- C-c / t（org-show-todo-tree）
  在稀疏树中查看 TODO 项（参见稀疏树）。折叠整个缓冲区，但显示所有 TODO 项（未完成状态）及其上方的标题层次结构。

- M-x org-agenda t（org-todo-list）
  显示全局 TODO 列表。将所有议程文件（参见议程视图）中的 TODO

项（未完成状态）收集到单个缓冲区中。有关更多信息，请参阅全局 TODO 列表。

- S-M-RET（org-insert-todo-heading）
  在当前 TODO 条目下方插入新的 TODO 条目。

5.2 多状态工作流

您可以使用 TODO 关键字来指示连续的工作进度状态：

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

垂直分隔符将“TODO”关键字（需要采取行动的状态）与“DONE”状态（不需要进一步行动的状态）分开。如果不提供分隔符，则最后一个状态会被用作“DONE”状态。在这种设置下，命令 C-c C-t 会将一个条目从“TODO”状态切换到“FEEDBACK”，接着是“VERIFY”，最后到“DONE”和“DELEGATED”。

有时你可能希望同时使用不同的 TODO 关键字集。例如，你可能希望拥有基本的“TODO=/=DONE”设置，同时也有一个用于修复 bug 的工作流程。那么你的设置可能会像这样：

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

关键词应该都不同，这有助于 Org 模式跟踪在给定条目中应使用哪个子序列。示例还展示了如何通过在每个关键词后添加括号中的字母来定义用于快速访问特定状态的键——在按下 C-c C-t 后，系统会提示输入键。要定义仅在单个文件中有效的 TODO 关键词，请在文件中的任何位置使用以下文本。

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

在更改了其中一行之后，请将光标保持在该行上，然后使用 C-c C-c 以使 Org 模式识别这些更改。

5.3 进度日志

要在更改 TODO 状态时记录时间戳和备注，可以使用带有前缀参数的 org-todo 命令。

- C-u C-c C-t（org-todo）
  提示输入备注并记录 TODO 状态更改的时间。

Org 模式还可以在将 TODO 项标记为 DONE 时自动记录时间戳，并可以选择性地添加备注，甚至可以在每次更改 TODO 项状态时都进行记录。这个系统高度可配置，设置可以按关键字定制，并且可以在文件或子树级别进行本地化。有关如何记录任务的工作时间的信息，请参阅“记录工作时间”。

- 关闭项目
  最基本的记录方式是跟踪某个 TODO 项被标记为完成的时间。这可以通过以下设置实现：

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

这样，每次将条目从 TODO（未完成）状态转换为任何 DONE 状态时，标题下方会插入一行 CLOSED: [时间戳]。如果你还想记录备注，可以使用：

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

此时系统会提示你输入备注，并将该备注以 Closing Note 作为标题存储在条目下方。

- 追踪 TODO 状态变更
  您可能希望追踪 TODO 状态的变化。您可以选择仅记录时间戳，或者记录带有时间戳的变更说明。这些记录会在标题后插入为项目化列表。当记录很多笔记时，您可能希望将这些笔记移入一个抽屉中。可以通

过自定义变量 org-log-into-drawer 来实现这一行为。对于状态日志记录，Org 模式要求按关键字配置。这是通过在每个关键字后面添加特殊标记!（用于时间戳）和 @（用于说明）来实现的。例如：

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

这段配置定义了 TODO 关键字和快速访问键，并且要求在条目状态设置为‘DONE’时记录时间，切换到‘WAIT’或‘CANCELED’时记录说明。当设置 org-todo-keywords 时，使用相同的语法也可以实现类似效果。

5.4 优先级

如果你大量使用 Org mode，你可能会有足够多的 TODO 项目，这时给它们排序就变得有意义了。可以通过在 TODO 项目的标题中添加优先级标记来进行排序，如下所示：

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

Org mode 支持三种优先级：‘A’，‘B’，和‘C’。‘A’是最高优先级，‘B’是默认优先级，如果没有指定则使用‘B’。优先级只在日程表中起作用。

- C-c , (org-priority)
  设置当前标题的优先级。按 A、B 或 C 选择优先级，按 SPC 移除标记。
- S-UP (org-priority-up) S-DOWN (org-priority-down)
  增加/减少当前标题的优先级。

5.5 将任务拆分为子任务

通常建议将大型任务拆分为更小、更易于管理的子任务。你可以通过在 TODO 项目下创建一个大纲树来完成这项工作，在树下列出详细的子任务。为了保持对已完成子任务比例的概览，可以在标题的任何位置插入‘[/]’或‘[%]’。这些标记会在子任务的 TODO 状态发生变化时更新，或者在按下 C-c C-c 时更新。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

在普通列表中的每一项都可以通过在项目前添加字符串‘[ ]’来变成复选框。复选框不会被纳入全局 TODO 列表，因此它们非常适合将一个任务拆分成多个简单步骤。以下是一个复选框列表的示例：

```
* TODO Organize party [2/4]
- [-] call people [1/2]
  - [ ] Peter
  - [X] Sarah
- [X] order food
```

5.7 复选框的层级结构

复选框具有层级结构，因此如果一个复选框项下有子项，这些子项也是复选框，当你切换子项的状态时，父项的复选框会反映出子项的状态：即是否所有子项、一些子项或没有子项被选中。以下命令适用于复选框：

- C-c C-c, C-u C-c C-c (org-toggle-checkbox)
  切换复选框的状态，或者使用前缀参数切换当前复选框项的存在。

- M-S-RET (org-insert-todo-heading)
  在普通列表项中插入一个带复选框的新项。这只在光标已经位于普通列表项时有效（见“普通列表”）。

6 标签

一个有效的实现标签和上下文以便交叉关联信息的方法是将标签分配给标题。Org mode 对标签提供了广泛的支持。每个标题都可以包含一个标签列表；标签出现在标题的末尾。标签是包含字母、数字、‘_’ 和 ‘@’ 的普通单词。标签必须由单个冒号包围，例如 ‘:work:’。可以指定多个标签，例如 ‘:work:urgent:’。默认情况下，标签以粗体显示，并与标题具有相同的颜色。

- 标签继承
  标签利用了大纲树的层级结构。如果一个标题有特定的标签，那么所有子标题也会继承该标签。例如，在下面的列表中：

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

  最终的标题具有标签 ‘work’，‘boss’，‘notes’，和 ‘action’，即使最终的标题并没有明确标记这些标签。你还可以设置文件中所有条目应该继承的标签，就像这些标签定义在一个假想的零级别，包围整个文件一样。使用如下格式的行来实现：

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

- 设置标签
  标签可以直接在标题末尾输入。在冒号后面，使用 M-TAB 可以对标签进行补全。还有一个用于插入标签的特殊命令：

  – C-c C-q (org-set-tags-command)
    输入当前标题的新标签。Org mode 提供标签补全或特殊的单键接口来设置标签，见下文。

  – C-c C-c (org-set-tags-command)
    当光标位于标题时，这个命令与 C-c C-q 执行相同的操作。

Org 支持基于标签列表的标签插入。默认情况下，这个列表是动态构建的，包含当前缓冲区中使用的所有标签。你也可以通过变量 org-tag-alist 全局指定一个固定的标签列表。最后，你可以使用 TAGS 关键字为特定文件设置默认标签，例如：

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

默认情况下，Org mode 使用标准的 minibuffer 补全功能来输入标签。然而，它也实现了另一种更快速的标签选择方法，称为快速标签选择。此方法允许你通过单个按键来选择和取消选择标签。为了使这一功能发挥作用，你应该为大多数常用标签分配唯一的字母。你可以通过在 Emacs 初始化文件中配置变量 org-tag-alist 来全局设置这些字母。例如，如果你发现需要在不同文件中为许多条目打上‘@home’标签，你可以设置如下：

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

如果标签仅对你正在处理的文件相关，你可以将 TAGS 关键字设置为：

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

- 标签组
  标签可以被定义为一组其他标签的组标签。组标签可以被看作是其标签集合的“更广泛的术语”。你可以通过使用括号并在组标签和相关标签之间插入冒号来设置组标签：

```
#+TAGS: [ GTD : Control Persp ]
```

或者，如果组中的标签应该是互斥的，可以使用花括号：

```
#+TAGS: { Context : @Home @Work }
```

当你搜索一个组标签时，它会返回组内及其子组中的所有成员的匹配项。在日程视图中，通过组标签进行筛选时，会显示或隐藏标记有组中

至少一个成员或任何子组的标题。如果你想暂时忽略组标签，可以使用 org-toggle-tags-groups 切换组标签支持，该命令绑定在 C-c C-x q。

- 标签搜索

  - C-c / m 或 *C-c * (org-match-sparse-tree)
    创建一个稀疏树，显示所有匹配标签搜索的标题。使用 C-u 前缀参数时，忽略非 TODO 行的标题。
  - M-x org-agenda m (org-tags-view)
    从所有日程文件中创建一个全局标签匹配列表。参见“匹配标签和属性”。
  - M-x org-agenda M (org-tags-view)
    从所有日程文件中创建一个全局标签匹配列表，但仅检查 TODO 项目。

这些命令都会提示输入匹配字符串，允许使用基本的布尔逻辑，比如 +boss+urgent-project1，查找标记为‘boss’和‘urgent’，但不包括‘project1’的条目，或者 Kathy|Sally，查找标记为‘Kathy’或‘Sally’的条目。搜索字符串的完整语法丰富，还允许匹配 TODO 关键字、条目级别和属性。有关更详细的描述和多个示例，请参见“匹配标签和属性”。

7 属性

属性是与条目关联的键值对。它们存在于名为‘PROPERTIES’的特殊抽屉中。每个属性在一行上指定，键（用冒号包围）在前，值在后：

```
* CD collection
** Classic
*** Goldberg Variations
    :PROPERTIES:
```

```
:Title:     Goldberg Variations
:Composer:  J.S. Bach
:Publisher: Deutsche Grammophon
:NDisks:    1
:END:
```

你可以通过设置属性 Xyz_{ALL} 来定义特定属性 Xyz 的允许值。这个特殊的属性是可以继承的，因此如果你在级别 1 的条目中设置它，它会应用于整个树。当定义了允许的值后，设置相应的属性变得更加容易，并且更不容易出现输入错误。例如，对于 CD 收藏，我们可以这样预定义出版商和一个盒子中的磁盘数量：

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

如果你想设置可以被文件中任何条目继承的属性，可以使用如下行：

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

以下命令有助于处理属性：

- C-c C-x p (org-set-property)
  设置一个属性。此命令会提示输入属性名称和一个值。
- C-c C-c d (org-delete-property)
  从当前条目中删除一个属性。

要创建基于属性的稀疏树和特殊列表，可以使用与标签搜索相同的命令（参见“标签”）。搜索字符串的语法在“匹配标签和属性”中有描述。

8 日期和时间

为了协助项目规划，TODO 项目可以标记日期和时间。携带日期和时间信息的特殊格式字符串在 Org 模式中称为时间戳。这个术语可能会有些

令人困惑，因为时间戳通常用于指示某物何时创建或最后何时更改。然而，在 Org 模式中，这个术语的使用范围要广泛得多。时间戳可以用于规划约会、安排任务、设定截止日期、跟踪时间等。以下部分将描述时间戳的格式以及 Org 模式为处理时间和时间间隔的常见用例提供的工具。

8.1 时间戳

时间戳是以特殊格式指定日期——可能还包括时间或时间范围——的说明，例如 *<2003-09-16 周二 >*、*<2003-09-16 周二 09:39>* 或 *<2003-09-16 周二 12:00>–<2003-09-16 周二 12:30>*。时间戳可以出现在 Org 树条目的标题或正文中的任何位置。其存在会使条目在议程中显示在特定的日期（参见《每周/每日议程》）。我们区分：

- 普通时间戳、事件、预约
  简单时间戳仅将日期/时间分配给一个条目。这就像在纸质日程表中记录一个预约或事件一样。一个条目中可以有多个时间戳。

  ```
  * Meet Peter at the movies
    <2006-11-01 Wed 19:15>
  * Discussion on climate change
    <2006-11-02 Thu 20:00-22:00>
  * My days off
    <2006-11-03 Fri>
    <2006-11-06 Mon>
  ```

- 时间戳与重复间隔
  时间戳可能包含一个重复间隔，表示它不仅适用于给定的日期，还会在经过一定的 N 小时（h）、天（d）、周（w）、月（m）或年（y）的间隔后重复出现。以下内容将在每周三出现在日程中：

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

- 日记式表达条目

对于更复杂的日期规格，Org 模式支持使用在 Emacs 日历包中实现的特殊表达日记条目。例如，带有可选时间的条目：

```
* 22:00-23:00 The nerd meeting on every 2nd Thursday of the month
  <%%(diary-float t 4 2)>
```

- 时间范围
  时间范围是指两个时间单位通过‘-’连接的时间戳。

```
* Discussion on climate change
  <2006-11-02 Thu 10:00-12:00>
```

- 时间/日期范围
  两个时间戳通过‘–’连接表示一个范围。在日程中，标题会显示在范围的第一天和最后一天，以及范围内的任何显示日期。第一个示例仅指定了范围的日期，而第二个示例则为每个日期指定了时间范围。

```
** Meeting in Amsterdam
   <2004-08-23 Mon>--<2004-08-26 Thu>
** This weeks committee meetings
   <2004-08-23 Mon 10:00-11:00>--<2004-08-26 Thu 10:00-11:00>
```

- 非活动时间戳
  与普通时间戳类似，但使用方括号而不是尖括号。这些时间戳是非活动的，即它们不会触发条目在日程中显示。

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

8.2 创建时间戳

为了让 Org 模式识别时间戳，时间戳需要遵循特定的格式。以下所有命令都能生成正确格式的时间戳。

- C-c . (org-timestamp)

提示输入日期并插入相应的时间戳。当光标位于缓冲区中现有的时间戳上时，此命令用于修改该时间戳，而不是插入新的时间戳。当连续使用此命令两次时，会插入一个时间范围。使用前缀参数时，它还会添加当前时间。

- C-c ! (org-timestamp-inactive)
  与 C-c . 类似，但插入一个非活动时间戳，该时间戳不会导致日程条目出现。

- S-LEFT (org-timestamp-down-day) S-RIGHT (org-timestamp-up-day)
  将光标处的日期改为前一天或后一天。

- S-UP (org-timestamp-up) S-DOWN (org-timestamp-down)
  在时间戳的开始或包含括号上，改变其类型。在时间戳内部，改变光标所在的项目。光标可以在年份、月份、日期、小时或分钟上。当时间戳包含时间范围（如‘15:30-16:30’）时，修改第一个时间也会调整第二个时间，从而保持时间段的固定长度。要改变长度，请修改第二个时间。

当 Org 模式提示输入日期/时间时，它接受任何包含日期和/或时间信息的字符串，并智能地解释该字符串，从当前日期和时间推导出未指定信息的默认值。您也可以在弹出日历中选择日期。有关日期/时间提示的详细信息，请参阅手册。

8.3 截止日期与日程安排

时间戳前可以加上特定的关键字以便于计划安排：

- C-c C-d (org-deadline)
  在标题下方的行中插入“DEADLINE”关键字及时间戳。意义：任务—很可能是 TODO 项目，尽管不一定—预计在该日期完成。在截止日期当天，任务会出现在日程中。此外，今天的日程还会显示有关即将到期或错过的截止日期的警告，从截止日期前的 org-deadline-warning-days 天开始，直到条目标记为完成为止。一个示例：

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

- C-c C-s (org-schedule)
  在标题下方的行中插入“SCHEDULED”关键字及时间戳。意义：你计划在给定日期开始处理该任务。标题会在给定日期下列出。此外，今天的汇总中会出现一个提醒，告知计划日期已经过去，直到条目标记为完成，即任务会自动推迟直到完成。

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

有些任务需要反复执行。Org 模式通过在“DEADLINE”、“SCHEDULED”或普通时间戳中使用所谓的重复器来帮助组织这些任务。以下是一个示例：

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

其中的“+1m”是一个重复器；其含义是该任务的截止日期是“*<2005-10-01* 周六 *>*”，并且从那时起每个月重复一次。

8.4 记录工作时间

Org 模式允许你记录在项目中特定任务上花费的时间。

- C-c C-x C-i (org-clock-in)
  开始记录当前项目的时间（记录开始）。这会插入“CLOCK”关键字和一个时间戳。当使用 C-u 前缀参数调用时，可以从最近记录的任务列表中选择任务。

- C-c C-x C-o (org-clock-out)
  停止记录时间（记录结束）。这会在上次启动计时的位置插入另一个时间戳，并直接计算结果时间并以“=>HH”的格式插入在时间范围之后。

- C-c C-x C-e (org-clock-modify-effort-estimate)
  更新当前计时任务的工作量估算。

- C-c C-x C-q (org-clock-cancel)
  取消当前计时。这在计时错误启动或转而处理其他任务时非常有用。

- C-c C-x C-j (org-clock-goto)
  跳转到当前计时任务的标题。使用 C-u 前缀参数时，从最近记录的任务列表中选择目标任务。

在日程中可以使用 l 键（参见《每周/每日日程》）来显示在一天内被处理或关闭的任务。

9 捕捉、归档、存档

任何组织系统的重要部分是能够快速捕捉新的想法和任务，并将相关的参考资料与其关联。Org 模式通过一个叫做捕捉（capture）的过程来实现这一点。它还可以将与任务相关的文件（附件）存储在一个特殊目录中。任务和项目一旦进入系统，需要进行移动。将完成的项目树移动到归档文件中可以保持系统的紧凑和高效。

9.1 捕捉

捕捉功能让你可以在工作流程中几乎没有中断地快速存储笔记。你可以为新的条目定义模板，并将它们关联到不同的目标位置以存储笔记。

- 设置捕捉
  以下自定义设置了一个默认的笔记目标文件：

  ```
  (setq org-default-notes-file (concat org-directory "/notes.org"))
  ```

  你还可以为捕捉新资料定义一个全局快捷键（见激活）。

- 使用捕捉

- M-x org-capture (org-capture)
  启动捕捉过程，将你置于一个缩小的间接缓冲区中进行编辑。
- C-c C-c (org-capture-finalize)
  在你完成在捕捉缓冲区中输入信息后，按 C-c C-c 将你带回到捕捉过程之前的窗口配置，以便你可以继续工作而不再受到干扰。
- C-c C-w (org-capture-refile)
  通过将笔记重新归档到不同的位置来完成捕捉过程（参见归档和复制）。
- C-c C-k (org-capture-kill)
  中止捕捉过程并返回到之前的状态。

- 捕捉模板
  你可以为不同类型的捕捉条目和不同的目标位置使用模板。例如，你希望使用一个模板来创建一般的 TODO 条目，并将这些条目放在文件 ~/org/gtd.org 中的“Tasks”标题下。同时，文件 journal.org 中的日期树应该用于捕捉日记条目。一个可能的配置如下：

```
(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")))
```

  如果你从捕捉菜单中按下 t，Org 会为你准备模板，如下所示：

```
* TODO
  [[file:LINK TO WHERE YOU INITIATED CAPTURE]]
```

  在模板展开过程中，特殊的%-转义字符允许动态插入内容。以下是一些可能性的小选集，详细信息请参阅手册：‘%a’ 注释，通常是通过 org-store-link 创建的链接‘%i’初始内容，当捕捉使用 C-u 调用时的区域‘%t’，‘%T’时间戳，仅日期，或日期和时间‘%u’，‘%U’如上，但为非活动时间戳‘%?’完成模板后，光标位置

9.2 归档和复制

在审查捕捉的数据时，你可能想将一些条目重新归档或复制到不同的列表中，例如项目。剪切、找到正确的位置，然后粘贴笔记是繁琐的。为了简化这一过程，你可以使用以下特殊命令：

- C-c C-w (org-agenda-refile)
  将光标处的条目或区域重新归档。此命令提供可能的归档位置，并让你通过自动补全选择一个。条目（或区域中的所有条目）会作为子项归档到目标标题下。默认情况下，当前缓冲区中的所有 1 级标题都被视为目标，但你可以在多个文件中定义更复杂的目标。有关详细信息，请参见变量 org-refile-targets。

- C-u C-c C-w (org-agenda-refile)
  使用归档界面跳转到一个标题。

- C-u C-u C-c C-w (org-refile-goto-last-stored)
  跳转到 org-refile 上次移动树的位置。

- C-c M-w (org-refile-copy)
  复制的工作方式与归档类似，只是原始笔记不会被删除。

9.3 归档

当一个由（子）树表示的项目完成时，你可能希望将该树移到其他地方，并停止它对日程的影响。归档对于保持工作文件的紧凑性和确保全局搜索（如构建日程视图）的速度非常重要。最常见的归档操作是将项目树移动到另一个文件，即归档文件。

- C-c C-x C-a (org-archive-subtree-default)
  使用 org-archive-default-command 变量中指定的命令归档当前条目。

- C-c C-x C-s 或简写为 C-c $ (org-archive-subtree)
  将光标位置开始的子树归档到由 org-archive-location 给定的位置。

默认的归档位置是与当前文件位于同一目录中的文件，其名称是将“$_{\text{archive}}$”附加到当前文件名后形成的。你还可以选择归档条目归档到哪个标题下，也可以将其添加到文件中的日期树中。有关如何指定文件和标题的信息和示例，请参见变量 org-archive-location 的文档字符串。在缓冲区内也可以设置此变量，例如：

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

10 议程视图

由于 Org 的工作方式，TODO 项目、带时间戳的项目和标记的标题可能会分散在一个文件中，甚至在多个文件中。为了获得开放的行动项目或特定日期的重要事件的概览，需要将这些信息收集、排序并以有组织的方式展示出来。提取的信息会显示在一个特殊的议程缓冲区中。这个缓冲区是只读的，但提供了访问原始 Org 文件中相应位置的命令，甚至可以远程编辑这些文件。从议程缓冲区进行远程编辑意味着，例如，你可以在议程缓冲区中更改截止日期和约会的日期。有关议程缓冲区中可用命令的更多信息，请参见议程缓冲区中的命令。

10.1 议程文件

要显示的信息通常会从所有议程文件中收集，这些文件列在变量 org-agenda-files 中。

- C-c [(org-agenda-file-to-front)
  将当前文件添加到议程文件列表中。文件会被添加到列表的前面。如果它已经在列表中，它会被移到前面。如果使用前缀参数，文件会被添加/移动到列表的末尾。

- C-c] (org-remove-file)
  从议程文件列表中移除当前文件。

- C-’

- C-, (org-cycle-agenda-files)
  循环浏览议程文件列表，依次访问每个文件。

10.2 议程调度器

视图是通过调度器创建的，可以通过 M-x org-agenda 访问，或者更好的是，绑定到全局快捷键（见激活）。调度器显示一个菜单，需要额外的字母来执行命令。调度器提供了以下默认命令：

- a
  创建类似日历的议程（见《每周/每日议程》）。
- t、T
  创建所有 TODO 项目的列表（见《全局 TODO 列表》）。
- m、M
  创建匹配给定表达式的标题列表（见《匹配标签和属性》）。
- s
  创建通过布尔表达式选择的条目列表，表达式由关键字和/或正则表达式组成，这些关键字和/或正则表达式必须或必须不出现于条目中。

10.3 每周/每日议程

每周/每日议程的目的是像纸质日程表的一页一样，显示当前一周或一天的所有任务。

- M-x org-agenda a (org-agenda-list)
  从 Org 文件列表中编译当前周的议程。议程会显示每一天的条目。

Org 模式理解日历的语法，并允许你在 Org 文件中直接使用日历表达式条目：

```
* Holidays
  :PROPERTIES:
  :CATEGORY: Holiday
```

```
  :END:
%%(org-calendar-holiday)   ; special function for holiday names

* Birthdays
  :PROPERTIES:
  :CATEGORY: Ann
  :END:
%%(org-anniversary 1956  5 14) Arthur Dent is %d years old
%%(org-anniversary 1869 10  2) Mahatma Gandhi would be %d years old
```

Org 还可以与 Emacs 的预约通知功能互动。要将议程文件的预约添加到通知中，可以使用命令 org-agenda-to-appt。

10.4 全局 TODO 列表

全局 TODO 列表包含所有未完成的 TODO 项目，并将其格式化并集中到一个地方。远程编辑 TODO 项目使你可以通过单个按键来更改 TODO 条目的状态。有关 TODO 列表中可用的命令，请参见《议程缓冲区中的命令》。

- M-x org-agenda t (org-todo-list)
  显示全局 TODO 列表。该列表将所有议程文件中的 TODO 项目（见《议程视图》）收集到一个缓冲区中。

- M-x org-agenda T (org-todo-list)
  与上面的命令类似，但允许选择特定的 TODO 关键字。

10.5 匹配标签和属性

如果议程文件中的标题标记了标签（见《标签》）或具有属性（见《属性》），你可以根据这些元数据选择标题，并将它们收集到议程缓冲区中。这里描述的匹配语法在创建稀疏树时也适用（按 C-c / m）。

- M-x org-agenda m (org-tags-view)

生成一个所有匹配给定标签集的标题列表。该命令会提示输入选择标准，这是一种使用标签的布尔逻辑表达式，例如 +work+urgent-withboss 或 work|home（见《标签》）。如果你经常需要特定的搜索，可以为其定义一个自定义命令（见《议程调度器》）。

- M-x org-agenda M (org-tags-view)
  类似于上面的命令，但仅选择那些也是 TODO 项目的标题。

搜索字符串可以使用布尔操作符 & 代表 AND 和 | 代表 OR。& 的优先级高于 |。目前不支持括号。搜索中的每个元素可以是标签、匹配标签的正则表达式，或者像 PROPERTY OPERATOR VALUE 这样的表达式，其中包含比较操作符，用于访问属性值。每个元素前可以加 - 来表示排除，并且 + 是正选的语法糖。当存在 + 或 - 时，AND 操作符 & 是可选的。以下是一些示例，仅使用标签：

- +work-boss
  选择标记为 work 的标题，但排除那些也标记为 boss 的标题。

- work|laptop
  选择标记为 work 或 laptop 的标题。

- work|laptop+night
  类似于之前的命令，但要求标记为 laptop 的标题也必须标记为 night。

你还可以同时测试属性与匹配标签，详情请参阅手册。

10.6 搜索视图

此议程视图是一个通用的文本搜索功能，用于 Org 模式条目。它特别有用来查找笔记。

- M-x org-agenda s (org-search-view)
  这是一个特殊的搜索功能，可以通过匹配子字符串或特定单词来选择条目，支持布尔逻辑。

例如，搜索字符串‘computer equipment’匹配包含‘computer equipment’作为子字符串的条目。搜索视图还可以使用布尔逻辑搜索条目中的特定关键字。搜索字符串‘+computer +wifi -ethernet -{8\.11[bg]}’匹配包含关键字‘computer’和‘wifi’的条目，但不包含关键字‘ethernet’，并且也不匹配正则表达式‘8\.11[bg]’，即排除‘8.11b’和‘8.11g’。请注意，除了议程文件之外，此命令还会搜索 org-agenda-text-search-extra-files 列出的文件。

10.7 议程缓冲区中的命令

议程缓冲区中的条目链接回它们来源的 Org 文件或日历文件。你不能直接编辑议程缓冲区本身，但提供了命令来显示并跳转到原始条目位置，并从议程缓冲区“远程”编辑 Org 文件。这只是众多命令中的一部分，查看议程菜单和手册以获取完整列表。

- 移动
  - n (org-agenda-next-line)
    下一行（与 DOWN 和 C-n 相同）。
  - p (org-agenda-previous-line)
    上一行（与 UP 和 C-p 相同）。
- 查看/跳转到 Org 文件
  - SPC (org-agenda-show-and-scroll-up)
    在另一个窗口中显示条目的原始位置。使用前缀参数时，确保抽屉保持折叠状态。
  - TAB (org-agenda-goto)
    跳转到条目的原始位置，在另一个窗口中打开。
  - RET (org-agenda-switch-to)
    跳转到条目的原始位置，并关闭其他窗口。

- 更改显示

  - o (delete-other-windows)
    关闭其他窗口。
  - v d 或短 d (org-agenda-day-view)
    切换到天视图。
  - v w 或短 w (org-agenda-week-view)
    切换到周视图。
  - f (org-agenda-later)
    向未来推进，显示当前时间段之后的内容。例如，如果当前显示的是一周，切换到下一周。
  - b (org-agenda-earlier)
    向过去推进，显示更早的日期。
  - . (org-agenda-goto-today)
    跳转到今天。
  - j (org-agenda-goto-date)
    提示输入一个日期并跳转到该日期。
  - v l 或 v L 或短 l (org-agenda-log-mode)
    切换到日志模式。在日志模式下，显示在日志开启时标记为完成的条目（见变量 org-log-done），以及当天已记录时间的条目。当使用 C-u 前缀参数调用时，显示所有可能的日志条目，包括状态变化。
  - r g (org-agenda-redo)
    重新创建议程缓冲区，例如在修改条目的时间戳后更新显示。
  - s (org-save-all-org-buffers)
    保存当前 Emacs 会话中的所有 Org 缓冲区，以及 ID 的位置。

- 远程编辑

- 0–9
  数字参数。
- t (org-agenda-todo)
  更改条目的 TODO 状态，同时在议程和原始 Org 文件中进行更新。
- C-k (org-agenda-kill)
  删除当前议程项目及其在原始 Org 文件中所属的整个子树。
- C-c C-w (org-agenda-refile)
  重新归档光标所在的条目。
- a (org-agenda-archive-default-with-confirmation)
  使用在 org-archive-default-command 中设置的默认归档命令归档光标所在条目对应的子树，并要求确认。
- $ (org-agenda-archive)
  归档当前标题对应的子树。
- C-c C-s (org-agenda-schedule)
  为该条目安排时间。使用前缀参数时，移除安排的时间戳。
- C-c C-d (org-agenda-deadline)
  为该条目设置截止日期。使用前缀参数时，移除截止日期。
- S-RIGHT (org-agenda-do-date-later)
  将当前行关联的时间戳提前一天。
- S-LEFT (org-agenda-do-date-earlier)
  将当前行关联的时间戳推迟一天。
- I (org-agenda-clock-in)
  开始记录当前条目的时间。
- O (org-agenda-clock-out)
  停止之前开始的时间记录。
- X (org-agenda-clock-cancel)
  取消当前正在运行的时间记录。

  - J (org-agenda-clock-goto)
    跳转到另一个窗口中的正在运行的时间记录。

- 退出和退出

  - q (org-agenda-quit)
    退出议程，关闭议程缓冲区。
  - x (org-agenda-exit)
    退出议程，关闭议程缓冲区以及所有为编译议程而加载的 Emacs 缓冲区。

10.8 自定义议程视图

自定义搜索的第一个应用是为常用的搜索定义键盘快捷键，这些搜索可以是创建议程缓冲区，也可以是稀疏树（后者仅适用于当前缓冲区）。自定义命令在变量 org-agenda-custom-commands 中配置。你可以通过在议程调度器中按 C 来自定义此变量（见《议程调度器》）。也可以直接在 Emacs 初始化文件中使用 Emacs Lisp 进行设置。以下示例包含所有有效的议程视图：

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

每个条目中的初始字符串定义了在调度器命令后需要按下的键，以便访问该命令。通常这只是一个字符。第二个参数是搜索类型，后跟用于匹配的字符串或正则表达式。上述示例将定义：

- w
  作为一个全局搜索，查找 TODO 条目中关键字为 WAITING 的条目。

- u
  作为一个全局标签搜索，查找标记为 boss 但不标记为 urgent 的标题。

- v
  同样的搜索，但限制为那些也是 TODO 项目的标题。

11 富内容的标记

Org 主要用于组织和搜索纯文本笔记。然而，它也提供了一种轻量但强大的标记语言，用于富文本格式化及其他功能。与导出框架（见《导出》）结合使用时，你可以在 Org 中创作出美观的文档。

11.1 段落

段落之间至少有一个空行。如果您需要在段落内强制换行，请使用'\ 在 ´ 行末。为了保留区域中的换行符、缩进和空行，但在其他方面使用正常格式，您可以使用此构造，它也可以用于格式化诗歌。

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

    ---AlexSchroeder
#+END_VERSE
```

引用另一份文档中的一段文字时，通常将其格式化为左右边距均缩进的段落。您可以像这样在 Org 文档中包含引文：

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

如果您想要将一些文本居中，请这样做：

```
#+BEGIN_CENTER
Everything should be made as simple as possible, \\
```

```
but not any simpler
#+END_CENTER
```

11.2 强调和等宽字体

你可以造词' **大胆的**'，' *斜体*'，'_下划线_'，' `逐字`' 和' `代码`'，如果你必须，'~~删除线~~'. 代码和逐字字符串中的文本不会根据 Org 特定语法进行处理；而是逐字导出。

11.3 嵌入式 LaTeX

对于需要包含数学符号和偶尔出现的公式的科学笔记，Org 模式支持将 LaTeX 代码嵌入到其文件中。您可以直接使用类似 TeX 的语法来输入特殊符号、公式和整个 LaTeX 环境。

```
The radius of the sun is R_sun = 6.96 x 10^8 m.  On the other hand,
the radius of Alpha Centauri is R_{Alpha Centauri} = 1.28 x R_{sun}.

\begin{equation}                        % arbitrary environments,
x=\sqrt{b}                              % even tables, figures
\end{equation}                          % etc

If $a^2=b$ and \( b=2 \), then the solution must be
either $$ a=+\sqrt{2} $$ or \[ a=-\sqrt{2} \].
```

11.4 文字示例

您可以添加不应受标记约束的文字示例。此类示例采用等宽字体排版，因此非常适合源代码和类似示例。

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

为了在使用小示例时简单起见，您还可以以冒号加空格作为示例行的开头。冒号前还可以有额外的空格：

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

如果示例是来自编程语言的源代码，或者任何其他可以通过 Emacs 中的 Font Lock 标记的文本，您可以要求示例看起来像字体化的 Emacs 缓冲区。

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

#+end$_{\text{src}}$ 要在支持该语言的特殊缓冲区中编辑示例，请使用 C-c ' 进入和离开编辑缓冲区。

11.5 图像

图像是指向没有描述部分的图像文件的链接，例如

```
./img/cat.jpg
```

如果您希望为图像定义标题，或者为内部交叉引用定义标签（请参阅超链接），请确保链接单独成行，并在其前面加上' 标题' 和' 姓名' 关键字如下：

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

11.6 创建脚注

脚注在段落中定义，以第 0 列中方括号内的脚注标记开头，不允许缩进。脚注引用只是文本中方括号内的标记。例如：

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

以下命令处理脚注：

- C-c C-x f（org-footnote-action）
  脚注操作命令。当光标位于脚注引用上时，跳转到定义。当光标位于定义上时，跳转到（第一个）引用。否则，创建一个新的脚注。当使用前缀参数调用此命令时，会提供一个附加选项菜单，包括重新编号。

- C-c C-c（org-ctrl-c-ctrl-c）
  在定义和参考之间跳转。

12 导出

org 可以将文档转换并导出为多种其他格式，同时尽可能保留文档的结构（参见文档结构）和标记（参见富文本标记）。

12.1 导出调度器

导出调度器是 Org 导出的主要界面。一个层级菜单展示了当前配置的导出格式。选项以简易切换开关的形式显示在同一屏幕上。

- C-c C-e (org-export-dispatch)
  调用导出调度器界面。

默认情况下，Org 会导出整个缓冲区。如果 Org 缓冲区中有活动区域，则 Org 只导出该区域。

12.2 导出设置

导出器会识别缓冲区中的特殊行，以提供额外的信息。这些行可以放置在文件中的任何位置：

```
#+TITLE: I'm in the Mood for Org
```

主要的导出选项包括：

- ‘TITLE’：要显示的标题

- ‘AUTHOR’：作者（默认为 user-full-name）
- ‘DATE’：日期，可以是固定日期或 Org 时间戳
- ‘EMAIL’：电子邮件地址（默认为 user-mail-address）
- ‘LANGUAGE’：语言代码，例如’en’

可以通过在导出调度器中使用’Insert template’ 命令（按 # 键）来插入选项关键字集。

12.3 目录

目录包括文档中的所有标题。因此，其深度与文件中的标题级别相同。如果需要使用不同的深度或完全关闭目录，请相应地设置 org-export-with-toc 变量。您也可以在每个文件中使用以下 OPTIONS 关键字项来实现相同的效果：

```
#+OPTIONS: toc:2          (only include two levels in TOC)
#+OPTIONS: toc:nil        (no default TOC at all)
```

Org 通常会在文件的第一个标题之前直接插入目录。

12.4 包含文件

在导出过程中，您可以包含另一个文件的内容。例如，要包含您的.emacs 文件，可以使用：

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

第一个参数是要包含的文件名。可选的第二个参数指定块类型：‘example’，‘export’ 或 ‘src’。可选的第三个参数指定用于格式化内容的源代码语言。这与 ‘export’ 和 ‘src’ 块类型相关。您可以通过 C-c ’ 访问包含的文件。

12.5 注释行

以零个或多个空白字符开头，紧接着一个 # 和一个空白字符的行被视为注释，因此不会被导出。同样，被'#+BEGIN$_{\text{COMMENT}}$ ⋯#+END$_{\text{COMMENT}}$'包围的区域也不会被导出。最后，条目开头的 COMMENT 关键字（但在其他任何关键字或优先级标记之后）会注释掉整个子树。下面的命令有助于切换标题的注释状态。

- C-c ; (org-toggle-comment) 切换条目开头的 COMMENT 关键字。

12.6 ASCII/UTF-8 导出

ASCII 导出生成的输出文件仅包含纯 ASCII 字符。这是最简单和直接的文本输出方式，不包含任何 Org 标记。UTF-8 导出使用了该编码标准中可用的附加字符和符号。

- C-c C-e t a
- C-c C-e t u (org-ascii-export-to-ascii)
  以.txt 扩展名导出为 ASCII 文件。对于 myfile.org，Org 会导出为 myfile.txt，并会覆盖原文件而不发出警告。对于 myfile.txt，Org 会导出为 myfile.txt.txt 以防数据丢失。

12.7 HTML 导出

Org 模式包含一个 HTML 导出器，具有与 XHTML 1.0 严格标准兼容的广泛 HTML 格式化功能。

- C-c C-e h h (org-html-export-to-html) 以.html 扩展名导出为 HTML 文件。对于 myfile.org，Org 会导出为 myfile.html，并会覆盖原文件而不发出警告。C-c C-e h o 将导出为 HTML 并在网页浏览器中打开。

HTML 导出后端将 < 和 > 转换为 < 和 >。要在 Org 文件中包含原始 HTML 代码，以便 HTML 导出后端可以将这些 HTML 代码插入到输出中，可以使用这种内联语法：”。例如：

```
@@html:<b>@@bold text@@html:</b>@@
```

对于较大的原始 HTML 代码块，可以使用以下 HTML 导出代码块：

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

#+BEGIN_EXPORT html
  All lines between these markers are exported literally
#+END_EXPORT
```

12.8 LaTeX 导出

LaTeX 导出后端可以处理复杂的文档，结合标准或自定义的 LaTeX 文档类，使用不同的 LaTeX 引擎生成文档，并生成完全链接的 PDF 文件，包括索引、参考文献和目录，适用于互动在线查看或高质量印刷出版。默认情况下，LaTeX 输出使用 article 类。您可以通过在文件中添加类似 #+LATEX$_{\text{CLASS}}$: myclass 的选项来更改此设置。类名必须列在 org-latex-classes 中。

- C-c C-e l l (org-latex-export-to-latex)
  导出为.tex 扩展名的 LaTeX 文件。对于 myfile.org，Org 会导出为 myfile.tex，并会覆盖原文件而不发出警告。

- C-c C-e l p (org-latex-export-to-pdf)
  导出为 LaTeX 文件并转换为 PDF 文件。

- C-c C-e l o (< 没有对应的命名命令 >)
  导出为 LaTeX 文件并转换为 PDF，然后使用默认查看器打开 PDF。

LaTeX 导出后端可以插入任何任意的 LaTeX 代码，参见嵌入 LaTeX。将此代码嵌入 Org 文件有三种方式，它们使用不同的引号语法。插入内联代码，使用 @ 符号引用：

```
Code embedded in-line @@latex:any arbitrary LaTeX code@@ in a paragraph.
```

作为一个或多个关键字行插入到 Org 文件中：

```
#+LATEX: any arbitrary LaTeX code
```

作为导出块插入到 Org 文件中，其中后端会导出开始和结束标记之间的所有代码：

```
#+BEGIN_EXPORT latex
any arbitrary LaTeX code
#+END_EXPORT
```

12.9 iCalendar 导出

Org 模式的一个重要互操作性优势是它能够轻松地导出到或从外部应用程序导入。iCalendar 导出后端从 Org 文件中提取日历数据并导出为标准 iCalendar 格式。

- C-c C-e c f (org-icalendar-export-to-ics)
  从当前 Org 缓冲区创建 iCalendar 条目，并将其存储在相同目录中，文件扩展名为.ics。

- C-c C-e c c (org-icalendar-combine-agenda-files)
  从 org-agenda-files 中的 Org 文件创建一个合并的 iCalendar 文件，并将其写入 org-icalendar-combined-agenda-file 文件名中。

13 发布

Org 包含一个发布管理系统，允许您配置项目的自动 HTML 转换，这些项目由互联的 Org 文件组成。您还可以配置 Org 以自动将导出的 HTML 页面和相关附件（如图像和源代码文件）上传到 Web 服务器。您还可以使用 Org 将文件转换为 PDF，甚至可以将 HTML 和 PDF 转换结合起来，使文件在服务器上同时提供这两种格式。有关设置的详细说明，请参阅手册。以下是一个示例：

```
(setq org-publish-project-alist
    '(("org"
```

```
        :base-directory "~/org/"
        :publishing-function org-html-publish-to-html
        :publishing-directory "~/public_html"
        :section-numbers nil
        :with-toc nil
        :html-head "<link rel=\"stylesheet\"
                    href=\"../other/mystyle.css\"
                    type=\"text/css\"/>")))
```

- C-c C-e P x (org-publish)
  提示选择特定的项目，并发布所有属于该项目的文件。
- C-c C-e P p (org-publish-current-project)
  发布包含当前文件的项目。
- C-c C-e P f (org-publish-current-file)
  仅发布当前文件。
- C-c C-e P a (org-publish-all)
  发布所有项目。

Org 使用时间戳来跟踪文件的更改。上述功能通常只发布已更改的文件。您可以通过在任何命令前加上前缀参数来覆盖此设置，强制发布所有文件。

14 处理源代码

Org 模式提供了许多处理源代码的功能，包括在其原生主要模式下编辑代码块、评估代码块、整理代码块以及以多种格式导出代码块及其结果。源代码块的结构如下：

```
#+NAME: <name>
#+BEGIN_SRC <language> <switches> <header arguments>
  <body>
#+END_SRC
```

其中：

- <name> 是用于唯一标识代码块的字符串，
- <language> 指定代码块的语言，例如 emacs-lisp、shell、R、python 等，
- <switches> 可用于控制代码块的导出，
- <header arguments> 可用于控制代码块行为的许多方面，如下所示，
- <body> 包含实际的源代码。

使用 C-c ' 来编辑当前的代码块。这将打开一个新的主要模式编辑缓冲区，包含源代码块的主体，准备进行任何编辑。再次使用 C-c ' 来关闭缓冲区并返回到 Org 缓冲区。

14.1 使用头部参数

头部参数以初始的冒号开始，后跟参数名称（小写字母）。头部参数可以通过多种方式设置；如果出现重叠或冲突，Org 会优先考虑本地设置。

- 系统范围的头部参数
  这些参数通过自定义 org-babel-default-header-args 变量指定，或者对于特定语言 LANG，通过 org-babel-default-header-args:LANG 进行指定。
- 属性中的头部参数
  您可以使用 header-args 属性设置头部参数（参见属性）——对于语言 LANG，使用 header-args:LANG。通过属性抽屉设置的头部参数适用于子树级别及以下。
- 代码块中的头部参数
  头部参数通常在源代码块级别设置，在 $\text{BEGIN}_{\text{SRC}}$ 行上：

```
#+NAME: factorial
#+BEGIN_SRC haskell :results silent :exports code :var n=0
```

```
  fac 0 = 1
  fac n = n * fac (n-1)
#+END_SRC
```

代码块的头部参数可以使用'HEADER' 关键字分多行指定，每行一个。

14.2 评估代码块

使用 C-c C-c 来评估当前代码块并将其结果插入到 Org 文档中。默认情况下，评估只对 emacs-lisp 代码块启用，但也支持评估多种语言的代码块。有关支持的语言的完整列表，请参见手册。以下示例显示了一个代码块及其结果。

```
#+BEGIN_SRC emacs-lisp
  (+ 1 2 3 4)
#+END_SRC

#+RESULTS:
: 10
```

以下语法用于通过 var 头部参数将参数传递给代码块。

```
:var=NAME=ASSIGN
```

其中，NAME 是在代码块主体中绑定的变量名。ASSIGN 是一个字面值，例如字符串、数字、表格引用、列表、字面示例、另一个代码块（无论是否有参数）或评估代码块的结果。

14.3 评估结果

Org 如何处理代码块执行的结果取决于多个头部参数的配合。主要的决定因素是 results 头部参数。它控制代码块结果的收集、类型、格式和处理方式。

- 收集
  如何从代码块中收集结果。您可以选择 output 或 value（默认值）。

- 类型
  预期代码块执行的结果类型。您可以选择 table、list、scalar 或 file。如果未提供，Org 会尝试猜测结果类型。

- 格式
  Org 如何处理结果。一些可能的值包括 code、drawer、html、latex、link 和 raw。

- 处理方式
  如何在适当格式化后插入结果。允许的值有 silent、replace（默认值）、append 或 prepend。

将结果输出到文件的代码块，例如：图表、图形和图示，可以接受:file FILENAME 头部参数，在这种情况下，结果会保存到指定的文件中，并在缓冲区中插入一个指向该文件的链接。

14.4 导出代码块

可以导出代码块的代码、代码块评估的结果、代码和评估结果两者，或不导出任何内容。Org 默认情况下会为大多数语言导出代码。exports 头部参数用于指定该部分 Org 文件是否导出为 HTML 或 LaTeX 格式。它可以设置为 code、results、both 或 none。

14.5 提取源代码

使用 C-c C-v t 从当前缓冲区的源代码块中提取代码，创建纯源代码文件。这称为“整理”（tangling）——这是一个来自文献化编程社区的术语。在整理代码块时，其主体会通过 org-babel-expand-src-block 展开，该函数可以展开变量和“Noweb”风格的引用。要整理代码块，它必须具有 tangle 头部参数，详细信息请参见手册。

15 杂项

15.1 补全

Org 提供了在缓冲区内的补全功能，使用 M-TAB 键，无需使用 minibuffer。输入一个或多个字母，然后调用快捷键以原位完成文本。例如，此命令将补全 TEX 符号（在 \ 后）、标题开头的 TODO 关键字，以及标题中: 后的标签。

15.2 结构模板

要快速插入空的结构块，例如 #+BEGIN$_{\text{SRC}}$ ···#+END$_{\text{SRC}}$，或将现有文本包裹在这样的块中，可以使用：

- C-c C-, (org-insert-structure-template)
  提示选择一个块结构的类型，并在光标位置插入该块。如果区域是活动的，它将被包裹在块中。

15.3 清晰视图

Org 的默认大纲使用星号和无缩进，对于短文档可能会显得过于杂乱。对于书籍般的长文档，效果则不那么显著。Org 提供了一种替代的星号和缩进方案，如下表右侧所示。它只使用一个星号，并将文本缩进与标题对齐：

```
* Top level headline                |    * Top level headline
** Second level                     |      * Second level
*** Third level                     |        * Third level
    some text                       |          some text
*** Third level                     |        * Third level
    more text                       |          more text
* Another top level headline        |    * Another top level headline
```

这种视图可以通过在显示时动态实现，使用 Org Indent 模式（M-x org-indent-mode RET），该模式在每行前添加不可见的空间。您可以通过自定

义变量 org-startup-indented 来为所有文件启用 Org Indent 模式，或使用以下方式为单个文件启用它：

```
#+STARTUP: indent
```

如果您希望缩进使用硬空格字符，以便纯文本文件与 Emacs 显示尽可能相似，Org 可以通过以下方式支持您：在每个标题下缩进（使用 TAB），隐藏前导星号，并且仅使用 1、3 等级别来为每个级别提供两个字符的缩进。要在文件中启用这种支持，请使用：

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