Org Mode マニュアル

リリース 7.5

by Carsten Dominik

with contributions by David O'Toole, Bastien Guerry, Philip Rooke, Dan Davison, Eric Schulte, and Thomas Dye

このマニュアルは，Org-mode 7.5 に対応しています.

Table of Contents

1 Introduction

1.1 Summary

Org-mode はノートを保存したり，TODO リストを管理したり，プロジェクトの計画を素早く効率良く行うプレーンテキストのシステムのための Emacs のモードです．

Org-mode は，複数のプロジェクトに関連するリストや情報を含んだ，プレーンなテキスト形式のノートをまとめることで，組織的に結びついたタスクを管理します．Org-mode は，アウトラインモードを元に実装されています．そのため，大きなファイルの内容をわかりやすく構造化された状態に保つことが可能です．文書の見出しや本文の表示と非表示を切り替えて，全体を把握しながら文書を編集するときには，ツリー形式をとると便利です．表は，ビルトインされたテーブルエディタで簡単に作ることができます．Org-mode は，TODO アイテム，デッドライン，タイムスタンプ，そしてスケジュール管理に対応しています．スケジュール管理は，タスクを動的にアジェンダへ蓄積します．アジェンダは，Emacs の calendar と diary の多くの機能を利用し，スムーズに統合しています．プレーンテキストの URL に似たリンクは，ウェブサイト，メール，ネットのメッセージ，BBDB のデータ，そして，プロジェクトに関連するどのようなファイルに対しても結びついています．印刷したりノートを共有するために，Org-mode のファイルは，構造化されたアスキー形式のファイルや HTML のファイル，または（TODO とアジェンダアイテムに限り）iCalendar 形式のファイルにエクスポートできます．リンクの張られたウェブページ一式を公開するツールとしても役立ちます．

プロジェクトを計画する環境として，Org-mode は，見出しとなるノードにメタデータを追加することで動作します．そのメタデータに基づくことで，クエリの中から特定のエントリーを抽出でき，動的な *agenda views* を生成します．

Org-mode は，Org-Babel 環境を含んでいます．この環境はあなたに次のようなことを許可します．すなわち，ファイルの中に組み込まれたソースコードのブロックを動作させること，コードの評価，文書化，そして，文芸的プログラミングを容易にすることです．

Org's automatic, context-sensitive table editor with spreadsheet capabilities can be integrated into any major mode by activating the minor Orgtbl mode. Using a translation step, it can be used to maintain tables in arbitrary file types, for example in LaTeX. The structure editing and list creation capabilities can be used outside Org with the minor Orgstruct mode.

Org keeps simple things simple. When first fired up, it should feel like a straightforward, easy to use outliner. Complexity is not imposed, but a large amount of functionality is available when you need it. Org is a toolbox and can be used in different ways and for different ends, for example:

- an outline extension with visibility cycling and structure editing
- an ASCII system and table editor for taking structured notes
- a TODO list editor
- a full agenda and planner with deadlines and work scheduling
- an environment in which to implement David Allenfls GTD system
- a simple hypertext system, with HTML and LaTeX export
- a publishing tool to create a set of interlinked webpages
- an environment for literate programming

There is a website for Org which provides links to the newest version of Org, as well as additional information, frequently asked questions (FAQ), links to tutorials, etc. This page is located at `http://orgmode.org`.

このマニュアルのバージョン 7.3 は paperback book from Network Theory Ltd. (`http://www.network-theory.co.uk/org/manual/`) で手に入ります.

1.2 Installation

Important: もしあなたが，*Emacs* に含まれた古いバージョンの *Org* を利用している，もしくは，*XEmacs* のパッケージを利用している場合には，このセクションを飛ばして直接 *Section 1.3 [Activation], page 4* に移動してください．あなたの *Emacs* に含まれている *Org*（もし存在するならば）のバージョンを見るためには，*M-x load-library RET org*を実行してから，*M-x org-version*を実行してください．

もしすでにインターネットから Org をダウンロードしているならば，‘`.zip`’か‘`.tar`’もしくは Git アーカイブかは問いませんが，以下の手順に沿ってインストールしてください．まず，配布された Org のディレクトリを解凍しそこに移動します．次に，‘`Makefile`’の最初のセクションを編集します．Emacs ライブラリの名前を記入しなければなりません．たとえば，‘`emacs`’もしくは‘`xemacs`’のような名前です．最後に，ローカルの Lisp と Info ファイルが保存されているディレクトリへのパスを記入します．もしも，あながシステムのディレクトリへのアクセス権を持っていないならば，Emacs のロードパスにサブディレクトリとして‘`lisp`’を加えることで，配布された Org のディレクトリから直接 Org を動かすことが簡単にできます．そのようにするために，‘`.emacs`’に次の行を加えてください．

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

If you plan to use code from the ‘`contrib`’ subdirectory, do a similar step for this directory:

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

Now byte-compile the Lisp files with the shell command:

```
make
```

If you are running Org from the distribution directory, this is all. If you want to install Org into the system directories, use (as administrator)

```
make install
```

Installing Info files is system dependent, because of differences in the ‘`install-info`’ program. In Debian it copies the info files into the correct directory and modifies the info directory file. In many other systems, the files need to be copied to the correct directory separately, and ‘`install-info`’ then only modifies the directory file. Check your system documentation to find out which of the following commands you need:

```
make install-info
make install-info-debian
```

Then add the following line to ‘`.emacs`’. It is needed so that Emacs can autoload functions that are located in files not immediately loaded when Org-mode starts.

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

Do not forget to activate Org as described in the following section.

1.3 Activation

To make sure files with extension '`.org`' use Org mode, add the following line to your '`.emacs`' file.

```
(add-to-list 'auto-mode-alist '("\\.org\\'" . org-mode))
```

Org mode buffers need font-lock to be turned on - this is the default in Emacs[1].

The four Org commands `org-store-link`, `org-capture`, `org-agenda`, and `org-iswitchb` should be accessible through global keys (i.e. anywhere in Emacs, not just in Org buffers). Here are suggested bindings for these keys, please modify the keys to your own liking.

```
(global-set-key "\C-cl" 'org-store-link)
(global-set-key "\C-cc" 'org-capture)
(global-set-key "\C-ca" 'org-agenda)
(global-set-key "\C-cb" 'org-iswitchb)
```

With this setup, all files with extension '`.org`' will be put into Org-mode. As an alternative, make the first line of a file look like this:

```
MY PROJECTS    -*- mode: org; -*-
```

which will select Org-mode for this buffer no matter what the file's name is. See also the variable `org-insert-mode-line-in-empty-file`.

Many commands in Org work on the region if the region is *active*. To make use of this, you need to have `transient-mark-mode` (`zmacs-regions` in XEmacs) turned on. In Emacs 23 this is the default, in Emacs 22 you need to do this yourself with

```
(transient-mark-mode 1)
```

If you do not like `transient-mark-mode`, you can create an active region by using the mouse to select a region, or pressing *C-SPC* twice before moving the cursor.

1.4 Feedback

If you find problems with Org, or if you have questions, remarks, or ideas about it, please mail to the Org mailing list `emacs-orgmode@gnu.org`. If you are not a member of the mailing list, your mail will be passed to the list after a moderator has approved it[2].

For bug reports, please first try to reproduce the bug with the latest version of Org available—if you are running an outdated version, it is quite possible that the bug has been fixed already. If the bug persists, prepare a report and provide as much information as possible, including the version information of Emacs (*M-x emacs-version RET*) and Org (*M-x org-version RET*), as well as the Org related setup in '`.emacs`'. The easiest way to do this is to use the command

```
M-x org-submit-bug-report
```

which will put all this information into an Emacs mail buffer so that you only need to add your description. If you re not sending the Email from within Emacs, please copy and paste the content into your Email program.

---

[1] If you don't use font-lock globally, turn it on in Org buffer with `(add-hook 'org-mode-hook 'turn-on-font-lock)`

[2] Please consider subscribing to the mailing list, in order to minimize the work the mailing list moderators have to do.

If an error occurs, a backtrace can be very useful (see below on how to create one). Often a small example file helps, along with clear information about:

1. What exactly did you do?
2. What did you expect to happen?
3. What happened instead?

Thank you for helping to improve this program.

How to create a useful backtrace

If working with Org produces an error with a message you don't understand, you may have hit a bug. The best way to report this is by providing, in addition to what was mentioned above, a *backtrace*. This is information from the built-in debugger about where and how the error occurred. Here is how to produce a useful backtrace:

1. Reload uncompiled versions of all Org-mode Lisp files. The backtrace contains much more information if it is produced with uncompiled code. To do this, use

   ```
   C-u M-x org-reload RET
   ```

   or select `Org -> Refresh/Reload -> Reload Org uncompiled` from the menu.
2. Go to the `Options` menu and select `Enter Debugger on Error` (XEmacs has this option in the `Troubleshooting` sub-menu).
3. Do whatever you have to do to hit the error. Don't forget to document the steps you take.
4. When you hit the error, a '`*Backtrace*`' buffer will appear on the screen. Save this buffer to a file (for example using *`C-x C-w`*) and attach it to your bug report.

1.5 Typesetting conventions used in this manual

Org uses three types of keywords: TODO keywords, tags, and property names. In this manual we use the following conventions:

`TODO`
`WAITING` TODO keywords are written with all capitals, even if they are user-defined.

`boss`
`ARCHIVE` User-defined tags are written in lowercase; built-in tags with special meaning are written with all capitals.

`Release`
`PRIORITY` User-defined properties are capitalized; built-in properties with special meaning are written with all capitals.

The manual lists both the keys and the corresponding commands for accessing functionality. Org mode often uses the same key for different functions, depending on context. The command that is bound to such keys has a generic name, like `org-metaright`. In the manual we will, wherever possible, give the function that is internally called by the generic command. For example, in the chapter on document structure, *`M-right`* will be listed to call `org-do-demote`, while in the chapter on tables, it will be listed to call org-table-move-column-right.

If you prefer, you can compile the manual without the command names by unsetting the flag `cmdnames` in '`org.texi`'.

2 ドキュメントの構造

Org is based on Outline mode and provides flexible commands to edit the structure of the document.

2.1 Outlines

Org is implemented on top of Outline mode. Outlines allow a document to be organized in a hierarchical structure, which (at least for me) is the best representation of notes and thoughts. An overview of this structure is achieved by folding (hiding) large parts of the document to show only the general document structure and the parts currently being worked on. Org greatly simplifies the use of outlines by compressing the entire show/hide functionality into a single command, `org-cycle`, which is bound to the `TAB` key.

2.2 Headlines

Headlines define the structure of an outline tree. The headlines in Org start with one or more stars, on the left margin[1]. For example:

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

* Another top level headline
```

Some people find the many stars too noisy and would prefer an outline that has whitespace followed by a single star as headline starters. Section 15.8 [Clean view], page 180, describes a setup to realize this.

An empty line after the end of a subtree is considered part of it and will be hidden when the subtree is folded. However, if you leave at least two empty lines, one empty line will remain visible after folding the subtree, in order to structure the collapsed view. See the variable `org-cycle-separator-lines` to modify this behavior.

2.3 Visibility cycling

Outlines make it possible to hide parts of the text in the buffer. Org uses just two commands, bound to `TAB` and *`S-TAB`* to change the visibility in the buffer.

`TAB` `org-cycle`

Subtree cycling: Rotate current subtree among the states

```
,-> FOLDED -> CHILDREN -> SUBTREE --.
'-----------------------------------'
```

The cursor must be on a headline for this to work[2]. When the cursor is at the beginning of the buffer and the first line is not a headline, then `TAB` actually

[1] See the variables `org-special-ctrl-a/e`, `org-special-ctrl-k`, and `org-ctrl-k-protect-subtree` to configure special behavior of *`C-a`*, *`C-e`*, and *`C-k`* in headlines.

[2] see, however, the option `org-cycle-emulate-tab`.

runs global cycling (see below)[3]. Also when called with a prefix argument (*C-u TAB*), global cycling is invoked.

S-TAB `org-global-cycle`

C-u TAB *Global cycling*: Rotate the entire buffer among the states

```
,-> OVERVIEW -> CONTENTS -> SHOW ALL --.
'--------------------------------------'
```

When *S-TAB* is called with a numeric prefix argument N, the CONTENTS view up to headlines of level N will be shown. Note that inside tables, *S-TAB* jumps to the previous field.

C-u C-u C-u TAB `show-all`

Show all, including drawers.

C-c C-r `org-reveal`

Reveal context around point, showing the current entry, the following heading and the hierarchy above. Useful for working near a location that has been exposed by a sparse tree command (see Section 2.6 [Sparse trees], page 11) or an agenda command (see Section 10.5 [Agenda commands], page 100). With a prefix argument show, on each level, all sibling headings. With double prefix arg, also show the entire subtree of the parent.

C-c C-k `show-branches`

Expose all the headings of the subtree, CONTENT view for just one subtree.

C-c C-x b `org-tree-to-indirect-buffer`

Show the current subtree in an indirect buffer[4]. With a numeric prefix argument N, go up to level N and then take that tree. If N is negative then go up that many levels. With a *C-u* prefix, do not remove the previously used indirect buffer.

When Emacs first visits an Org file, the global state is set to OVERVIEW, i.e. only the top level headlines are visible. This can be configured through the variable `org-startup-folded`, or on a per-file basis by adding one of the following lines anywhere in the buffer:

```
#+STARTUP: overview
#+STARTUP: content
#+STARTUP: showall
#+STARTUP: showeverything
```

Furthermore, any entries with a '`VISIBILITY`' property (see Chapter 7 [Properties and Columns], page 57) will get their visibility adapted accordingly. Allowed values for this property are `folded`, `children`, `content`, and `all`.

C-u C-u TAB `org-set-startup-visibility`

Switch back to the startup visibility of the buffer, i.e. whatever is requested by startup options and '`VISIBILITY`' properties in individual entries.

---

[3] see the option `org-cycle-global-at-bob`.

[4] The indirect buffer (see the Emacs manual for more information about indirect buffers) will contain the entire buffer, but will be narrowed to the current tree. Editing the indirect buffer will also change the original buffer, but without affecting visibility in that buffer.

2.4 Motion

The following commands jump to other headlines in the buffer.

C-c C-n `outline-next-visible-heading`
: Next heading.

C-c C-p `outline-previous-visible-heading`
: Previous heading.

C-c C-f `org-forward-same-level`
: Next heading same level.

C-c C-b `org-backward-same-level`
: Previous heading same level.

C-c C-u `outline-up-heading`
: Backward to higher level heading.

C-c C-j `org-goto`
: Jump to a different place without changing the current outline visibility. Shows the document structure in a temporary buffer, where you can use the following keys to find your destination:

`TAB`	Cycle visibility.
`down / up`	Next/previous visible headline.
`RET`	Select this location.
`/`	Do a Sparse-tree search

The following keys work if you turn off `org-goto-auto-isearch`

`n / p`	Next/previous visible headline.
`f / b`	Next/previous headline same level.
`u`	One level up.
`0-9`	Digit argument.
`q`	Quit

See also the variable `org-goto-interface`.

2.5 Structure editing

M-RET `org-insert-heading`
: Insert new heading with same level as current. If the cursor is in a plain list item, a new item is created (see Section 2.7 [Plain lists], page 12). To force creation of a new headline, use a prefix argument. When this command is used in the middle of a line, the line is split and the rest of the line becomes the new headline[5]. If the command is used at the beginning of a headline, the new headline is created before the current line. If at the beginning of any other line, the content of that line is made the new heading. If the command is used at the end of a folded subtree (i.e. behind the ellipses at the end of a headline), then a headline like the current one will be inserted after the end of the subtree.

[5] If you do not want the line to be split, customize the variable `org-M-RET-may-split-line`.

C-RET `org-insert-heading-respect-content`
Just like *M-RET*, except when adding a new heading below the current heading, the new heading is placed after the body instead of before it. This command works from anywhere in the entry.

M-S-RET `org-insert-todo-heading`
Insert new TODO entry with same level as current heading. See also the variable `org-treat-insert-todo-heading-as-state-change`.

C-S-RET `org-insert-todo-heading-respect-content`
Insert new TODO entry with same level as current heading. Like *C-RET*, the new headline will be inserted after the current subtree.

TAB `org-cycle`
In a new entry with no text yet, the first `TAB` demotes the entry to become a child of the previous one. The next `TAB` makes it a parent, and so on, all the way to top level. Yet another `TAB`, and you are back to the initial level.

M-left `org-do-promote`
Promote current heading by one level.

M-right `org-do-demote`
Demote current heading by one level.

M-S-left `org-promote-subtree`
Promote the current subtree by one level.

M-S-right `org-demote-subtree`
Demote the current subtree by one level.

M-S-up `org-move-subtree-up`
Move subtree up (swap with previous subtree of same level).

M-S-down `org-move-subtree-down`
Move subtree down (swap with next subtree of same level).

C-c C-x C-w `org-cut-subtree`
Kill subtree, i.e. remove it from buffer but save in kill ring. With a numeric prefix argument N, kill N sequential subtrees.

C-c C-x M-w `org-copy-subtree`
Copy subtree to kill ring. With a numeric prefix argument N, copy the N sequential subtrees.

C-c C-x C-y `org-paste-subtree`
Yank subtree from kill ring. This does modify the level of the subtree to make sure the tree fits in nicely at the yank position. The yank level can also be specified with a numeric prefix argument, or by yanking after a headline marker like '`****`'.

C-y `org-yank`
Depending on the variables `org-yank-adjusted-subtrees` and `org-yank-folded-subtrees`, Org's internal `yank` command will paste subtrees folded and in a clever way, using the same command as *C-c C-x C-y*. With the

default settings, no level adjustment will take place, but the yanked tree will be folded unless doing so would swallow text previously visible. Any prefix argument to this command will force a normal `yank` to be executed, with the prefix passed along. A good way to force a normal yank is *C-u C-y*. If you use `yank-pop` after a yank, it will yank previous kill items plainly, without adjustment and folding.

C-c C-x c `org-clone-subtree-with-time-shift`

Clone a subtree by making a number of sibling copies of it. You will be prompted for the number of copies to make, and you can also specify if any timestamps in the entry should be shifted. This can be useful, for example, to create a number of tasks related to a series of lectures to prepare. For more details, see the docstring of the command `org-clone-subtree-with-time-shift`.

C-c C-w `org-refile`

Refile entry or region to a different location. See Section 9.5 [Refiling notes], page 86.

C-c ^ `org-sort-entries-or-items`

Sort same-level entries. When there is an active region, all entries in the region will be sorted. Otherwise the children of the current headline are sorted. The command prompts for the sorting method, which can be alphabetically, numerically, by time (first timestamp with active preferred, creation time, scheduled time, deadline time), by priority, by TODO keyword (in the sequence the keywords have been defined in the setup) or by the value of a property. Reverse sorting is possible as well. You can also supply your own function to extract the sorting key. With a *C-u* prefix, sorting will be case-sensitive. With two *C-u C-u* prefixes, duplicate entries will also be removed.

C-x n s `org-narrow-to-subtree`

Narrow buffer to current subtree.

C-x n b `org-narrow-to-block`

Narrow buffer to current block.

C-x n w `widen`

Widen buffer to remove narrowing.

*C-c ** `org-toggle-heading`

Turn a normal line or plain list item into a headline (so that it becomes a subheading at its location). Also turn a headline into a normal line by removing the stars. If there is an active region, turn all lines in the region into headlines. If the first line in the region was an item, turn only the item lines into headlines. Finally, if the first line is a headline, remove the stars from all headlines in the region.

When there is an active region (Transient Mark mode), promotion and demotion work on all headlines in the region. To select a region of headlines, it is best to place both point and mark at the beginning of a line, mark at the beginning of the first headline, and point at the line just after the last headline to change. Note that when the cursor is inside a table (see Chapter 3 [Tables], page 18), the Meta-Cursor keys have different functionality.

2.6 Sparse trees

An important feature of Org-mode is the ability to construct *sparse trees* for selected information in an outline tree, so that the entire document is folded as much as possible, but the selected information is made visible along with the headline structure above it[6]. Just try it out and you will see immediately how it works.

Org-mode contains several commands creating such trees, all these commands can be accessed through a dispatcher:

`C-c /` `org-sparse-tree`
: This prompts for an extra key to select a sparse-tree creating command.

`C-c / r` `org-occur`
: Occur. Prompts for a regexp and shows a sparse tree with all matches. If the match is in a headline, the headline is made visible. If the match is in the body of an entry, headline and body are made visible. In order to provide minimal context, also the full hierarchy of headlines above the match is shown, as well as the headline following the match. Each match is also highlighted; the highlights disappear when the buffer is changed by an editing command[7], or by pressing `C-c C-c`. When called with a `C-u` prefix argument, previous highlights are kept, so several calls to this command can be stacked.

`M-g n` or `M-g M-n` `next-error`
: Jump to the next sparse tree match in this buffer.

`M-g p` or `M-g M-p` `previous-error`
: Jump to the previous sparse tree match in this buffer.

For frequently used sparse trees of specific search strings, you can use the variable `org-agenda-custom-commands` to define fast keyboard access to specific sparse trees. These commands will then be accessible through the agenda dispatcher (see Section 10.2 [Agenda dispatcher], page 90). For example:

```
(setq org-agenda-custom-commands
      '(("f" occur-tree "FIXME")))
```

will define the key `C-c a f` as a shortcut for creating a sparse tree matching the string '`FIXME`'.

The other sparse tree commands select headings based on TODO keywords, tags, or properties and will be discussed later in this manual.

To print a sparse tree, you can use the Emacs command `ps-print-buffer-with-faces` which does not print invisible parts of the document[8]. Or you can use the command `C-c C-e v` to export only the visible part of the document and print the resulting file.

---

[6] See also the variables `org-show-hierarchy-above`, `org-show-following-heading`, `org-show-siblings`, and `org-show-entry-below` for detailed control on how much context is shown around each match.

[7] This depends on the option `org-remove-highlights-with-change`

[8] This does not work under XEmacs, because XEmacs uses selective display for outlining, not text properties.

2.7 Plain lists

Within an entry of the outline tree, hand-formatted lists can provide additional structure. They also provide a way to create lists of checkboxes (see Section 5.6 [Checkboxes], page 50). Org supports editing such lists, and every exporter (see Chapter 12 [Exporting], page 124) can parse and format them.

Org knows ordered lists, unordered lists, and description lists.

- *Unordered* list items start with '`-`', '`+`', or '`*`'[9] as bullets.
- *Ordered* list items start with a numeral followed by either a period or a right parenthesis[10], such as '`1.`' or '`1)`'[11]. If you want a list to start with a different value (e.g. 20), start the text of the item with `@[20]`[12]. Those constructs can be used in any item of the list in order to enforce a particular numbering.
- *Description* list items are unordered list items, and contain the separator ' `::` ' to distinguish the description *term* from the description.

Items belonging to the same list must have the same indentation on the first line. In particular, if an ordered list reaches number '`10.`', then the 2–digit numbers must be written left-aligned with the other numbers in the list. An item ends before the next line that is less or equally indented than its bullet/number.

Two methods[13] are provided to terminate lists. A list ends whenever every item has ended, which means before any line less or equally indented than items at top level. It also ends before two blank lines[14]. In that case, all items are closed. For finer control, you can end lists with any pattern set in `org-list-end-regexp`. Here is an example:

---

[9] When using '`*`' as a bullet, lines must be indented or they will be seen as top-level headlines. Also, when you are hiding leading stars to get a clean outline view, plain list items starting with a star may be hard to distinguish from true headlines. In short: even though '`*`' is supported, it may be better to not use it for plain list items.

[10] You can filter out any of them by configuring `org-plain-list-ordered-item-terminator`.

[11] You can also get '`a.`', '`A.`', '`a)`' and '`A)`' by configuring `org-alphabetical-lists`. To minimize confusion with normal text, those are limited to one character only. Beyond that limit, bullets will automatically fallback to numbers.

[12] If there's a checkbox in the item, the cookie must be put *before* the checkbox. If you have activated alphabetical lists, you can also use counters like `[@b]`.

[13] To disable either of them, configure `org-list-ending-method`.

[14] See also `org-empty-line-terminates-plain-lists`.

```
** 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.
   3. Peter Jackson being shot by Legolas
      - on DVD only
      He makes a really funny face when it happens.
   But in the end, no individual scenes matter but the film as a whole.
   Important actors in this film are:
   - Elijah Wood :: He plays Frodo
   - Sean Austin :: He plays Sam, Frodo's friend.  I still remember
     him very well from his role as Mikey Walsh in The Goonies.
```

Org supports these lists by tuning filling and wrapping commands to deal with them correctly[15], and by exporting them properly (see Chapter 12 [Exporting], page 124). Since indentation is what governs the structure of these lists, many structural constructs like `#+BEGIN_...` blocks can be indented to signal that they belong to a particular item.

If you find that using a different bullet for a sub-list (than that used for the current list-level) improves readability, customize the variable `org-list-demote-modify-bullet`.

The following commands act on items when the cursor is in the first line of an item (the line with the bullet or number). Some of them imply the application of automatic rules to keep list structure intact. If some of these actions get in your way, configure `org-list-automatic-rules` to disable them individually.

TAB `org-cycle`

Items can be folded just like headline levels. Normally this works only if the cursor is on a plain list item. For more details, see the variable `org-cycle-include-plain-lists`. If this variable is set to `integrate`, plain list items will be treated like low-level headlines. The level of an item is then given by the indentation of the bullet/number. Items are always subordinate to real headlines, however; the hierarchies remain completely separated.

M-RET `org-insert-heading`

Insert new item at current level. With a prefix argument, force a new heading (see Section 2.5 [Structure editing], page 8). If this command is used in the middle of an item, that item is *split* in two, and the second part becomes the new item[16]. If this command is executed *before item's body*, the new item is created *before* the current one.

M-S-RET Insert a new item with a checkbox (see Section 5.6 [Checkboxes], page 50).

TAB `org-cycle`

In a new item with no text yet, the first `TAB` demotes the item to become a child of the previous one. Subsequent `TAB`s move the item to meaningful levels in the list and eventually get it back to its initial position.

[15] Org only changes the filling settings for Emacs. For XEmacs, you should use Kyle E. Jones' '`filladapt.el`'. To turn this on, put into '`.emacs`': `(require 'filladapt)`

[16] If you do not want the item to be split, customize the variable `org-M-RET-may-split-line`.

S-up
S-down Jump to the previous/next item in the current list, but only if `org-support-shift-select` is off. If not, you can still use paragraph jumping commands like *`C-up`* and *`C-down`* to quite similar effect.

M-S-up
M-S-down Move the item including subitems up/down (swap with previous/next item of same indentation). If the list is ordered, renumbering is automatic.

M-left
M-right Decrease/increase the indentation of an item, leaving children alone.

M-S-left
M-S-right
Decrease/increase the indentation of the item, including subitems. Initially, the item tree is selected based on current indentation. When these commands are executed several times in direct succession, the initially selected region is used, even if the new indentation would imply a different hierarchy. To use the new hierarchy, break the command chain with a cursor motion or so.

As a special case, using this command on the very first item of a list will move the whole list. This behavior can be disabled by configuring `org-list-automatic-rules`. The global indentation of a list has no influence on the text *after* the list.

C-c C-c If there is a checkbox (see Section 5.6 [Checkboxes], page 50) in the item line, toggle the state of the checkbox. In any case, verify bullets and indentation consistency in the whole list.

C-c - Cycle the entire list level through the different itemize/enumerate bullets ('`-`', '`+`', '`*`', '`1.`', '`1)`') or a subset of them, depending on `org-plain-list-ordered-item-terminator`, the type of list, and its position[17]. With a numeric prefix argument N, select the Nth bullet from this list. If there is an active region when calling this, selected text will be changed into an item. With a prefix argument, all lines will be converted to list items. If the first line already was a list item, any item marker will be removed from the list. Finally, even without an active region, a normal line will be converted into a list item.

C-c * Turn a plain list item into a headline (so that it becomes a subheading at its location). See Section 2.5 [Structure editing], page 8, for a detailed explanation.

C-c C-* Turn the whole plain list into a subtree of the current heading. Checkboxes (see Section 5.6 [Checkboxes], page 50) will become TODO (resp. DONE) keywords when unchecked (resp. checked).

S-left/right
This command also cycles bullet styles when the cursor in on the bullet or anywhere in an item line, details depending on `org-support-shift-select`.

C-c ^ Sort the plain list. You will be prompted for the sorting method: numerically, alphabetically, by time, or by custom function.

[17] See `bullet` rule in `org-list-automatic-rules` for more information.

2.8 Drawers

Sometimes you want to keep information associated with an entry, but you normally don't want to see it. For this, Org-mode has *drawers*. Drawers need to be configured with the variable `org-drawers`[18]. Drawers look like this:

```
** This is a headline
   Still outside the drawer
   :DRAWERNAME:
   This is inside the drawer.
   :END:
   After the drawer.
```

Visibility cycling (see Section 2.3 [Visibility cycling], page 6) on the headline will hide and show the entry, but keep the drawer collapsed to a single line. In order to look inside the drawer, you need to move the cursor to the drawer line and press `TAB` there. Org-mode uses the `PROPERTIES` drawer for storing properties (see Chapter 7 [Properties and Columns], page 57), and you can also arrange for state change notes (see Section 5.3.2 [Tracking TODO state changes], page 46) and clock times (see Section 8.4 [Clocking work time], page 72) to be stored in a drawer `LOGBOOK`. If you want to store a quick note in the LOGBOOK drawer, in a similar way to state changes, use

`C-c C-z` Add a time-stamped note to the LOGBOOK drawer.

2.9 Blocks

Org-mode uses begin...end blocks for various purposes from including source code examples (see Section 11.3 [Literal examples], page 117) to capturing time logging information (see Section 8.4 [Clocking work time], page 72). These blocks can be folded and unfolded by pressing TAB in the begin line. You can also get all blocks folded at startup by configuring the variable `org-hide-block-startup` or on a per-file basis by using

```
#+STARTUP: hideblocks
#+STARTUP: nohideblocks
```

2.10 Footnotes

Org-mode supports the creation of footnotes. In contrast to the '`footnote.el`' package, Org-mode's footnotes are designed for work on a larger document, not only for one-off documents like emails. The basic syntax is similar to the one used by '`footnote.el`', i.e. a footnote is defined in a paragraph that is started by a footnote marker in square brackets in column 0, no indentation allowed. If you need a paragraph break inside a footnote, use the LaTeX idiom '`\par`'. The footnote reference is simply the marker in square brackets, inside text. For example:

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

Org-mode extends the number-based syntax to *named* footnotes and optional inline definition. Using plain numbers as markers (as '`footnote.el`' does) is supported for backward

[18] You can define drawers on a per-file basis with a line like `#+DRAWERS: HIDDEN PROPERTIES STATE`

compatibility, but not encouraged because of possible conflicts with LaTeX snippets (see Section 11.7 [Embedded LaTeX], page 120). Here are the valid references:

`[1]` A plain numeric footnote marker. Compatible with '`footnote.el`', but not recommended because something like '`[1]`' could easily be part of a code snippet.

`[fn:name]`
A named footnote reference, where `name` is a unique label word, or, for simplicity of automatic creation, a number.

`[fn:: This is the inline definition of this footnote]`
A LaTeX-like anonymous footnote where the definition is given directly at the reference point.

`[fn:name: a definition]`
An inline definition of a footnote, which also specifies a name for the note. Since Org allows multiple references to the same note, you can then use `[fn:name]` to create additional references.

Footnote labels can be created automatically, or you can create names yourself. This is handled by the variable `org-footnote-auto-label` and its corresponding `#+STARTUP` keywords. See the docstring of that variable for details.

The following command handles footnotes:

C-c C-x f The footnote action command.

When the cursor is on a footnote reference, jump to the definition. When it is at a definition, jump to the (first) reference.

Otherwise, create a new footnote. Depending on the variable `org-footnote-define-inline`[19], the definition will be placed right into the text as part of the reference, or separately into the location determined by the variable `org-footnote-section`.

When this command is called with a prefix argument, a menu of additional options is offered:

- `s` Sort the footnote definitions by reference sequence. During editing, Org makes no effort to sort footnote definitions into a particular sequence. If you want them sorted, use this command, which will also move entries according to `org-footnote-section`. Automatic sorting after each insertion/deletion can be configured using the variable `org-footnote-auto-adjust`.
- `r` Renumber the simple `fn:N` footnotes. Automatic renumbering after each insertion/deletion can be configured using the variable `org-footnote-auto-adjust`.
- `S` Short for first `r`, then `s` action.
- `n` Normalize the footnotes by collecting all definitions (including inline definitions) into a special section, and then numbering them in sequence. The references will then also be numbers. This is meant to be the final step before finishing a document (e.g. sending

[19] The corresponding in-buffer setting is: `#+STARTUP: fninline` or `#+STARTUP: nofninline`

off an email). The exporters do this automatically, and so could something like `message-send-hook`.

`d` Delete the footnote at point, and all definitions of and references to it.

Depending on the variable `org-footnote-auto-adjust`[20], renumbering and sorting footnotes can be automatic after each insertion or deletion.

C-c C-c If the cursor is on a footnote reference, jump to the definition. If it is a the definition, jump back to the reference. When called at a footnote location with a prefix argument, offer the same menu as *C-c C-x f*.

C-c C-o or *mouse-1/2*
Footnote labels are also links to the corresponding definition/reference, and you can use the usual commands to follow these links.

2.11 The Orgstruct minor mode

If you like the intuitive way the Org-mode structure editing and list formatting works, you might want to use these commands in other modes like Text mode or Mail mode as well. The minor mode `orgstruct-mode` makes this possible. Toggle the mode with *M-x orgstruct-mode*, or turn it on by default, for example in Message mode, with one of:

```
(add-hook 'message-mode-hook 'turn-on-orgstruct)
(add-hook 'message-mode-hook 'turn-on-orgstruct++)
```

When this mode is active and the cursor is on a line that looks to Org like a headline or the first line of a list item, most structure editing commands will work, even if the same keys normally have different functionality in the major mode you are using. If the cursor is not in one of those special lines, Orgstruct mode lurks silently in the shadows. When you use `orgstruct++-mode`, Org will also export indentation and autofill settings into that mode, and detect item context after the first line of an item.

[20] the corresponding in-buffer options are `fnadjust` and `nofnadjust`.

3 Tables

Org comes with a fast and intuitive table editor. Spreadsheet-like calculations are supported using the Emacs 'calc' package (see the Emacs Calculator manual for more information about the Emacs calculator).

3.1 The built-in table editor

Org makes it easy to format tables in plain ASCII. Any line with '|' as the first non-whitespace character is considered part of a table. '|' is also the column separator. A table might look like this:

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

A table is re-aligned automatically each time you press TAB or RET or *C-c C-c* inside the table. TAB also moves to the next field (RET to the next row) and creates new table rows at the end of the table or before horizontal lines. The indentation of the table is set by the first line. Any line starting with '|-' is considered as a horizontal separator line and will be expanded on the next re-align to span the whole table width. So, to create the above table, you would only type

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

and then press TAB to align the table and start filling in fields. Even faster would be to type |Name|Phone|Age followed by *C-c RET*.

When typing text into a field, Org treats DEL, Backspace, and all character keys in a special way, so that inserting and deleting avoids shifting other fields. Also, when typing *immediately after the cursor was moved into a new field with* TAB, *S-TAB or* RET, the field is automatically made blank. If this behavior is too unpredictable for you, configure the variables org-enable-table-editor and org-table-auto-blank-field.

Creation and conversion

C-c | org-table-create-or-convert-from-region

Convert the active region to table. If every line contains at least one TAB character, the function assumes that the material is tab separated. If every line contains a comma, comma-separated values (CSV) are assumed. If not, lines are split at whitespace into fields. You can use a prefix argument to force a specific separator: *C-u* forces CSV, *C-u C-u* forces TAB, and a numeric argument N indicates that at least N consecutive spaces, or alternatively a TAB will be the separator.

If there is no active region, this command creates an empty Org table. But it's easier just to start typing, like *|Name|Phone|Age RET |- TAB*.

Re-aligning and field motion

C-c C-c org-table-align

Re-align the table without moving the cursor.

<TAB> `org-table-next-field`
Re-align the table, move to the next field. Creates a new row if necessary.

S-TAB `org-table-previous-field`
Re-align, move to previous field.

RET `org-table-next-row`
Re-align the table and move down to next row. Creates a new row if necessary. At the beginning or end of a line, `RET` still does NEWLINE, so it can be used to split a table.

M-a `org-table-beginning-of-field`
Move to beginning of the current table field, or on to the previous field.

M-e `org-table-end-of-field`
Move to end of the current table field, or on to the next field.

Column and row editing

M-left `org-table-move-column-left`
M-right `org-table-move-column-right`
Move the current column left/right.

M-S-left `org-table-delete-column`
Kill the current column.

M-S-right `org-table-insert-column`
Insert a new column to the left of the cursor position.

M-up `org-table-move-row-up`
M-down `org-table-move-row-down`
Move the current row up/down.

M-S-up `org-table-kill-row`
Kill the current row or horizontal line.

M-S-down `org-table-insert-row`
Insert a new row above the current row. With a prefix argument, the line is created below the current one.

C-c - `org-table-insert-hline`
Insert a horizontal line below current row. With a prefix argument, the line is created above the current line.

C-c RET `org-table-hline-and-move`
Insert a horizontal line below current row, and move the cursor into the row below that line.

C-c ^ `org-table-sort-lines`
Sort the table lines in the region. The position of point indicates the column to be used for sorting, and the range of lines is the range between the nearest horizontal separator lines, or the entire table. If point is before the first column, you will be prompted for the sorting column. If there is an active region, the mark specifies the first line and the sorting column, while point should be in the last line to be included into the sorting. The command prompts for the

sorting type (alphabetically, numerically, or by time). When called with a prefix argument, alphabetic sorting will be case-sensitive.

Regions

C-c C-x M-w `org-table-copy-region`
Copy a rectangular region from a table to a special clipboard. Point and mark determine edge fields of the rectangle. If there is no active region, copy just the current field. The process ignores horizontal separator lines.

C-c C-x C-w `org-table-cut-region`
Copy a rectangular region from a table to a special clipboard, and blank all fields in the rectangle. So this is the "cut" operation.

C-c C-x C-y `org-table-paste-rectangle`
Paste a rectangular region into a table. The upper left corner ends up in the current field. All involved fields will be overwritten. If the rectangle does not fit into the present table, the table is enlarged as needed. The process ignores horizontal separator lines.

M-RET `org-table-wrap-region`
Split the current field at the cursor position and move the rest to the line below. If there is an active region, and both point and mark are in the same column, the text in the column is wrapped to minimum width for the given number of lines. A numeric prefix argument may be used to change the number of desired lines. If there is no region, but you specify a prefix argument, the current field is made blank, and the content is appended to the field above.

Calculations

C-c + `org-table-sum`
Sum the numbers in the current column, or in the rectangle defined by the active region. The result is shown in the echo area and can be inserted with *C-y*.

S-RET `org-table-copy-down`
When current field is empty, copy from first non-empty field above. When not empty, copy current field down to next row and move cursor along with it. Depending on the variable `org-table-copy-increment`, integer field values will be incremented during copy. Integers that are too large will not be incremented. Also, a `0` prefix argument temporarily disables the increment. This key is also used by shift-selection and related modes (see Section 15.10.2 [Conflicts], page 184).

Miscellaneous

C-c ‘ `org-table-edit-field`
Edit the current field in a separate window. This is useful for fields that are not fully visible (see Section 3.2 [Column width and alignment], page 21). When called with a *C-u* prefix, just make the full field visible, so that it can be edited in place.

M-x org-table-import
Import a file as a table. The table should be TAB or whitespace separated. Use, for example, to import a spreadsheet table or data from a database, because

these programs generally can write TAB-separated text files. This command works by inserting the file into the buffer and then converting the region to a table. Any prefix argument is passed on to the converter, which uses it to determine the separator.

`C-c |` `org-table-create-or-convert-from-region`
Tables can also be imported by pasting tabular text into the Org buffer, selecting the pasted text with `C-x C-x` and then using the `C-c |` command (see above under *Creation and conversion*).

`M-x org-table-export`
Export the table, by default as a TAB-separated file. Use for data exchange with, for example, spreadsheet or database programs. The format used to export the file can be configured in the variable `org-table-export-default-format`. You may also use properties `TABLE_EXPORT_FILE` and `TABLE_EXPORT_FORMAT` to specify the file name and the format for table export in a subtree. Org supports quite general formats for exported tables. The exporter format is the same as the format used by Orgtbl radio tables, see Section A.5.3 [Translator functions], page 191, for a detailed description.

If you don't like the automatic table editor because it gets in your way on lines which you would like to start with '`|`', you can turn it off with

```
(setq org-enable-table-editor nil)
```

Then the only table command that still works is `C-c C-c` to do a manual re-align.

3.2 Column width and alignment

The width of columns is automatically determined by the table editor. And also the alignment of a column is determined automatically from the fraction of number-like versus non-number fields in the column.

Sometimes a single field or a few fields need to carry more text, leading to inconveniently wide columns. Or maybe you want to make a table with several columns having a fixed width, regardless of content. To set[1] the width of a column, one field anywhere in the column may contain just the string '`<N>`' where '`N`' is an integer specifying the width of the column in characters. The next re-align will then set the width of this column to this value.

```
|---+------------------------------|               |---+--------|
|   |                              |               |   | <6>    |
| 1 | one                          |               | 1 | one    |
| 2 | two                          |     ----\     | 2 | two    |
| 3 | This is a long chunk of text |     ----/     | 3 | This=> |
| 4 | four                         |               | 4 | four   |
|---+------------------------------|               |---+--------|
```

Fields that are wider become clipped and end in the string '`=>`'. Note that the full text is still in the buffer but is hidden. To see the full text, hold the mouse over the field—a tool-tip window will show the full content. To edit such a field, use the command `C-c '`

[1] This feature does not work on XEmacs.

(that is `C-c` followed by the backquote). This will open a new window with the full field. Edit it and finish with `C-c C-c`.

When visiting a file containing a table with narrowed columns, the necessary character hiding has not yet happened, and the table needs to be aligned before it looks nice. Setting the option `org-startup-align-all-tables` will realign all tables in a file upon visiting, but also slow down startup. You can also set this option on a per-file basis with:

```
#+STARTUP: align
#+STARTUP: noalign
```

If you would like to overrule the automatic alignment of number-rich columns to the right and of string-rich column to the left, you can use '`<r>`', '`c`'[2] or '`<l>`' in a similar fashion. You may also combine alignment and field width like this: '`<l10>`'.

Lines which only contain these formatting cookies will be removed automatically when exporting the document.

3.3 Column groups

When Org exports tables, it does so by default without vertical lines because that is visually more satisfying in general. Occasionally however, vertical lines can be useful to structure a table into groups of columns, much like horizontal lines can do for groups of rows. In order to specify column groups, you can use a special row where the first field contains only '`/`'. The further fields can either contain '`<`' to indicate that this column should start a group, '`>`' to indicate the end of a column, or '`<>`' to make a column a group of its own. Boundaries between column groups will upon export be marked with vertical lines. Here is an example:

```
| N | N^2 | N^3 | N^4 | sqrt(n) | sqrt[4](N) |
|---+-----+-----+-----+---------+------------|
| / |   < |     |   > |       < |          > |
| 1 |   1 |   1 |   1 |       1 |          1 |
| 2 |   4 |   8 |  16 |  1.4142 |     1.1892 |
| 3 |   9 |  27 |  81 |  1.7321 |     1.3161 |
|---+-----+-----+-----+---------+------------|
#+TBLFM: $2=$1^2::$3=$1^3::$4=$1^4::$5=sqrt($1)::$6=sqrt(sqrt(($1)))
```

It is also sufficient to just insert the column group starters after every vertical line you would like to have:

```
|  N | N^2 | N^3 | N^4 | sqrt(n) | sqrt[4](N) |
|----+-----+-----+-----+---------+------------|
| /  | <   |     |     | <       |            |
```

3.4 The Orgtbl minor mode

If you like the intuitive way the Org table editor works, you might also want to use it in other modes like Text mode or Mail mode. The minor mode Orgtbl mode makes this possible. You can always toggle the mode with `M-x orgtbl-mode`. To turn it on by default, for example in Message mode, use

[2] Centering does not work inside Emacs, but it does have an effect when exporting to HTML.

```
(add-hook 'message-mode-hook 'turn-on-orgtbl)
```

Furthermore, with some special setup, it is possible to maintain tables in arbitrary syntax with Orgtbl mode. For example, it is possible to construct LaTeX tables with the underlying ease and power of Orgtbl mode, including spreadsheet capabilities. For details, see Section A.5 [Tables in arbitrary syntax], page 188.

3.5 The spreadsheet

The table editor makes use of the Emacs '`calc`' package to implement spreadsheet-like capabilities. It can also evaluate Emacs Lisp forms to derive fields from other fields. While fully featured, Org's implementation is not identical to other spreadsheets. For example, Org knows the concept of a *column formula* that will be applied to all non-header fields in a column without having to copy the formula to each relevant field. There is also a formula debugger, and a formula editor with features for highlighting fields in the table corresponding to the references at the point in the formula, moving these references by arrow keys

3.5.1 References

To compute fields in the table from other fields, formulas must reference other fields or ranges. In Org, fields can be referenced by name, by absolute coordinates, and by relative coordinates. To find out what the coordinates of a field are, press *C-c ?* in that field, or press *C-c }* to toggle the display of a grid.

Field references

Formulas can reference the value of another field in two ways. Like in any other spreadsheet, you may reference fields with a letter/number combination like `B3`, meaning the 2nd field in the 3rd row.

Org prefers[3] to use another, more general operator that looks like this:

```
@row$column
```

and allows relative references, i.e. references relative to the row/column of the field whose value is being computed. These relative references make it possible to store a formula only once and use it in many fields without copying and modifying it.

Column references can be absolute like '`1`', '`2`',...'*N*', or relative to the current column like '`+1`' or '`-2`'. `$>` references the last column in the table, and you can use offsets like `$>-2`, meaning the third column from the right.

The row specification only counts data lines and ignores horizontal separator lines (hlines). Like with columns, you can use absolute row numbers '`1`'...'*N*', and row numbers relative to the current row like '`+3`' or '`-1`', and `@>` references the last row in the table[4]. You may also specify the row relative to one of the hlines: '`I`' refers to the first hline[5], '`II`'

[3] Org will understand references typed by the user as '`B4`', but it will not use this syntax when offering a formula for editing. You can customize this behavior using the variable `org-table-use-standard-references`.

[4] For backward compatibility you can also use special names like '`$LR5`' and '`$LR12`' to refer in a stable way to the 5th and 12th field in the last row of the table. However, this syntax is deprecated, it should not be used for new documents.

[5] Note that only hlines are counted that *separate* table lines. If the table starts with a hline above the header, it does not count.

to the second, etc. '`-I`' refers to the first such line above the current line, '`+I`' to the first such line below the current line. You can also write '`III+2`' which is the second data line after the third hline in the table.

'`0`' refers to the current row and column. Also, if you omit either the column or the row part of the reference, the current row/column is implied.

Org's references with *unsigned* numbers are fixed references in the sense that if you use the same reference in the formula for two different fields, the same field will be referenced each time. Org's references with *signed* numbers are floating references because the same reference operator can reference different fields depending on the field being calculated by the formula.

Here are a few examples:

`@2$3`	2nd row, 3rd column
`C2`	same as previous
`$5`	column 5 in the current row
`E&`	same as previous
`@2`	current column, row 2
`@-1$-3`	the field one row up, three columns to the left
`@-I$2`	field just under hline above current row, column 2

Range references

You may reference a rectangular range of fields by specifying two field references connected by two dots '`..`'. If both fields are in the current row, you may simply use '`$2..$7`', but if at least one field is in a different row, you need to use the general `@row$column` format at least for the first field (i.e the reference must start with @‘' in order to be interpreted correctly). Examples:

`$1..$3`	First three fields in the current row
`$P..$Q`	Range, using column names (see under Advanced)
`@2$1..@4$3`	6 fields between these two fields
`A2..C4`	Same as above
`@-1$-2..@-1`	3 numbers from the column to the left, 2 up to current row
`@I..II`	@@Between first and second hline, short for `I..II`

Range references return a vector of values that can be fed into Calc vector functions. Empty fields in ranges are normally suppressed, so that the vector contains only the non-empty fields (but see the '`E`' mode switch below). If there are no non-empty fields, '`[0]`' is returned to avoid syntax errors in formulas.

Field coordinates in formulas

For Calc formulas and Lisp formulas `@#` and `$#` can be used to get the row or column number of the field where the formula result goes. The traditional Lisp formula equivalents are `org-table-current-dline` and `org-table-current-column`. Examples:

`if(@# % 2, $#, string(""))`	column number on odd lines only
`$3 = remote(FOO, @@#$2)`	copy column 2 from table FOO into column 3 of the current table

For the second example, table FOO must have at least as many rows as the current table. Note that this is inefficient[6] for large number of rows.

Named references

'`$name`' is interpreted as the name of a column, parameter or constant. Constants are defined globally through the variable `org-table-formula-constants`, and locally (for the file) through a line like

```
#+CONSTANTS: c=299792458. pi=3.14 eps=2.4e-6
```

Also properties (see Chapter 7 [Properties and Columns], page 57) can be used as constants in table formulas: for a property '`:Xyz:`' use the name '`$PROP_Xyz`', and the property will be searched in the current outline entry and in the hierarchy above it. If you have the '`constants.el`' package, it will also be used to resolve constants, including natural constants like '`$h`' for Planck's constant, and units like '`$km`' for kilometers[7]. Column names and parameters can be specified in special table lines. These are described below, see Section 3.5.8 [Advanced features], page 30. All names must start with a letter, and further consist of letters and numbers.

Remote references

You may also reference constants, fields and ranges from a different table, either in the current file or even in a different file. The syntax is

```
remote(NAME-OR-ID,REF)
```

where NAME can be the name of a table in the current file as set by a `#+TBLNAME: NAME` line before the table. It can also be the ID of an entry, even in a different file, and the reference then refers to the first table in that entry. REF is an absolute field or range reference as described above for example `@3$3` or `$somename`, valid in the referenced table.

3.5.2 Formula syntax for Calc

A formula can be any algebraic expression understood by the Emacs '`Calc`' package. **Note that '`calc`' has the non-standard convention that '`/`' has lower precedence than '`*`', so that '`a/b*c`' is interpreted as '`a/(b*c)`'.** Before evaluation by `calc-eval` (see Section "Calling Calc from Your Lisp Programs" in *GNU Emacs Calc Manual*), variable substitution takes place according to the rules described above. The range vectors can be directly fed into the Calc vector functions like '`vmean`' and '`vsum`'.

A formula can contain an optional mode string after a semicolon. This string consists of flags to influence Calc and other modes during execution. By default, Org uses the standard Calc modes (precision 12, angular units degrees, fraction and symbolic modes off). The display format, however, has been changed to `(float 8)` to keep tables compact. The default settings can be configured using the variable `org-calc-default-modes`.

`p20`	set the internal Calc calculation precision to 20 digits
`n3 s3 e2 f4`	Normal, scientific, engineering, or fixed format of the result of Calc passed back to Org.

[6] The computation time scales as O(N^2) because table FOO is parsed for each field to be copied.

[7] '`constants.el`' can supply the values of constants in two different unit systems, `SI` and `cgs`. Which one is used depends on the value of the variable `constants-unit-system`. You can use the `#+STARTUP` options `constSI` and `constcgs` to set this value for the current buffer.

	Calc formatting is unlimited in precision as long as the Calc calculation precision is greater.
`D R`	angle modes: degrees, radians
`F S`	fraction and symbolic modes
`N`	interpret all fields as numbers, use 0 for non-numbers
`T`	force text interpretation
`E`	keep empty fields in ranges
`L`	literal

Unless you use large integer numbers or high-precision-calculation and -display for floating point numbers you may alternatively provide a `printf` format specifier to reformat the Calc result after it has been passed back to Org instead of letting Calc already do the formatting[8]. A few examples:

`$1+$2`	Sum of first and second field
`$1+$2;%.2f`	Same, format result to two decimals
`exp($2)+exp($1)`	Math functions can be used
`$0;%.1f`	Reformat current cell to 1 decimal
`($3-32)*5/9`	Degrees F -> C conversion
`$c/$1/$cm`	Hz -> cm conversion, using '`constants.el`'
`tan($1);Dp3s1`	Compute in degrees, precision 3, display SCI 1
`sin($1);Dp3%.1e`	Same, but use printf specifier for display
`vmean($2..$7)`	Compute column range mean, using vector function
`vmean($2..$7);EN`	Same, but treat empty fields as 0
`taylor($3,x=7,2)`	Taylor series of $3, at x=7, second degree

Calc also contains a complete set of logical operations. For example

`if($1<20,teen,string(""))` ``teenfflfi if age $1 less than 20, else empty

3.5.3 Emacs Lisp forms as formulas

It is also possible to write a formula in Emacs Lisp; this can be useful for string manipulation and control structures, if Calc's functionality is not enough. If a formula starts with a single-quote followed by an opening parenthesis, then it is evaluated as a Lisp form. The evaluation should return either a string or a number. Just as with '`calc`' formulas, you can specify modes and a printf format after a semicolon. With Emacs Lisp forms, you need to be conscious about the way field references are interpolated into the form. By default, a reference will be interpolated as a Lisp string (in double-quotes) containing the field. If you provide the '`N`' mode switch, all referenced elements will be numbers (non-number fields will be zero) and interpolated as Lisp numbers, without quotes. If you provide the '`L`' flag, all fields will be interpolated literally, without quotes. i.e., if you want a reference to be interpreted as a string by the Lisp form, enclose the reference operator itself in double-quotes, like `"$3"`. Ranges are inserted as space-separated fields, so you can embed them in list or vector syntax. Here are a few examples—note how the '`N`' mode is used when we do computations in Lisp:

Swap the first two characters of the content of column 1

[8] The `printf` reformatting is limited in precision because the value passed to it is converted into an `integer` or `double`. The `integer` is limited in size by truncating the signed value to 32 bits. The `double` is limited in precision to 64 bits overall which leaves approximately 16 significant decimal digits.

```
'(concat (substring $1 1 2) (substring $1 0 1) (substring $1 2))
```

Add columns 1 and 2, equivalent to Calcfls `$1+$2`

```
'(+ $1 $2);N
```

Compute the sum of columns 1-4, like Calcfls `vsum($1..$4)`

```
'(apply '+ '($1..$4));N
```

3.5.4 Field and range formulas

To assign a formula to a particular field, type it directly into the field, preceded by '`:=`', for example @'`:=vsum(II..III)`'. When you press `TAB` or `RET` or *C-c C-c* with the cursor still in the field, the formula will be stored as the formula for this field, evaluated, and the current field will be replaced with the result.

Formulas are stored in a special line starting with '`#+TBLFM:`' directly below the table. If you type the equation in the 4th field of the 3rd data line in the table, the formula will look like '`@3$4=$1+$2`'. When inserting/deleting/swapping column and rows with the appropriate commands, *absolute references* (but not relative ones) in stored formulas are modified in order to still reference the same field. Of course this is not true if you edit the table structure with normal editing commands—then you must fix the equations yourself. Instead of typing an equation into the field, you may also use the following command

C-u C-c = `org-table-eval-formula`

Install a new formula for the current field. The command prompts for a formula with default taken from the '`#+TBLFM:`' line, applies it to the current field, and stores it.

The left-hand side of a formula can also be a special expression in order to assign the formula to a number of different fields. There is no keyboard shortcut to enter such range formulas. To add them, use the formula editor (see Section 3.5.6 [Editing and debugging formulas], page 28) or edit the `#+TBLFM:` line directly.

`$2=` Column formula, valid for the entire column. This is so common that Org treats these formulas in a special way, see Section 3.5.5 [Column formulas], page 27.

`@3=` Row formula, applies to all fields in the specified row. `@L=` means the last row.

`@1$2..@4$3=`

Range formula, applies to all fields in the given rectangular range. This can also be used to assign a formula to some but not all fields in a row.

`$name=` Named field, see Section 3.5.8 [Advanced features], page 30.

3.5.5 Column formulas

When you assign a formula to a simple column reference like `$3=`, the same formula will be used in all fields of that column, with the following very convenient exceptions: (i) If the table contains horizontal separator hlines, everything before the first such line is considered part of the table *header* and will not be modified by column formulas. (ii) Fields that already get a value from a field/range formula will be left alone by column formulas. These conditions make column formulas very easy to use.

To assign a formula to a column, type it directly into any field in the column, preceded by an equal sign, like '`=$1+$2`'. When you press `TAB` or `RET` or *C-c C-c* with the cursor

still in the field, the formula will be stored as the formula for the current column, evaluated and the current field replaced with the result. If the field contains only '=', the previously stored formula for this column is used. For each column, Org will only remember the most recently used formula. In the '#+TBLFM:' line, column formulas will look like '$4=$1+$2'. The left-hand side of a column formula can not be the name of column, it must be the numeric column reference or $>.

Instead of typing an equation into the field, you may also use the following command:

C-c = `org-table-eval-formula`
: Install a new formula for the current column and replace current field with the result of the formula. The command prompts for a formula, with default taken from the '#+TBLFM' line, applies it to the current field and stores it. With a numeric prefix argument(e.g. *C-5 C-c =*) the command will apply it to that many consecutive fields in the current column.

3.5.6 Editing and debugging formulas

You can edit individual formulas in the minibuffer or directly in the field. Org can also prepare a special buffer with all active formulas of a table. When offering a formula for editing, Org converts references to the standard format (like B3 or D&) if possible. If you prefer to only work with the internal format (like @3$2 or $4), configure the variable `org-table-use-standard-references`.

C-c = or *C-u C-c =* `org-table-eval-formula`
: Edit the formula associated with the current column/field in the minibuffer. See Section 3.5.5 [Column formulas], page 27, and Section 3.5.4 [Field and range formulas], page 27.

C-u C-u C-c = `org-table-eval-formula`
: Re-insert the active formula (either a field formula, or a column formula) into the current field, so that you can edit it directly in the field. The advantage over editing in the minibuffer is that you can use the command *C-c ?*.

C-c ? `org-table-field-info`
: While editing a formula in a table field, highlight the field(s) referenced by the reference at the cursor position in the formula.

C-c }
: Toggle the display of row and column numbers for a table, using overlays (`org-table-toggle-coordinate-overlays`). These are updated each time the table is aligned; you can force it with *C-c C-c*.

C-c {
: Toggle the formula debugger on and off (`org-table-toggle-formula-debugger`). See below.

C-c ' `org-table-edit-formulas`
: Edit all formulas for the current table in a special buffer, where the formulas will be displayed one per line. If the current field has an active formula, the cursor in the formula editor will mark it. While inside the special buffer, Org will automatically highlight any field or range reference at the cursor position. You may edit, remove and add formulas, and use the following commands:

C-c C-c or *C-x C-s* `org-table-fedit-finish`
: Exit the formula editor and store the modified formulas. With *C-u* prefix, also apply the new formulas to the entire table.

C-c C-q `org-table-fedit-abort`
: Exit the formula editor without installing changes.

C-c C-r `org-table-fedit-toggle-ref-type`
: Toggle all references in the formula editor between standard (like `B3`) and internal (like `@3$2`).

TAB `org-table-fedit-lisp-indent`
: Pretty-print or indent Lisp formula at point. When in a line containing a Lisp formula, format the formula according to Emacs Lisp rules. Another `TAB` collapses the formula back again. In the open formula, `TAB` re-indents just like in Emacs Lisp mode.

M-TAB `lisp-complete-symbol`
: Complete Lisp symbols, just like in Emacs Lisp mode.

S-up/down/left/right
: Shift the reference at point. For example, if the reference is `B3` and you press *S-right*, it will become `C3`. This also works for relative references and for hline references.

M-S-up `org-table-fedit-line-up`
M-S-down `org-table-fedit-line-down`
: Move the test line for column formulas in the Org buffer up and down.

M-up `org-table-fedit-scroll-down`
M-down `org-table-fedit-scroll-up`
: Scroll the window displaying the table.

C-c }
: Turn the coordinate grid in the table on and off.

Making a table field blank does not remove the formula associated with the field, because that is stored in a different line (the '`#+TBLFM`' line)—during the next recalculation the field will be filled again. To remove a formula from a field, you have to give an empty reply when prompted for the formula, or to edit the '`#+TBLFM`' line.

You may edit the '`#+TBLFM`' directly and re-apply the changed equations with *C-c C-c* in that line or with the normal recalculation commands in the table.

Debugging formulas

When the evaluation of a formula leads to an error, the field content becomes the string '`#ERROR`'. If you would like see what is going on during variable substitution and calculation in order to find a bug, turn on formula debugging in the `Tbl` menu and repeat the calculation, for example by pressing *C-u C-u C-c = RET* in a field. Detailed information will be displayed.

3.5.7 Updating the table

Recalculation of a table is normally not automatic, but needs to be triggered by a command. See Section 3.5.8 [Advanced features], page 30, for a way to make recalculation at least semi-automatic.

In order to recalculate a line of a table or the entire table, use the following commands:

*C-c ** `org-table-recalculate`
: Recalculate the current row by first applying the stored column formulas from left to right, and all field/range formulas in the current row.

*C-u C-c **
C-u C-c C-c
: Recompute the entire table, line by line. Any lines before the first hline are left alone, assuming that these are part of the table header.

*C-u C-u C-c ** or *C-u C-u C-c C-c* `org-table-iterate`
: Iterate the table by recomputing it until no further changes occur. This may be necessary if some computed fields use the value of other fields that are computed *later* in the calculation sequence.

M-x org-table-recalculate-buffer-tables
: Recompute all tables in the current buffer.

M-x org-table-iterate-buffer-tables
: Iterate all tables in the current buffer, in order to converge table-to-table dependencies.

3.5.8 Advanced features

If you want the recalculation of fields to happen automatically, or if you want to be able to assign *names* to fields and columns, you need to reserve the first column of the table for special marking characters.

C-# `org-table-rotate-recalc-marks`
: Rotate the calculation mark in first column through the states ' ', '#', '*', '!', '$'. When there is an active region, change all marks in the region.

Here is an example of a table that collects exam results of students and makes use of these features:

```
|---+---------+--------+--------+--------+-------+------|
|   | Student | Prob 1 | Prob 2 | Prob 3 | Total | Note |
|---+---------+--------+--------+--------+-------+------|
| ! |         |     P1 |     P2 |     P3 |   Tot |      |
| # | Maximum |     10 |     15 |     25 |    50 | 10.0 |
| ^ |         |     m1 |     m2 |     m3 |    mt |      |
|---+---------+--------+--------+--------+-------+------|
| # | Peter   |     10 |      8 |     23 |    41 |  8.2 |
| # | Sam     |      2 |      4 |      3 |     9 |  1.8 |
|---+---------+--------+--------+--------+-------+------|
|   | Average |        |        |        |  29.7 |      |
| ^ |         |        |        |        |    at |      |
| $ | max=50  |        |        |        |       |      |
|---+---------+--------+--------+--------+-------+------|
#+TBLFM: $6=vsum($P1..$P3)::$7=10*$Tot/$max;%.1f::$at=vmean(@-II..@-I);%.1f
```

Important: please note that for these special tables, recalculating the table with *C-u C-c ** will only affect rows that are marked '#' or '*', and fields that have a formula assigned to the field itself. The column formulas are not applied in rows with empty first field.

The marking characters have the following meaning:

'!' The fields in this line define names for the columns, so that you may refer to a column as '$Tot' instead of '$6'.

'^' This row defines names for the fields *above* the row. With such a definition, any formula in the table may use '$m1' to refer to the value '10'. Also, if you assign a formula to a names field, it will be stored as '$name=...'.

'_' Similar to '^', but defines names for the fields in the row *below*.

'$' Fields in this row can define *parameters* for formulas. For example, if a field in a '$' row contains 'max=50', then formulas in this table can refer to the value 50 using '$max'. Parameters work exactly like constants, only that they can be defined on a per-table basis.

'#' Fields in this row are automatically recalculated when pressing TAB or RET or *S-TAB* in this row. Also, this row is selected for a global recalculation with *C-u C-c **. Unmarked lines will be left alone by this command.

'*' Selects this line for global recalculation with *C-u C-c **, but not for automatic recalculation. Use this when automatic recalculation slows down editing too much.

'' Unmarked lines are exempt from recalculation with *C-u C-c **. All lines that should be recalculated should be marked with '#' or '*'.

'/' Do not export this line. Useful for lines that contain the narrowing '<N>' markers or column group markers.

Finally, just to whet your appetite for what can be done with the fantastic 'calc.el' package, here is a table that computes the Taylor series of degree n at location x for a couple of functions.

```
|---+-------------+---+-----+--------------------------------------|
|   | Func        | n | x   | Result                               |
|---+-------------+---+-----+--------------------------------------|
| # | exp(x)      | 1 | x   | 1 + x                                |
| # | exp(x)      | 2 | x   | 1 + x + x^2 / 2                      |
| # | exp(x)      | 3 | x   | 1 + x + x^2 / 2 + x^3 / 6            |
| # | x^2+sqrt(x) | 2 | x=0 | x*(0.5 / 0) + x^2 (2 - 0.25 / 0) / 2 |
| # | x^2+sqrt(x) | 2 | x=1 | 2 + 2.5 x - 2.5 + 0.875 (x - 1)^2    |
| * | tan(x)      | 3 | x   | 0.0175 x + 1.77e-6 x^3               |
|---+-------------+---+-----+--------------------------------------|
#+TBLFM: $5=taylor($2,$4,$3);n3
```

3.6 Org-Plot

Org-Plot can produce 2D and 3D graphs of information stored in org tables using 'Gnuplot' http://www.gnuplot.info/ and 'gnuplot-mode' http://cars9.uchicago.edu/~ravel/

`software/gnuplot-mode.html`. To see this in action, ensure that you have both Gnuplot and Gnuplot mode installed on your system, then call `org-plot/gnuplot` on the following table.

```
#+PLOT: title:"Citas" ind:1 deps:(3) type:2d with:histograms set:"yrange [0:]"
| Sede      | Max cites | H-index |
|-----------+-----------+---------|
| Chile     |    257.72 |   21.39 |
| Leeds     |    165.77 |   19.68 |
| Sao Paolo |     71.00 |   11.50 |
| Stockholm |    134.19 |   14.33 |
| Morelia   |    257.56 |   17.67 |
```

Notice that Org Plot is smart enough to apply the table's headers as labels. Further control over the labels, type, content, and appearance of plots can be exercised through the `#+PLOT:` lines preceding a table. See below for a complete list of Org-plot options. For more information and examples see the Org-plot tutorial at `http://orgmode.org/worg/org-tutorials/org-plot.html`.

Plot Options

`set` Specify any `gnuplot` option to be set when graphing.

`title` Specify the title of the plot.

`ind` Specify which column of the table to use as the `x` axis.

`deps` Specify the columns to graph as a Lisp style list, surrounded by parentheses and separated by spaces for example `dep:(3 4)` to graph the third and fourth columns (defaults to graphing all other columns aside from the `ind` column).

`type` Specify whether the plot will be `2d`, `3d`, or `grid`.

`with` Specify a `with` option to be inserted for every col being plotted (e.g. `lines`, `points`, `boxes`, `impulses`, etc...). Defaults to `lines`.

`file` If you want to plot to a file, specify "*`path/to/desired/output-file`*".

`labels` List of labels to be used for the `deps` (defaults to the column headers if they exist).

`line` Specify an entire line to be inserted in the Gnuplot script.

`map` When plotting `3d` or `grid` types, set this to `t` to graph a flat mapping rather than a `3d` slope.

`timefmt` Specify format of Org-mode timestamps as they will be parsed by Gnuplot. Defaults to '`%Y-%m-%d-%H:%M:%S`'.

`script` If you want total control, you can specify a script file (place the file name between double-quotes) which will be used to plot. Before plotting, every instance of `$datafile` in the specified script will be replaced with the path to the generated data file. Note: even if you set this option, you may still want to specify the plot type, as that can impact the content of the data file.

4 Hyperlinks

HTML のように、Org-mode はファイル内でリンクしたり、他のファイルや Usenet の記事やメールなど、外部へリンクしたりすることができます。

4.1 Link format

Org will recognize plain URL-like links and activate them as clickable links. The general link format, however, looks like this:

```
[[link][description]]       or alternatively       [[link]]
```

Once a link in the buffer is complete (all brackets present), Org will change the display so that '`description`' is displayed instead of '`[[link][description]]`' and '`link`' is displayed instead of '`[[link]]`'. Links will be highlighted in the face `org-link`, which by default is an underlined face. You can directly edit the visible part of a link. Note that this can be either the '`link`' part (if there is no description) or the '`description`' part. To edit also the invisible '`link`' part, use *C-c C-l* with the cursor on the link.

If you place the cursor at the beginning or just behind the end of the displayed text and press `BACKSPACE`, you will remove the (invisible) bracket at that location. This makes the link incomplete and the internals are again displayed as plain text. Inserting the missing bracket hides the link internals again. To show the internal structure of all links, use the menu entry `Org->Hyperlinks->Literal links`.

4.2 Internal links

If the link does not look like a URL, it is considered to be internal in the current file. The most important case is a link like '`[[#my-custom-id]]`' which will link to the entry with the `CUSTOM_ID` property '`my-custom-id`'. Such custom IDs are very good for HTML export (see Section 12.5 [HTML export], page 127) where they produce pretty section links. You are responsible yourself to make sure these custom IDs are unique in a file.

Links such as '`[[My Target]]`' or '`[[My Target][Find my target]]`' lead to a text search in the current file.

The link can be followed with *C-c C-o* when the cursor is on the link, or with a mouse click (see Section 4.4 [Handling links], page 35). Links to custom IDs will point to the corresponding headline. The preferred match for a text link is a *dedicated target*: the same string in double angular brackets. Targets may be located anywhere; sometimes it is convenient to put them into a comment line. For example

```
# <<My Target>>
```

In HTML export (see Section 12.5 [HTML export], page 127), such targets will become named anchors for direct access through '`http`' links[1].

If no dedicated target exists, Org will search for a headline that is exactly the link text but may also include a TODO keyword and tags[2]. In non-Org files, the search will look for the words in the link text. In the above example the search would be for '`my target`'.

[1] Note that text before the first headline is usually not exported, so the first such target should be after the first headline, or in the line directly before the first headline.

[2] To insert a link targeting a headline, in-buffer completion can be used. Just type a star followed by a few optional letters into the buffer and press *M-TAB*. All headlines in the current buffer will be offered as completions.

Following a link pushes a mark onto Org's own mark ring. You can return to the previous position with `C-c &`. Using this command several times in direct succession goes back to positions recorded earlier.

4.2.1 Radio targets

Org can automatically turn any occurrences of certain target names in normal text into a link. So without explicitly creating a link, the text connects to the target radioing its position. Radio targets are enclosed by triple angular brackets. For example, a target '`<<<My Target>>>`' causes each occurrence of '`my target`' in normal text to become activated as a link. The Org file is scanned automatically for radio targets only when the file is first loaded into Emacs. To update the target list during editing, press `C-c C-c` with the cursor on or at a target.

4.3 External links

Org supports links to files, websites, Usenet and email messages, BBDB database entries and links to both IRC conversations and their logs. External links are URL-like locators. They start with a short identifying string followed by a colon. There can be no space after the colon. The following list shows examples for each link type.

`http://www.astro.uva.nl/~dominik`	on the web
`doi:10.1000/182`	DOI for an electronic resource
`file:/home/dominik/images/jupiter.jpg`	file, absolute path
`/home/dominik/images/jupiter.jpg`	same as above
`file:papers/last.pdf`	file, relative path
`./papers/last.pdf`	same as above
`file:/myself@some.where:papers/last.pdf`	file, path on remote machine
`/myself@some.where:papers/last.pdf`	same as above
`file:sometextfile::NNN`	file with line number to jump to
`file:projects.org`	another Org file
`file:projects.org::some words`	text search in Org file
`file:projects.org::*task title`	heading search in 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
`vm://myself@some.where.org/folder#id`	VM on remote machine
`wl:folder`	WANDERLUST folder link
`wl:folder#id`	WANDERLUST message link
`mhe:folder`	MH-E folder link
`mhe:folder#id`	MH-E message link
`rmail:folder`	RMAIL folder link
`rmail:folder#id`	RMAIL message link
`gnus:group`	Gnus group 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)
`shell:ls *.org`	A shell command
`elisp:org-agenda`	Interactive Elisp command
`elisp:(find-file-other-frame "Elisp.org")`	Elisp form to evaluate

For customizing Org to add new link types Section A.3 [Adding hyperlink types], page 186.

A link should be enclosed in double brackets and may contain a descriptive text to be displayed instead of the URL (see Section 4.1 [Link format], page 33), for example:

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

If the description is a file name or URL that points to an image, HTML export (see Section 12.5 [HTML export], page 127) will inline the image as a clickable button. If there is no description at all and the link points to an image, that image will be inlined into the exported HTML file.

Org also finds external links in the normal text and activates them as links. If spaces must be part of the link (for example in '`bbdb:Richard Stallman`'), or if you need to remove ambiguities about the end of the link, enclose them in square brackets.

4.4 Handling links

Org provides methods to create a link in the correct syntax, to insert it into an Org file, and to follow the link.

C-c l `org-store-link`

Store a link to the current location. This is a *global* command (you must create the key binding yourself) which can be used in any buffer to create a link. The link will be stored for later insertion into an Org buffer (see below). What kind of link will be created depends on the current buffer:

Org-mode buffers
For Org files, if there is a '`<<target>>`' at the cursor, the link points to the target. Otherwise it points to the current headline, which will also be the description.

If the headline has a `CUSTOM_ID` property, a link to this custom ID will be stored. In addition or alternatively (depending on the value of `org-link-to-org-use-id`), a globally unique `ID` property will be created and/or used to construct a link. So using this command in Org buffers will potentially create two links: a human-readable from the custom ID, and one that is globally unique and works even if the entry is moved from file to file. Later, when inserting the link, you need to decide which one to use.

Email/News clients: VM, Rmail, Wanderlust, MH-E, Gnus
Pretty much all Emacs mail clients are supported. The link will point to the current article, or, in some GNUS buffers, to the group. The description is constructed from the author and the subject.

Web browsers: W3 and W3M
Here the link will be the current URL, with the page title as description.

Contacts: BBDB
Links created in a BBDB buffer will point to the current entry.

Chat: IRC
For IRC links, if you set the variable `org-irc-link-to-logs` to `t`, a '`file:/`' style link to the relevant point in the logs for the current conversation is created. Otherwise an '`irc:/`' style link to the user/channel/server under the point will be stored.

Other files
For any other files, the link will point to the file, with a search string (see Section 4.7 [Search options], page 39) pointing to the contents of the current line. If there is an active region, the selected words will form the basis of the search string. If the automatically created link is not working correctly or accurately enough, you can write custom functions to select the search string and to do the search for particular file types—see Section 4.8 [Custom searches], page 39. The key binding *C-c l* is only a suggestion—see Section 1.2 [Installation], page 3.

Agenda view
When the cursor is in an agenda view, the created link points to the entry referenced by the current line.

C-c C-l `org-insert-link`

Insert a link[3]. This prompts for a link to be inserted into the buffer. You can just type a link, using text for an internal link, or one of the link type prefixes mentioned in the examples above. The link will be inserted into the buffer[4], along with a descriptive text. If some text was selected when this command is called, the selected text becomes the default description.

Inserting stored links
All links stored during the current session are part of the history for this prompt, so you can access them with `up` and `down` (or *M-p/n*).

Completion support
Completion with `TAB` will help you to insert valid link prefixes like '`http:`' or '`ftp:`', including the prefixes defined through link abbreviations (see Section 4.6 [Link abbreviations], page 38). If you press `RET` after inserting only the *prefix*, Org will offer specific completion support for some link types[5] For example, if you type *file RET*, file name completion (alternative access: *C-u C-c C-l*, see below) will be offered, and after *bbdb RET* you can complete contact names.

C-u C-c C-l

When *C-c C-l* is called with a *C-u* prefix argument, a link to a file will be inserted and you may use file name completion to select the name of the file. The path to the file is inserted relative to the directory of the current Org file, if

[3] Note that you don't have to use this command to insert a link. Links in Org are plain text, and you can type or paste them straight into the buffer. By using this command, the links are automatically enclosed in double brackets, and you will be asked for the optional descriptive text.

[4] After insertion of a stored link, the link will be removed from the list of stored links. To keep it in the list later use, use a triple *C-u* prefix argument to *C-c C-l*, or configure the option `org-keep-stored-link-after-insertion`.

[5] This works by calling a special function `org-PREFIX-complete-link`.

the linked file is in the current directory or in a sub-directory of it, or if the path is written relative to the current directory using '`../`'. Otherwise an absolute path is used, if possible with '`~/`' for your home directory. You can force an absolute path with two *C-u* prefixes.

C-c C-l (with cursor on existing link)
When the cursor is on an existing link, *C-c C-l* allows you to edit the link and description parts of the link.

C-c C-o `org-open-at-point`
Open link at point. This will launch a web browser for URLs (using `browse-url-at-point`), run VM/MH-E/Wanderlust/Rmail/Gnus/BBDB for the corresponding links, and execute the command in a shell link. When the cursor is on an internal link, this command runs the corresponding search. When the cursor is on a TAG list in a headline, it creates the corresponding TAGS view. If the cursor is on a timestamp, it compiles the agenda for that date. Furthermore, it will visit text and remote files in '`file:`' links with Emacs and select a suitable application for local non-text files. Classification of files is based on file extension only. See option `org-file-apps`. If you want to override the default application and visit the file with Emacs, use a *C-u* prefix. If you want to avoid opening in Emacs, use a *C-u C-u* prefix.
If the cursor is on a headline, but not on a link, offer all links in the headline and entry text.

RET When `org-return-follows-link` is set, `RET` will also follow the link at point.

mouse-2
mouse-1 On links, *mouse-2* will open the link just as *C-c C-o* would. Under Emacs 22 and later, *mouse-1* will also follow a link.

mouse-3 Like *mouse-2*, but force file links to be opened with Emacs, and internal links to be displayed in another window[6].

C-c C-x C-v `org-toggle-inline-images`
Toggle the inline display of linked images. Normally this will only inline images that have no description part in the link, i.e. images that will also be inlined during export. When called with a prefix argument, also display images that do have a link description. You can ask for inline images to be displayed at startup by configuring the variable `org-startup-with-inline-images`[7].

C-c % `org-mark-ring-push`
Push the current position onto the mark ring, to be able to return easily. Commands following an internal link do this automatically.

C-c & `org-mark-ring-goto`
Jump back to a recorded position. A position is recorded by the commands following internal links, and by *C-c %*. Using this command several times in direct succession moves through a ring of previously recorded positions.

[6] See the variable `org-display-internal-link-with-indirect-buffer`

[7] with corresponding `#+STARTUP` keywords `inlineimages` and `inlineimages`

C-c C-x C-n org-next-link
C-c C-x C-p org-previous-link

Move forward/backward to the next link in the buffer. At the limit of the buffer, the search fails once, and then wraps around. The key bindings for this are really too long; you might want to bind this also to *C-n* and *C-p*

```
(add-hook 'org-load-hook
  (lambda ()
    (define-key org-mode-map "\C-n" 'org-next-link)
    (define-key org-mode-map "\C-p" 'org-previous-link)))
```

4.5 Using links outside Org

You can insert and follow links that have Org syntax not only in Org, but in any Emacs buffer. For this, you should create two global commands, like this (please select suitable global keys yourself):

```
(global-set-key "\C-c L" 'org-insert-link-global)
(global-set-key "\C-c o" 'org-open-at-point-global)
```

4.6 Link abbreviations

Long URLs can be cumbersome to type, and often many similar links are needed in a document. For this you can use link abbreviations. An abbreviated link looks like this

```
[[linkword:tag][description]]
```

where the tag is optional. The *linkword* must be a word, starting with a letter, followed by letters, numbers, '-', and '_'. Abbreviations are resolved according to the information in the variable `org-link-abbrev-alist` that relates the linkwords to replacement text. Here is an example:

```
(setq org-link-abbrev-alist
  '(("bugzilla" . "http://10.1.2.9/bugzilla/show_bug.cgi?id=")
    ("google"   . "http://www.google.com/search?q=")
    ("gmap"     . "http://maps.google.com/maps?q=%s")
    ("omap"     . "http://nominatim.openstreetmap.org/search?q=%s&polygon=1")
    ("ads"      . "http://adsabs.harvard.edu/cgi-bin/nph-abs_connect?author=%s&db_key=AST")))
```

If the replacement text contains the string '`%s`', it will be replaced with the tag. Otherwise the tag will be appended to the string in order to create the link. You may also specify a function that will be called with the tag as the only argument to create the link.

With the above setting, you could link to a specific bug with `[[bugzilla:129]]`, search the web for '`OrgMode`' with `[[google:OrgMode]]`, show the map location of the Free Software Foundation `[[gmap:51 Franklin Street, Boston]]` or of Carsten office `[[omap:Science Park 904, Amsterdam, The Netherlands]]` and find out what the Org author is doing besides Emacs hacking with `[[ads:Dominik,C]]`.

If you need special abbreviations just for a single Org buffer, you can define them in the file with

```
#+LINK: bugzilla  http://10.1.2.9/bugzilla/show_bug.cgi?id=
#+LINK: google    http://www.google.com/search?q=%s
```

In-buffer completion (see Section 15.1 [Completion], page 174) can be used after '`[`' to complete link abbreviations. You may also define a function `org-PREFIX-complete-link`

that implements special (e.g. completion) support for inserting such a link with `C-c C-l`. Such a function should not accept any arguments, and return the full link with prefix.

4.7 Search options in file links

File links can contain additional information to make Emacs jump to a particular location in the file when following a link. This can be a line number or a search option after a double[8] colon. For example, when the command `C-c l` creates a link (see Section 4.4 [Handling links], page 35) to a file, it encodes the words in the current line as a search string that can be used to find this line back later when following the link with `C-c C-o`.

Here is the syntax of the different ways to attach a search to a file link, together with an explanation:

```
[[file:~/code/main.c::255]]
[[file:~/xx.org::My Target]]
[[file:~/xx.org::*My Target]]
[[file:~/xx.org::#my-custom-id]]
[[file:~/xx.org::/regexp/]]
```

`255` Jump to line 255.

`My Target` Search for a link target '`<<My Target>>`', or do a text search for '`my target`', similar to the search in internal links, see Section 4.2 [Internal links], page 33. In HTML export (see Section 12.5 [HTML export], page 127), such a file link will become an HTML reference to the corresponding named anchor in the linked file.

`*My Target`
In an Org file, restrict search to headlines.

`#my-custom-id`
Link to a heading with a `CUSTOM_ID` property

`/regexp/` Do a regular expression search for `regexp`. This uses the Emacs command `occur` to list all matches in a separate window. If the target file is in Org-mode, `org-occur` is used to create a sparse tree with the matches.

As a degenerate case, a file link with an empty file name can be used to search the current file. For example, `[[file:::find me]]` does a search for '`find me`' in the current file, just as '`[[find me]]`' would.

4.8 Custom Searches

The default mechanism for creating search strings and for doing the actual search related to a file link may not work correctly in all cases. For example, BibTEX database files have many entries like '`year="1993"`' which would not result in good search strings, because the only unique identification for a BibTEX entry is the citation key.

If you come across such a problem, you can write custom functions to set the right search string for a particular file type, and to do the search for the string in the file.

[8] For backward compatibility, line numbers can also follow a single colon.

Using `add-hook`, these functions need to be added to the hook variables `org-create-file-search-functions` and `org-execute-file-search-functions`. See the docstring for these variables for more information. Org actually uses this mechanism for BibTEX database files, and you can use the corresponding code as an implementation example. See the file '`org-bibtex.el`'.

5 TODO アイテム

Org-mode では TODO リストを個別の文書として管理するわけではありません。[1] その変りに、TODO アイテムはノートファイルの一部として存在します。なぜなら TODO アイテムはメモを書いている最中に頭に浮かぶものだからです!Org-mode では、ツリーの中のどの項目でも簡単にマークして TODO アイテムとするだけです。この方法により特定の情報を複数個所にもつ必要は無くなり、TODO アイテムを作成するのに使用した全文書が常に最新であることになります。

もちろん、こうした手法をとることで、あなたのノートファイルの中のあちこちに、TODO アイテムが散らばることになります。それを補うために Org-mode では、やらなければならない事柄の全体を見渡す方法が提供されています。

5.1 基本的な TODO の機能

どの見出しでも 'TODO'という言葉を前につけることで、TODO アイテムとみなします。例えば:

```
*** TODO サム フォーチュンに手紙を書く。
```

TODO 項目を入力するときの重要なコマンドは以下のとおりです。

C-c C-t `org-todo`

現在の TODO の状態を次のように切り替えます。

```
,-> (マーク無し) -> TODO -> DONE --.
'----------------------------------'
```

同じような状態の切り替えは、タイムラインとアジェンダバッファで `t`コマンドキー (see Section 10.5 [Agenda commands], page 100 参照) を入力することで「リモートで」完了にすることもできます。

C-u C-c C-t

補完や「すでに設定されていれば」さらに速い選択方法を提供するインターフェイスを使用して特定のキーワードを選択します。後者の方法では、TODO の状態に対してキーを割り振る必要があります。詳細は、Section 5.2.5 [Per-file keywords], page 44 と Section 6.2 [Setting tags], page 53 を参照してください。

S-right / *S-left*

切り替えの機能に似て、後にくる TODO の状態、あるいは前にくるものを選択します。もっとも役に立つのは TODO の状態が 2 段階以上の場合です。(see Section 5.2 [TODO extensions], page 42).`shift-selection-mode`との連携については、Section 15.10.2 [Conflicts], page 184 も参照してください。変数`org-treat-S-cursor-todo-selection-as-state-change`.

C-c / t `org-show-todo-key`

も参照してください。ツリーの抽出機能を使って TODO を確認します (see Section 2.6 [Sparse trees], page 11) 参照。バッファ全体を折り畳みますが、全ての TODO 項目「DONE 状態以外の」とその階層の見出しを表示します。接頭辞をつけることで (もしくは、キーバインド *C-c / T*)、ある特定の DONE 状態の項目も表示させることができます。検索用のキーワードを入力するためのプロンプトが表示されます。さらにキーワードのリストを次のように入力することもでき `KWD1|KWD2|...`、この内のどれかに一致するものが表示されます。前置引数 N を使って、変数 `org-todo-keywords`内の N 番

[1] もちろん、長い TODO リストだけを含む個別の文書を作成することもできますが、そうする必要はないということです。

目のキーワードを含むツリーを表示することもできます。2回の前置引数を指定すると、すべての TODO 状態「DONE とそれ以外を含む」を見つけることができます。

C-c a t `org-todo-list`
グローバル TODO リストを表示します。すべての「DONE 状態以外の」TODO アイテムをすべてのアジェンダファイル (see Chapter 10 [Agenda Views], page 89) から集めて、一つのバッファに表示します。その新しくできたバッファは、`agenda-mode`で表示され、確認や修正を加えるためのコマンドも提供されます。(see Section 10.5 [Agenda commands], page 100). See Section 10.3.2 [Global TODO list], page 93. を参照してください。

S-M-RET `org-insert-todo-heading`
新しい TODO を現在の位置に入力します。

Changing a TODO state can also trigger tag changes. See the docstring of the option `org-todo-state-tags-triggers` for details.

5.2 TODO キーワードの拡張的な使い方

デフォルトでは、マークされた TODO の状態は、TODO と DONE の2つあります。さらに Org-mode は、*TODO* キーワード「`org-todo-keywords`に指定されています。」を使って、より複雑に TODO アイテムを分類できます。特別な設定により、TODO キーワードシステムは、ファイルによって異なる働きにすることできます。

注記、タグは見出しと特に TODO アイテムの分類のもう一つの方法です。(see Chapter 6 [Tags], page 53).

5.2.1 ワークフローの状態としての TODO キーワード

TODO キーワードを使用して、アイテムの連続した異なる状態を表すことができます。例えば、[2]:

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

縦線は、TODO キーワード「アクションが必要な状態」と DONE 状態「アクションが必要ない状態」からを分離します。縦線が指定されていない場合は、最後の状態が、DONE 状態として使用されます。この設定により、コマンド *C-c C-t*で、TODO、FEEDBACK、それから VERIFY、最後に DONE、DELEGATED というように順番切り替えます。前置引数を使用することで、特定の状態を即座に選択することもできます。例えば *C-3 C-c C-t*と入力すると、ダイレクトに3番目の VERIFY に変更することができます。もしくは、*S-left*により、逆の方向に順番に切り替えることもできます。もしも、たくさんのキーワードを定義した場合は、入力補完機能（see Section 15.1 [Completion], page 174）か特別な一つのキーによる選択によりバッファに入力することができます。(see Section 5.2.4 [Fast access to TODO states], page 43)TODO の状態の変更は、タイムスタンプと共にログをとることができます。(より詳しい情報は Section 5.3.2 [Tracking TODO state changes], page 46 参照)

5.2.2 種類としての TODO キーワード

TODO キーワードの2つ目の使い方として、異なる種類のアクションアイテムを定義できることです。例えば、アイテムを「仕事」または「家庭」を示すようにも使えます。もしくは、複数の人と同じプロジェクトに参加するとき、その中の何人かに彼らの名前を使って直接アクションアイテムを割り当てたいかもしれません。これは、以下のように設定します。:

[2] この変数の変更は、Org-mode をバッファ内で再起動した場合のみ有効になります。

```
(setq org-todo-keywords '((type "Fred" "Sara" "Lucy" "|" "DONE")))
```

この場合、それぞれのキーワードは作業の順序を表しているのではなく、別々のタイプを表すことになります。そのため、通常の作業の流れとしてタスクを一人に割り振ることになり、その後の DONE になります。Org-mode は、このような形式をサポートするため、「*C-c C-t*」コマンドの動作が少し変化します ([3])。まずは、適当なタイプを選択するのに、繰り返し押すことで、順番にキーワードの名称が表示されます。しばらく間をおいてその項目に戻ってきて、「*C-c C-t*」を再度実行すると、そのときは、すぐ DONE に切り替えられます。前置引数か補完を使えば、適当なタイプをすぐに選ぶことができます。さらに *C-c / t*に前置引数を指定することにより、抽出されたツリーの中で探している TODO のタイプを確認することもできます。例えば、Lucy がやらねばならないにすべての項目を見るには、「*C-3 C-c / t*」を実行します。すべてのアジェンダのファイルの中から Lucy の項目を一つのバッファに集約するのに、グローバルな todo リストを作成し、次のように前置引数を使用します:「*C-3 C-c a t*」。

5.2.3 同一ファイル内での複数のキーワードセット

時には、異なるセットの TODO キーワードを同時に使いたい場合があるかもしれません。例えば、通常の TODO/DONEを使用しつつ、バグフィックスのワークフロー、さらにアイテムがキャンセルをされたことを表すその次の状態を使用したい場合などです「つまり DONE ではないが、次のアクションが必要ない場合」。その場合の設定は次のようになります:

```
(setq org-todo-keywords
      '((sequence "TODO" "|" "DONE")
        (sequence "REPORT" "BUG" "KNOWNCAUSE" "|" "FIXED")
        (sequence "|" "CANCELED")))
```

キーワードは、すべて異るようにすべきで、そうすると Org-mode が、現在の状態の次に続くものを認識するのに役立ちます。この設定では、*C-c C-t*は、サブグループ内だけで働きます。つまり DONEから (何も無い状態) から TODOへ、そして FIXEDから (何も無い状態) から REPORTへ。その為、まず使いたいサブグループを選ぶ方法が必要です。当然通常行うようにキーワードをタイプするか、補完、または次のコマンドを使ううこともできます:

C-u C-u C-c C-t
C-S-right
C-S-left These keys jump from one TODO subset to the next. In the above example, *C-u C-u C-c C-t* or *C-S-right* would jump from TODO or DONE to REPORT, and any of the words in the second row to CANCELED. Note that the *C-S-* key binding conflict with `shift-selection-mode` (see Section 15.10.2 [Conflicts], page 184).

S-right
S-left *S-<left>*と *S-<right>*は、すべてのサブグループのすべてのキーワード切り替えいきます。例えば、上記の例では、*S-<right>*は、DONEに切り替えられ、さらに REPORT になります。`shift-selection-mode`と連携させる方法については、Section 15.10.2 [Conflicts], page 184 を参照してください。

5.2.4 Fast access to TODO states

もし、切り替えせずに任意の TODO の状態にすばやく変更したい場合は、キー登録して一文字でその状態に変更できます。それには、各キーワードに対して括弧で括ってセクションキーを割り当てることにより実現できます。例えば:

[3] タイムラインやアジェンダのバッファでは、「t」コマンドも同じ仕様です。

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

C-c C-tを押して、選択の為のキーを押せば、その選ばれた状態へ切り替えられます。さらに SPC を使って、どの TODO キーワードも削除することができます。[4]

5.2.5 ファイル別にキーワードを設定する

異なるファイルごとに、TODO の機能をさまざまなの方法で使用できるととても便利です。ファイル単位のローカルな設定をするためには、そのファイルだけに通用するキーワードを特別な行を記入することで設定する必要があります。例えば、前述した２つの例のうちの一つを設定するの場合、次のような行を、そのファイルのどこかで行頭から開始する必要があります。

```
#+TODO: TODO FEEDBACK VERIFY | DONE CANCELED
```

(you may also write #+SEQ_TODO to be explicit about the interpretation, but it means the same as #+TODO), or

```
#+TYP_TODO: Fred Sara Lucy Mike | DONE
```

同時に複数のキーワードセットの設定には:

```
#+TODO: TODO | DONE
#+TODO: REPORT BUG KNOWNCAUSE | FIXED
#+TODO: | CANCELED
```

To make sure you are using the correct keyword, type 間違いなく正しいキーワードを使うため、そのバッファ内で '#+'をタイプして、*M-TAB*を使って補完してください。

縦線の後のキーワード「もしくは、縦線が指定されてない場合は、最後のキーワード」は、そのアイテムがいつも DONE「最後のもの」であることを覚えていてください「と言っても DONE 以外のキーワードも使えます」。これらの変更を加えた後、Org-mode に変更を認識させるため、カーソルを変更した場所に置いたままで *C-c C-c* してください。[5]

5.2.6 Faces for TODO keywords

Org-mode は、TODO キーワードを特別なフェイスを使ってハイライトします:org-todoは、あるアイテムがアクションが必要なキーワードであることを指しています。org-doneは、あるアイテムが完了していることを指しています。もし２つ以上の異なる状態を使用しているのであれば、特別なフェイスを使いたくなるかもしれません。これは、変数 org-todo-keyword-facesを変更することで可能です。例えば:

```
(setq org-todo-keyword-faces
      '(("TODO" . org-warning) ("STARTED" . "yellow")
        ("CANCELED" . (:foreground "blue" :weight bold))))
```

CANCELED にあるようにフェイスプロパティのリストを使うのは、上手くいくはずですが、いつもうまくいってるように見えないかもしれません。必要であれば、特別なフェイスを定義してそれ

---

[4] 変数 org-fast-tag-selection-include-todoも見てください、この変数は、タグを使って状態の変更を可能にします (see Section 6.2 [Setting tags], page 53)、この二つを混ぜて使いたいならですが。この場合、それぞれのキーワードセットに単一なキーを準備する必要があります。

[5] Org-mode がこれらの行を読み込むのは、ファイルを開いて Org-mode が実行された場合だけです。'#+'で始まる行にカーソルを置いて *C-c C-c*をすると、現在のバッファで Org-mode を再起動したことになります。

を使うのもいいかもしれません。文字列は、カラーとして解釈されます。変数 `org-faces-easy-properties`により、文字の色にするか、背景色にするか指定できます。

5.2.7 TODO dependencies

The structure of Org files (hierarchy and lists) makes it easy to define TODO dependencies. Usually, a parent TODO task should not be marked DONE until all subtasks (defined as children tasks) are marked as DONE. And sometimes there is a logical sequence to a number of (sub)tasks, so that one task cannot be acted upon before all siblings above it are done. If you customize the variable `org-enforce-todo-dependencies`, Org will block entries from changing state to DONE while they have children that are not DONE. Furthermore, if an entry has a property `ORDERED`, each of its children will be blocked until all earlier siblings are marked DONE. Here is an example:

```
* TODO Blocked until (two) is done
** DONE one
** TODO two

* Parent
  :PROPERTIES:
    :ORDERED: t
  :END:
** TODO a
** TODO b, needs to wait for (a)
** TODO c, needs to wait for (a) and (b)
```

C-c C-x o `org-toggle-ordered-property`
: Toggle the `ORDERED` property of the current entry. A property is used for this behavior because this should be local to the current entry, not inherited like a tag. However, if you would like to *track* the value of this property with a tag for better visibility, customize the variable `org-track-ordered-property-with-tag`.

C-u C-u C-u C-c C-t
: Change TODO state, circumventing any state blocking.

If you set the variable `org-agenda-dim-blocked-tasks`, TODO entries that cannot be closed because of such dependencies will be shown in a dimmed font or even made invisible in agenda views (see Chapter 10 [Agenda Views], page 89).

You can also block changes of TODO states by looking at checkboxes (see Section 5.6 [Checkboxes], page 50). If you set the variable `org-enforce-todo-checkbox-dependencies`, an entry that has unchecked checkboxes will be blocked from switching to DONE.

If you need more complex dependency structures, for example dependencies between entries in different trees or files, check out the contributed module '`org-depend.el`'.

5.3 Progress logging

Org-mode can automatically record a timestamp and possibly a note when you mark a TODO item as DONE, or even each time you change the state of a TODO item. This system is highly configurable, settings can be on a per-keyword basis and can be localized to a file or even a subtree. For information on how to clock working time for a task, see Section 8.4 [Clocking work time], page 72.

5.3.1 Closing items

The most basic logging is to keep track of *when* a certain TODO item was finished. This is achieved with[1]

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

Then each time you turn an entry from a TODO (not-done) state into any of the DONE states, a line '`CLOSED: [timestamp]`' will be inserted just after the headline. If you turn the entry back into a TODO item through further state cycling, that line will be removed again. If you want to record a note along with the timestamp, use[2]

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

You will then be prompted for a note, and that note will be stored below the entry with a '`Closing Note`' heading.

In the timeline (see Section 10.3.4 [Timeline], page 96) and in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 91), you can then use the `l` key to display the TODO items with a '`CLOSED`' timestamp on each day, giving you an overview of what has been done.

5.3.2 Tracking TODO state changes

When TODO keywords are used as workflow states (see Section 5.2.1 [Workflow states], page 42), you might want to keep track of when a state change occurred and maybe take a note about this change. You can either record just a timestamp, or a time-stamped note for a change. These records will be inserted after the headline as an itemized list, newest first[3]. When taking a lot of notes, you might want to get the notes out of the way into a drawer (see Section 2.8 [Drawers], page 15). Customize the variable `org-log-into-drawer` to get this behavior—the recommended drawer for this is called `LOGBOOK`. You can also overrule the setting of this variable for a subtree by setting a `LOG_INTO_DRAWER` property.

Since it is normally too much to record a note for every state, Org-mode expects configuration on a per-keyword basis for this. This is achieved by adding special markers '`!`' (for a timestamp) and @'' (for a note) in parentheses after each keyword. For example, with the setting

```
(setq org-todo-keywords
  '((sequence "TODO(t)" "WAIT(w@/!)" "|" "DONE(d!)" "CANCELED(c@)")))
```

[1] The corresponding in-buffer setting is: `#+STARTUP: logdone`

[2] The corresponding in-buffer setting is: `#+STARTUP: lognotedone`

[3] See the variable `org-log-states-order-reversed`

you not only define global TODO keywords and fast access keys, but also request that a time is recorded when the entry is set to DONE[4], and that a note is recorded when switching to WAIT or CANCELED. The setting for WAIT is even more special: the '`!`' after the slash means that in addition to the note taken when entering the state, a timestamp should be recorded when *leaving* the WAIT state, if and only if the *target* state does not configure logging for entering it. So it has no effect when switching from WAIT to DONE, because DONE is configured to record a timestamp only. But when switching from WAIT back to TODO, the '`/!`' in the WAIT setting now triggers a timestamp even though TODO has no logging configured.

You can use the exact same syntax for setting logging preferences local to a buffer:

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

In order to define logging settings that are local to a subtree or a single item, define a LOGGING property in this entry. Any non-empty LOGGING property resets all logging settings to nil. You may then turn on logging for this specific tree using STARTUP keywords like `lognotedone` or `logrepeat`, as well as adding state specific settings like `TODO(!)`. For example

```
* TODO Log each state with only a time
  :PROPERTIES:
  :LOGGING: TODO(!) WAIT(!) DONE(!) CANCELED(!)
  :END:
* TODO Only log when switching to WAIT, and when repeating
  :PROPERTIES:
  :LOGGING: WAIT(@) logrepeat
  :END:
* TODO No logging at all
  :PROPERTIES:
  :LOGGING: nil
  :END:
```

5.3.3 習慣の追跡

Org has the ability to track the consistency of a special category of TODOs, called "habits". A habit has the following properties:

1. You have enabled the `habits` module by customizing the variable `org-modules`.
2. The habit is a TODO, with a TODO keyword representing an open state.
3. The property `STYLE` is set to the value `habit`.
4. The TODO has a scheduled date, usually with a `.+` style repeat interval. A `++` style may be appropriate for habits with time constraints, e.g., must be done on weekends, or a `+` style for an unusual habit that can have a backlog, e.g., weekly reports.
5. The TODO may also have minimum and maximum ranges specified by using the syntax '`.+2d/3d`', which says that you want to do the task at least every three days, but at most every two days.

[4] It is possible that Org-mode will record two timestamps when you are using both `org-log-done` and state change logging. However, it will never prompt for two notes—if you have configured both, the state change recording note will take precedence and cancel the '`Closing Note`'.

6. You must also have state logging for the DONE state enabled, in order for historical data to be represented in the consistency graph. If it's not enabled it's not an error, but the consistency graphs will be largely meaningless.

To give you an idea of what the above rules look like in action, here's an actual habit with some history:

```
** TODO Shave
   SCHEDULED: <2009-10-17 Sat .+2d/4d>
   - State "DONE"       from "TODO"       [2009-10-15 Thu]
   - State "DONE"       from "TODO"       [2009-10-12 Mon]
   - State "DONE"       from "TODO"       [2009-10-10 Sat]
   - State "DONE"       from "TODO"       [2009-10-04 Sun]
   - State "DONE"       from "TODO"       [2009-10-02 Fri]
   - State "DONE"       from "TODO"       [2009-09-29 Tue]
   - State "DONE"       from "TODO"       [2009-09-25 Fri]
   - State "DONE"       from "TODO"       [2009-09-19 Sat]
   - State "DONE"       from "TODO"       [2009-09-16 Wed]
   - State "DONE"       from "TODO"       [2009-09-12 Sat]
   :PROPERTIES:
   :STYLE:    habit
   :LAST_REPEAT: [2009-10-19 Mon 00:36]
   :END:
```

What this habit says is: I want to shave at most every 2 days (given by the SCHEDULED date and repeat interval) and at least every 4 days. If today is the 15th, then the habit first appears in the agenda on Oct 17, after the minimum of 2 days has elapsed, and will appear overdue on Oct 19, after four days have elapsed.

What's really useful about habits is that they are displayed along with a consistency graph, to show how consistent you've been at getting that task done in the past. This graph shows every day that the task was done over the past three weeks, with colors for each day. The colors used are:

`Blue` If the task wasn't to be done yet on that day.

`Green` If the task could have been done on that day.

`Yellow` If the task was going to be overdue the next day.

`Red` If the task was overdue on that day.

In addition to coloring each day, the day is also marked with an asterisk if the task was actually done that day, and an exclamation mark to show where the current day falls in the graph.

There are several configuration variables that can be used to change the way habits are displayed in the agenda.

`org-habit-graph-column`
: The buffer column at which the consistency graph should be drawn. This will overwrite any text in that column, so it's a good idea to keep your habits' titles brief and to the point.

`org-habit-preceding-days`
: The amount of history, in days before today, to appear in consistency graphs.

`org-habit-following-days`
: The number of days after today that will appear in consistency graphs.

`org-habit-show-habits-only-for-today`
: If non-nil, only show habits in today's agenda view. This is set to true by default.

Lastly, pressing *K* in the agenda buffer will cause habits to temporarily be disabled and they won't appear at all. Press *K* again to bring them back. They are also subject to tag filtering, if you have habits which should only be done in certain contexts, for example.

5.4 Priorities

If you use Org-mode extensively, you may end up with enough TODO items that it starts to make sense to prioritize them. Prioritizing can be done by placing a *priority cookie* into the headline of a TODO item, like this

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

By default, Org-mode supports three priorities: '`A`', '`B`', and '`C`'. '`A`' is the highest priority. An entry without a cookie is treated just like priority '`B`'. Priorities make a difference only for sorting in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 91); outside the agenda, they have no inherent meaning to Org-mode. The cookies can be highlighted with special faces by customizing the variable `org-priority-faces`.

Priorities can be attached to any outline node; they do not need to be TODO items.

C-c ,
: Set the priority of the current headline (`org-priority`). The command prompts for a priority character '`A`', '`B`' or '`C`'. When you press `SPC` instead, the priority cookie is removed from the headline. The priorities can also be changed "remotely" from the timeline and agenda buffer with the `,` command (see Section 10.5 [Agenda commands], page 100).

S-up `org-priority-up`
S-down `org-priority-down`
: Increase/decrease priority of current headline[5]. Note that these keys are also used to modify timestamps (see Section 8.2 [Creating timestamps], page 66). See also Section 15.10.2 [Conflicts], page 184, for a discussion of the interaction with `shift-selection-mode`.

You can change the range of allowed priorities by setting the variables `org-highest-priority`, `org-lowest-priority`, and `org-default-priority`. For an individual buffer, you may set these values (highest, lowest, default) like this (please make sure that the highest priority is earlier in the alphabet than the lowest priority):

```
#+PRIORITIES: A C B
```

[5] See also the option `org-priority-start-cycle-with-default`.

5.5 Breaking tasks down into subtasks

It is often advisable to break down large tasks into smaller, manageable subtasks. You can do this by creating an outline tree below a TODO item, with detailed subtasks on the tree[6]. To keep the overview over the fraction of subtasks that are already completed, insert either '`[/]`' or '`[%]`' anywhere in the headline. These cookies will be updated each time the TODO status of a child changes, or when pressing `C-c C-c` on the cookie. For example:

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

If a heading has both checkboxes and TODO children below it, the meaning of the statistics cookie become ambiguous. Set the property `COOKIE_DATA` to either '`checkbox`' or '`todo`' to resolve this issue.

If you would like to have the statistics cookie count any TODO entries in the subtree (not just direct children), configure the variable `org-hierarchical-todo-statistics`. To do this for a single subtree, include the word '`recursive`' into the value of the `COOKIE_DATA` property.

```
* Parent capturing statistics [2/20]
  :PROPERTIES:
  :COOKIE_DATA: todo recursive
  :END:
```

If you would like a TODO entry to automatically change to DONE when all children are done, you can use the following setup:

```
(defun org-summary-todo (n-done n-not-done)
  "Switch entry to DONE when all subentries are done, to TODO otherwise."
  (let (org-log-done org-log-states)   ; turn off logging
    (org-todo (if (= n-not-done 0) "DONE" "TODO"))))

(add-hook 'org-after-todo-statistics-hook 'org-summary-todo)
```

Another possibility is the use of checkboxes to identify (a hierarchy of) a large number of subtasks (see Section 5.6 [Checkboxes], page 50).

5.6 Checkboxes

Every item in a plain list[7] (see Section 2.7 [Plain lists], page 12) can be made into a checkbox by starting it with the string '`[ ]`'. This feature is similar to TODO items (see Chapter 5 [TODO Items], page 41), but is more lightweight. Checkboxes are not included into the global TODO list, so they are often great to split a task into a number of simple steps. Or you can use them in a shopping list. To toggle a checkbox, use `C-c C-c`, or use the mouse (thanks to Piotr Zielinski's '`org-mouse.el`').

[6] To keep subtasks out of the global TODO list, see the `org-agenda-todo-list-sublevels`.

[7] With the exception of description lists. But you can allow it by modifying `org-list-automatic-rules` accordingly.

Here is an example of a checkbox list.

```
* TODO Organize party [2/4]
  - [-] call people [1/3]
    - [ ] Peter
    - [X] Sarah
    - [ ] Sam
  - [X] order food
  - [ ] think about what music to play
  - [X] talk to the neighbors
```

Checkboxes work hierarchically, so if a checkbox item has children that are checkboxes, toggling one of the children checkboxes will make the parent checkbox reflect if none, some, or all of the children are checked.

The '`[2/4]`' and '`[1/3]`' in the first and second line are cookies indicating how many checkboxes present in this entry have been checked off, and the total number of checkboxes present. This can give you an idea on how many checkboxes remain, even without opening a folded entry. The cookies can be placed into a headline or into (the first line of) a plain list item. Each cookie covers checkboxes of direct children structurally below the headline/item on which the cookie appears[8]. You have to insert the cookie yourself by typing either '`[/]`' or '`[%]`'. With '`[/]`' you get an '`n out of m`' result, as in the examples above. With '`[%]`' you get information about the percentage of checkboxes checked (in the above example, this would be '`[50%]`' and '`[33%]`', respectively). In a headline, a cookie can count either checkboxes below the heading or TODO states of children, and it will display whatever was changed last. Set the property `COOKIE_DATA` to either '`checkbox`' or '`todo`' to resolve this issue.

If the current outline node has an `ORDERED` property, checkboxes must be checked off in sequence, and an error will be thrown if you try to check off a box while there are unchecked boxes above it.

The following commands work with checkboxes:

C-c C-c `org-toggle-checkbox`

Toggle checkbox status or (with prefix arg) checkbox presence at point. With double prefix argument, set it to '`[-]`', which is considered to be an intermediate state.

C-c C-x C-b `org-toggle-checkbox`

Toggle checkbox status or (with prefix arg) checkbox presence at point. With double prefix argument, set it to '`[-]`', which is considered to be an intermediate state.

- If there is an active region, toggle the first checkbox in the region and set all remaining boxes to the same status as the first. With a prefix arg, add or remove the checkbox for all items in the region.
- If the cursor is in a headline, toggle checkboxes in the region between this headline and the next (so *not* the entire subtree).
- If there is no active region, just toggle the checkbox at point.

[8] Set the variable `org-hierarchical-checkbox-statistics` if you want such cookies to represent the all checkboxes below the cookie, not just the direct children.

M-S-RET `org-insert-todo-heading`
: Insert a new item with a checkbox. This works only if the cursor is already in a plain list item (see Section 2.7 [Plain lists], page 12).

C-c C-x o `org-toggle-ordered-property`
: Toggle the `ORDERED` property of the entry, to toggle if checkboxes must be checked off in sequence. A property is used for this behavior because this should be local to the current entry, not inherited like a tag. However, if you would like to *track* the value of this property with a tag for better visibility, customize the variable `org-track-ordered-property-with-tag`.

C-c # `org-update-statistics-cookies`
: Update the statistics cookie in the current outline entry. When called with a *C-u* prefix, update the entire file. Checkbox statistic cookies are updated automatically if you toggle checkboxes with *C-c C-c* and make new ones with *M-S-RET*. TODO statistics cookies update when changing TODO states. If you delete boxes/entries or add/change them by hand, use this command to get things back into sync. Or simply toggle any entry twice (checkboxes with *C-c C-c*).

6 Tags

An excellent way to implement labels and contexts for cross-correlating information is to assign *tags* to headlines. Org-mode has extensive support for tags.

Every headline can contain a list of tags; they occur at the end of the headline. Tags are normal words containing letters, numbers, '_', and '@'. Tags must be preceded and followed by a single colon, e.g., ':work:'. Several tags can be specified, as in ':work:urgent:'. Tags will by default be in bold face with the same color as the headline. You may specify special faces for specific tags using the variable `org-tag-faces`, in much the same way as you can for TODO keywords (see Section 5.2.6 [Faces for TODO keywords], page 44).

6.1 Tag inheritance

Tags make use of the hierarchical structure of outline trees. If a heading has a certain tag, all subheadings will inherit the tag as well. For example, in the list

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

the final heading will have the tags ':work:', ':boss:', ':notes:', and ':action:' even though the final heading is not explicitly marked with those tags. You can also set tags that all entries in a file should inherit just as if these tags were defined in a hypothetical level zero that surrounds the entire file. Use a line like this[1]:

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

To limit tag inheritance to specific tags, or to turn it off entirely, use the variables `org-use-tag-inheritance` and `org-tags-exclude-from-inheritance`.

When a headline matches during a tags search while tag inheritance is turned on, all the sublevels in the same tree will (for a simple match form) match as well[2]. The list of matches may then become very long. If you only want to see the first tags match in a subtree, configure the variable `org-tags-match-list-sublevels` (not recommended).

6.2 Setting tags

Tags can simply be typed into the buffer at the end of a headline. After a colon, *M-TAB* offers completion on tags. There is also a special command for inserting tags:

C-c C-q `org-set-tags-command`

Enter new tags for the current headline. Org-mode will either offer completion or a special single-key interface for setting tags, see below. After pressing `RET`, the tags will be inserted and aligned to `org-tags-column`. When called with a *C-u* prefix, all tags in the current buffer will be aligned to that column, just to make things look nice. TAGS are automatically realigned after promotion, demotion, and TODO state changes (see Section 5.1 [TODO basics], page 41).

[1] As with all these in-buffer settings, pressing *C-c C-c* activates any changes in the line.

[2] This is only true if the search does not involve more complex tests including properties (see Section 7.3 [Property searches], page 59).

`C-c C-c` `org-set-tags-command`

When the cursor is in a headline, this does the same as `C-c C-q`.

Org will support tag insertion based on a *list of tags*. By default this list is constructed dynamically, containing all tags currently used in the buffer. You may also globally specify a hard list of tags with the variable `org-tag-alist`. Finally you can set the default tags for a given file with lines like

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

If you have globally defined your preferred set of tags using the variable `org-tag-alist`, but would like to use a dynamic tag list in a specific file, add an empty TAGS option line to that file:

```
#+TAGS:
```

If you have a preferred set of tags that you would like to use in every file, in addition to those defined on a per-file basis by TAGS option lines, then you may specify a list of tags with the variable `org-tag-persistent-alist`. You may turn this off on a per-file basis by adding a STARTUP option line to that file:

```
#+STARTUP: noptag
```

By default Org-mode uses the standard minibuffer completion facilities for entering tags. However, it also implements another, quicker, tag selection method called *fast tag selection*. This allows you to select and deselect tags with just a single key press. For this to work well you should assign unique letters to most of your commonly used tags. You can do this globally by configuring the variable `org-tag-alist` in your '`.emacs`' file. For example, you may find the need to tag many items in different files with '`:@home:`'. In this case you can set something like:

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

If the tag is only relevant to the file you are working on, then you can instead set the TAGS option line as:

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

The tags interface will show the available tags in a splash window. If you want to start a new line after a specific tag, insert '`\n`' into the tag list

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

or write them in two lines:

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

You can also group together tags that are mutually exclusive by using braces, as in:

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

you indicate that at most one of '`@work`', '`@home`', and '`@tennisclub`' should be selected. Multiple such groups are allowed.

Don't forget to press `C-c C-c` with the cursor in one of these lines to activate any changes.

To set these mutually exclusive groups in the variable `org-tags-alist`, you must use the dummy tags `:startgroup` and `:endgroup` instead of the braces. Similarly, you can use `:newline` to indicate a line break. The previous example would be set globally by the following configuration:

```
(setq org-tag-alist '((:startgroup . nil)
                      ("@work" . ?w) ("@home" . ?h)
                      ("@tennisclub" . ?t)
                      (:endgroup . nil)
                      ("laptop" . ?l) ("pc" . ?p)))
```

If at least one tag has a selection key then pressing *C-c C-c* will automatically present you with a special interface, listing inherited tags, the tags of the current headline, and a list of all valid tags with corresponding keys[3]. In this interface, you can use the following keys:

`a-z...` Pressing keys assigned to tags will add or remove them from the list of tags in the current line. Selecting a tag in a group of mutually exclusive tags will turn off any other tags from that group.

`TAB` Enter a tag in the minibuffer, even if the tag is not in the predefined list. You will be able to complete on all tags present in the buffer. You can also add several tags: just separate them with a comma.

`SPC` Clear all tags for this line.

`RET` Accept the modified set.

C-g Abort without installing changes.

q If *q* is not assigned to a tag, it aborts like *C-g*.

! Turn off groups of mutually exclusive tags. Use this to (as an exception) assign several tags from such a group.

C-c Toggle auto-exit after the next change (see below). If you are using expert mode, the first *C-c* will display the selection window.

This method lets you assign tags to a headline with very few keys. With the above setup, you could clear the current tags and set '`@home`', '`laptop`' and '`pc`' tags with just the following keys: *C-c C-c SPC h l p RET*. Switching from @'`home`' to @'`work`' would be done with *C-c C-c w RET* or alternatively with *C-c C-c C-c w*. Adding the non-predefined tag '`Sarah`' could be done with *C-c C-c TAB S* `a` *r* `a` *h RET RET*.

If you find that most of the time you need only a single key press to modify your list of tags, set the variable `org-fast-tag-selection-single-key`. Then you no longer have to press `RET` to exit fast tag selection—it will immediately exit after the first change. If you then occasionally need more keys, press *C-c* to turn off auto-exit for the current tag selection process (in effect: start selection with *C-c C-c C-c* instead of *C-c C-c*). If you set the variable to the value `expert`, the special window is not even shown for single-key tag selection, it comes up only when you press an extra *C-c*.

6.3 Tag searches

Once a system of tags has been set up, it can be used to collect related information into special lists.

[3] Keys will automatically be assigned to tags which have no configured keys.

`C-c / m` or `C-c \` `org-match-sparse-tree`
: Create a sparse tree with all headlines matching a tags search. With a `C-u` prefix argument, ignore headlines that are not a TODO line.

`C-c a m` `org-tags-view`
: Create a global list of tag matches from all agenda files. See Section 10.3.3 [Matching tags and properties], page 94.

`C-c a M` `org-tags-view`
: Create a global list of tag matches from all agenda files, but check only TODO items and force checking subitems (see variable `org-tags-match-list-sublevels`).

These commands all prompt for a match string which allows basic Boolean logic like '`+boss+urgent-project1`', to find entries with tags '`boss`' and '`urgent`', but not '`project1`', or '`Kathy|Sally`' to find entries which are tagged, like '`Kathy`' or '`Sally`'. The full syntax of the search string is rich and allows also matching against TODO keywords, entry levels and properties. For a complete description with many examples, see Section 10.3.3 [Matching tags and properties], page 94.

7 プロパティ（属性）とカラム（列）

Properties are a set of key-value pairs associated with an entry. There are two main applications for properties in Org-mode. First, properties are like tags, but with a value. Second, you can use properties to implement (very basic) database capabilities in an Org buffer. For an example of the first application, imagine maintaining a file where you document bugs and plan releases for a piece of software. Instead of using tags like `:release_1:`, `:release_2:`, one can use a property, say `:Release:`, that in different subtrees has different values, such as `1.0` or `2.0`. For an example of the second application of properties, imagine keeping track of your music CDs, where properties could be things such as the album, artist, date of release, number of tracks, and so on.

Properties can be conveniently edited and viewed in column view (see Section 7.5 [Column view], page 60).

7.1 Property syntax

Properties are key-value pairs. They need to be inserted into a special drawer (see Section 2.8 [Drawers], page 15) with the name `PROPERTIES`. Each property is specified on a single line, with the key (surrounded by colons) first, and the value after it. Here is an example:

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

You may define the allowed values for a particular property '`:Xyz:`' by setting a property '`:Xyz_ALL:`'. This special property is *inherited*, so if you set it in a level 1 entry, it will apply to the entire tree. When allowed values are defined, setting the corresponding property becomes easier and is less prone to typing errors. For the example with the CD collection, we can predefine publishers and the number of disks in a box like this:

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

If you want to set properties that can be inherited by any entry in a file, use a line like

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

Property values set with the global variable `org-global-properties` can be inherited by all entries in all Org files.

The following commands help to work with properties:

M-TAB `pcomplete`
: After an initial colon in a line, complete property keys. All keys used in the current file will be offered as possible completions.

C-c C-x p `org-set-property`
: Set a property. This prompts for a property name and a value. If necessary, the property drawer is created as well.

M-x org-insert-property-drawer
: Insert a property drawer into the current entry. The drawer will be inserted early in the entry, but after the lines with planning information like deadlines.

C-c C-c `org-property-action`
: With the cursor in a property drawer, this executes property commands.

C-c C-c s `org-set-property`
: Set a property in the current entry. Both the property and the value can be inserted using completion.

S-right `org-property-next-allowed-value`
S-left `org-property-previous-allowed-value`
: Switch property at point to the next/previous allowed value.

C-c C-c d `org-delete-property`
: Remove a property from the current entry.

C-c C-c D `org-delete-property-globally`
: Globally remove a property, from all entries in the current file.

C-c C-c c `org-compute-property-at-point`
: Compute the property at point, using the operator and scope from the nearest column format definition.

7.2 Special properties

Special properties provide an alternative access method to Org-mode features, like the TODO state or the priority of an entry, discussed in the previous chapters. This interface exists so that you can include these states in a column view (see Section 7.5 [Column view], page 60), or to use them in queries. The following property names are special and (except for `:CATEGORY:`) should not be used as keys in the properties drawer:

`TODO`	The TODO keyword of the entry.
`TAGS`	The tags defined directly in the headline.
`ALLTAGS`	All tags, including inherited ones.
`CATEGORY`	The category of an entry.
`PRIORITY`	The priority of the entry, a string with a single letter.
`DEADLINE`	The deadline time string, without the angular brackets.
`SCHEDULED`	The scheduling timestamp, without the angular brackets.
`CLOSED`	When was this entry closed?
`TIMESTAMP`	The first keyword-less timestamp in the entry.
`TIMESTAMP_IA`	The first inactive timestamp in the entry.
`CLOCKSUM`	The sum of CLOCK intervals in the subtree. `org-clock-sum` must be run first to compute the values.

`BLOCKED` "t" if task is currently blocked by children or siblings
`ITEM` The content of the entry.
`FILE` The filename the entry is located in.

7.3 Property searches

To create sparse trees and special lists with selection based on properties, the same commands are used as for tag searches (see Section 6.3 [Tag searches], page 55).

C-c / m or *C-c org-match-sparse-tree* Create a sparse tree with all matching entries. With a *C-u* prefix argument
ignore headlines that are not a TODO line.

C-c a m org-tags-view
Create a global list of tag/property matches from all agenda files. See Section 10.3.3 [Matching tags and properties], page 94.

C-c a M org-tags-view
Create a global list of tag matches from all agenda files, but check only TODO items and force checking of subitems (see variable `org-tags-match-list-sublevels`).

The syntax for the search string is described in Section 10.3.3 [Matching tags and properties], page 94.

There is also a special command for creating sparse trees based on a single property:

C-c / p Create a sparse tree based on the value of a property. This first prompts for the name of a property, and then for a value. A sparse tree is created with all entries that define this property with the given value. If you enclose the value in curly braces, it is interpreted as a regular expression and matched against the property values.

7.4 Property Inheritance

The outline structure of Org-mode documents lends itself to an inheritance model of properties: if the parent in a tree has a certain property, the children can inherit this property. Org-mode does not turn this on by default, because it can slow down property searches significantly and is often not needed. However, if you find inheritance useful, you can turn it on by setting the variable `org-use-property-inheritance`. It may be set to `t` to make all properties inherited from the parent, to a list of properties that should be inherited, or to a regular expression that matches inherited properties. If a property has the value '`nil`', this is interpreted as an explicit undefine of the property, so that inheritance search will stop at this value and return `nil`.

Org-mode has a few properties for which inheritance is hard-coded, at least for the special applications for which they are used:

`COLUMNS` The `:COLUMNS:` property defines the format of column view (see Section 7.5 [Column view], page 60). It is inherited in the sense that the level where a `:COLUMNS:` property is defined is used as the starting point for a column view table, independently of the location in the subtree from where columns view is turned on.

`CATEGORY` For agenda view, a category set through a `:CATEGORY:` property applies to the entire subtree.

`ARCHIVE` For archiving, the `:ARCHIVE:` property may define the archive location for the entire subtree (see Section 9.6.1 [Moving subtrees], page 87).

`LOGGING` The LOGGING property may define logging settings for an entry or a subtree (see Section 5.3.2 [Tracking TODO state changes], page 46).

7.5 Column view

A great way to view and edit properties in an outline tree is *column view*. In column view, each outline node is turned into a table row. Columns in this table provide access to properties of the entries. Org-mode implements columns by overlaying a tabular structure over the headline of each item. While the headlines have been turned into a table row, you can still change the visibility of the outline tree. For example, you get a compact table by switching to CONTENTS view (*S-TAB S-TAB*, or simply *c* while column view is active), but you can still open, read, and edit the entry below each headline. Or, you can switch to column view after executing a sparse tree command and in this way get a table only for the selected items. Column view also works in agenda buffers (see Chapter 10 [Agenda Views], page 89) where queries have collected selected items, possibly from a number of files.

7.5.1 Defining columns

Setting up a column view first requires defining the columns. This is done by defining a column format line.

7.5.1.1 Scope of column definitions

To define a column format for an entire file, use a line like

```
#+COLUMNS: %25ITEM %TAGS %PRIORITY %TODO
```

To specify a format that only applies to a specific tree, add a `:COLUMNS:` property to the top node of that tree, for example:

```
** Top node for columns view
   :PROPERTIES:
   :COLUMNS: %25ITEM %TAGS %PRIORITY %TODO
   :END:
```

If a `:COLUMNS:` property is present in an entry, it defines columns for the entry itself, and for the entire subtree below it. Since the column definition is part of the hierarchical structure of the document, you can define columns on level 1 that are general enough for all sublevels, and more specific columns further down, when you edit a deeper part of the tree.

7.5.1.2 Column attributes

A column definition sets the attributes of a column. The general definition looks like this:

```
%[width]property[(title)][{summary-type}]
```

Except for the percent sign and the property name, all items are optional. The individual parts have the following meaning:

width An integer specifying the width of the column in characters. If omitted, the width will be determined automatically.

property The property that should be edited in this column. Special properties representing meta data are allowed here as well (see Section 7.2 [Special properties], page 58)

title The header text for the column. If omitted, the property name is used.

{*summary-type*} The summary type. If specified, the column values for parent nodes are computed from the children. Supported summary types are:

{+}	Sum numbers in this column.
{+;%.1f}	Like '+', but format result with '%.1f'.
{$}	Currency, short for '+;%.2f'.
{:}	Sum times, HH:MM, plain numbers are hours.
{X}	Checkbox status, '[X]' if all children are '[X]'.
{X/}	Checkbox status, '[n/m]'.
{X%}	Checkbox status, '[n%]'.
{min}	Smallest number in column.
{max}	Largest number.
{mean}	Arithmetic mean of numbers.
{:min}	Smallest time value in column.
{:max}	Largest time value.
{:mean}	Arithmetic mean of time values.
{@min}	Minimum age (in days/hours/mins/seconds).
{@max}	Maximum age (in days/hours/mins/seconds).
@{mean}	Arithmetic mean of ages (in days/hours/mins/seconds).
{est+}	Add low-high estimates.

Be aware that you can only have one summary type for any property you include. Subsequent columns referencing the same property will all display the same summary information.

The `est+` summary type requires further explanation. It is used for combining estimates, expressed as low-high ranges. For example, instead of estimating a particular task will take 5 days, you might estimate it as 5-6 days if you're fairly confident you know how much work is required, or 1-10 days if you don't really know what needs to be done. Both ranges average at 5.5 days, but the first represents a more predictable delivery.

When combining a set of such estimates, simply adding the lows and highs produces an unrealistically wide result. Instead, `est+` adds the statistical mean and variance of the sub-tasks, generating a final estimate from the sum. For example, suppose you had ten tasks, each of which was estimated at 0.5 to 2 days of work. Straight addition produces an estimate of 5 to 20 days, representing what to expect if everything goes either extremely well or extremely poorly. In contrast, `est+` estimates the full job more realistically, at 10-15 days.

Here is an example for a complete columns definition, along with allowed values.

```
:COLUMNS:  %25ITEM %9Approved(Approved?){X} %Owner %11Status \
```

[1] Please note that the COLUMNS definition must be on a single line—it is wrapped here only because of formatting constraints.

```
                    %10Time_Estimate{:} %CLOCKSUM
  :Owner_ALL:    Tammy Mark Karl Lisa Don
  :Status_ALL:   "In progress" "Not started yet" "Finished" ""
  :Approved_ALL: "[ ]" "[X]"
```

The first column, '`%25ITEM`', means the first 25 characters of the item itself, i.e. of the headline. You probably always should start the column definition with the '`ITEM`' specifier. The other specifiers create columns '`Owner`' with a list of names as allowed values, for '`Status`' with four different possible values, and for a checkbox field '`Approved`'. When no width is given after the '`%`' character, the column will be exactly as wide as it needs to be in order to fully display all values. The '`Approved`' column does have a modified title ('`Approved?`', with a question mark). Summaries will be created for the '`Time_Estimate`' column by adding time duration expressions like HH:MM, and for the '`Approved`' column, by providing an '`[X]`' status if all children have been checked. The '`CLOCKSUM`' column is special, it lists the sum of CLOCK intervals in the subtree.

7.5.2 Using column view

Turning column view on and off

C-c C-x C-c `org-columns`

Turn on column view. If the cursor is before the first headline in the file, column view is turned on for the entire file, using the `#+COLUMNS` definition. If the cursor is somewhere inside the outline, this command searches the hierarchy, up from point, for a `:COLUMNS:` property that defines a format. When one is found, the column view table is established for the tree starting at the entry that contains the `:COLUMNS:` property. If no such property is found, the format is taken from the `#+COLUMNS` line or from the variable `org-columns-default-format`, and column view is established for the current entry and its subtree.

r `org-columns-redo`

Recreate the column view, to include recent changes made in the buffer.

g `org-columns-redo`

Same as *r*.

q `org-columns-quit`

Exit column view.

Editing values

left right up down

Move through the column view from field to field.

S-left/right

Switch to the next/previous allowed value of the field. For this, you have to have specified allowed values for a property.

1..9,0 Directly select the Nth allowed value, *0* selects the 10th value.

n `org-columns-next-allowed-value`

p `org-columns-previous-allowed-value`

Same as *S-left/right*

`e` `org-columns-edit-value`
: Edit the property at point. For the special properties, this will invoke the same interface that you normally use to change that property. For example, when editing a TAGS property, the tag completion or fast selection interface will pop up.

`C-c C-c` `org-columns-set-tags-or-toggle`
: When there is a checkbox at point, toggle it.

`v` `org-columns-show-value`
: View the full value of this property. This is useful if the width of the column is smaller than that of the value.

`a` `org-columns-edit-allowed`
: Edit the list of allowed values for this property. If the list is found in the hierarchy, the modified values is stored there. If no list is found, the new value is stored in the first entry that is part of the current column view.

Modifying the table structure

`<` `org-columns-narrow`
`>` `org-columns-widen`
: Make the column narrower/wider by one character.

`S-M-right` `org-columns-new`
: Insert a new column, to the left of the current column.

`S-M-left` `org-columns-delete`
: Delete the current column.

7.5.3 カラム表示の保存

Since column view is just an overlay over a buffer, it cannot be exported or printed directly. If you want to capture a column view, use a `columnview` dynamic block (see Section A.6 [Dynamic blocks], page 192). The frame of this block looks like this:

```
* The column view
#+BEGIN: columnview :hlines 1 :id "label"

#+END:
```

This dynamic block has the following parameters:

`:id`
: This is the most important parameter. Column view is a feature that is often localized to a certain (sub)tree, and the capture block might be at a different location in the file. To identify the tree whose view to capture, you can use 4 values:

  `local` use the tree in which the capture block is located
  `global` make a global view, including all headings in the file
  `"file:`*`path-to-file`*`"` run column view at the top of this file
  `"`*`ID`*`"` call column view in the tree that has an `:ID:` property with the value *label*. You can use *`M-x org-id-copy`* to create a globally unique ID for the current entry and copy it to the kill-ring.

`:hlines` When `t`, insert an hline after every line. When a number *N*, insert an hline before each headline with level `<=` *N*.

`:vlines` When set to `t`, force column groups to get vertical lines.

`:maxlevel`
When set to a number, don't capture entries below this level.

`:skip-empty-rows`
When set to `t`, skip rows where the only non-empty specifier of the column view is `ITEM`.

The following commands insert or update the dynamic block:

C-c C-x i `org-insert-columns-dblock`
Insert a dynamic block capturing a column view. You will be prompted for the scope or ID of the view.

C-c C-c or *C-c C-x C-u* `org-dblock-update`
Update dynamic block at point. The cursor needs to be in the `#+BEGIN` line of the dynamic block.

C-u C-c C-x C-u `org-update-all-dblocks`
Update all dynamic blocks (see Section A.6 [Dynamic blocks], page 192). This is useful if you have several clock table blocks, column-capturing blocks or other dynamic blocks in a buffer.

You can add formulas to the column view table and you may add plotting instructions in front of the table—these will survive an update of the block. If there is a `#+TBLFM:` after the table, the table will actually be recalculated automatically after an update.

An alternative way to capture and process property values into a table is provided by Eric Schulte's '`org-collector.el`' which is a contributed package[2]. It provides a general API to collect properties from entries in a certain scope, and arbitrary Lisp expressions to process these values before inserting them into a table or a dynamic block.

7.6 The Property API

There is a full API for accessing and changing properties. This API can be used by Emacs Lisp programs to work with properties and to implement features based on them. For more information see Section A.9 [Using the property API], page 196.

[2] Contributed packages are not part of Emacs, but are distributed with the main distribution of Org (visit `http://orgmode.org`).

8 日付と時刻

To assist project planning, TODO items can be labeled with a date and/or a time. The specially formatted string carrying the date and time information is called a *timestamp* in Org-mode. This may be a little confusing because timestamp is often used as indicating when something was created or last changed. However, in Org-mode this term is used in a much wider sense.

8.1 Timestamps, deadlines, and scheduling

A timestamp is a specification of a date (possibly with a time or a range of times) in a special format, either '`<2003-09-16 Tue>`' or '`<2003-09-16 Tue 09:39>`' or '`<2003-09-16 Tue 12:00-12:30>`'[1]. A timestamp can appear anywhere in the headline or body of an Org tree entry. Its presence causes entries to be shown on specific dates in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 91). We distinguish:

Plain timestamp; Event; Appointment

A simple timestamp just assigns a date/time to an item. This is just like writing down an appointment or event in a paper agenda. In the timeline and agenda displays, the headline of an entry associated with a plain timestamp will be shown exactly on that date.

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

Timestamp with repeater interval

A timestamp may contain a *repeater interval*, indicating that it applies not only on the given date, but again and again after a certain interval of N days (d), weeks (w), months (m), or years (y). The following will show up in the agenda every Wednesday:

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

Diary-style sexp entries

For more complex date specifications, Org-mode supports using the special sexp diary entries implemented in the Emacs calendar/diary package. For example

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

Time/Date range

Two timestamps connected by '`--`' denote a range. The headline will be shown on the first and last day of the range, and on any dates that are displayed and fall in the range. Here is an example:

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

Inactive timestamp

Just like a plain timestamp, but with square brackets instead of angular ones. These timestamps are inactive in the sense that they do *not* trigger an entry to show up in the agenda.

[1] This is inspired by the standard ISO 8601 date/time format. To use an alternative format, see Section 8.2.2 [Custom time format], page 68.

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

8.2 Creating timestamps

For Org-mode to recognize timestamps, they need to be in the specific format. All commands listed below produce timestamps in the correct format.

C-c . `org-time-stamp`
: Prompt for a date and insert a corresponding timestamp. When the cursor is at an existing timestamp in the buffer, the command is used to modify this timestamp instead of inserting a new one. When this command is used twice in succession, a time range is inserted.

C-c ! `org-time-stamp-inactive`
: Like *C-c .*, but insert an inactive timestamp that will not cause an agenda entry.

C-u C-c .
C-u C-c !
: Like *C-c .* and *C-c !*, but use the alternative format which contains date and time. The default time can be rounded to multiples of 5 minutes, see the option `org-time-stamp-rounding-minutes`.

C-c < `org-date-from-calendar`
: Insert a timestamp corresponding to the cursor date in the Calendar.

C-c > `org-goto-calendar`
: Access the Emacs calendar for the current date. If there is a timestamp in the current line, go to the corresponding date instead.

C-c C-o `org-open-at-point`
: Access the agenda for the date given by the timestamp or -range at point (see Section 10.3.1 [Weekly/daily agenda], page 91).

S-left `org-timestamp-down-day`
S-right `org-timestamp-up-day`
: Change date at cursor by one day. These key bindings conflict with shift-selection and related modes (see Section 15.10.2 [Conflicts], page 184).

S-up `org-timestamp-up`
S-down `org-timestamp-down-down`
: Change the item under the cursor in a timestamp. The cursor can be on a year, month, day, hour or minute. When the timestamp contains a time range like '`15:30-16:30`', modifying the first time will also shift the second, shifting the time block with constant length. To change the length, modify the second time. Note that if the cursor is in a headline and not at a timestamp, these same keys modify the priority of an item. (see Section 5.4 [Priorities], page 49). The key bindings also conflict with shift-selection and related modes (see Section 15.10.2 [Conflicts], page 184).

C-c C-y `org-evaluate-time-range`
: Evaluate a time range by computing the difference between start and end. With a prefix argument, insert result after the time range (in a table: into the following column).

8.2.1 The date/time prompt

When Org-mode prompts for a date/time, the default is shown in default date/time format, and the prompt therefore seems to ask for a specific format. But it will in fact accept any string containing some date and/or time information, and it is really smart about interpreting your input. You can, for example, use `C-y` to paste a (possibly multi-line) string copied from an email message. Org-mode will find whatever information is in there and derive anything you have not specified from the *default date and time*. The default is usually the current date and time, but when modifying an existing timestamp, or when entering the second stamp of a range, it is taken from the stamp in the buffer. When filling in information, Org-mode assumes that most of the time you will want to enter a date in the future: if you omit the month/year and the given day/month is *before* today, it will assume that you mean a future date[2]. If the date has been automatically shifted into the future, the time prompt will show this with '`(=>F).`'

For example, let's assume that today is **June 13, 2006**. Here is how various inputs will be interpreted, the items filled in by Org-mode are in **bold**.

```
3-2-5         ⇒ 2003-02-05
2/5/3         ⇒ 2003-02-05
14            ⇒ 2006-06-14
12            ⇒ 2006-07-12
2/5           ⇒ 2007-02-05
Fri           ⇒ nearest Friday (default date or later)
sep 15        ⇒ 2006-09-15
feb 15        ⇒ 2007-02-15
sep 12 9      ⇒ 2009-09-12
12:45         ⇒ 2006-06-13 12:45
22 sept 0:34  ⇒ 2006-09-22 0:34
w4            ⇒ ISO week for of the current year 2006
2012 w4 fri   ⇒ Friday of ISO week 4 in 2012
2012-w04-5    ⇒ Same as above
```

Furthermore you can specify a relative date by giving, as the *first* thing in the input: a plus/minus sign, a number and a letter ([dwmy]) to indicate change in days, weeks, months, or years. With a single plus or minus, the date is always relative to today. With a double plus or minus, it is relative to the default date. If instead of a single letter, you use the abbreviation of day name, the date will be the Nth such day. e.g.

```
+0            ⇒ today
.             ⇒ today
+4d           ⇒ four days from today
+4            ⇒ same as above
+2w           ⇒ two weeks from today
++5           ⇒ five days from default date
+2tue         ⇒ second Tuesday from now.
```

[2] See the variable `org-read-date-prefer-future`. You may set that variable to the symbol `time` to even make a time before now shift the date to tomorrow.

The function understands English month and weekday abbreviations. If you want to use unabbreviated names and/or other languages, configure the variables `parse-time-months` and `parse-time-weekdays`.

You can specify a time range by giving start and end times or by giving a start time and a duration (in HH:MM format). Use '-' or '-{}-' as the separator in the former case and use '+' as the separator in the latter case. E.g.

```
11am-1:15pm     ⇒ 11:00-13:15
11am--1:15pm    ⇒ same as above
11am+2:15       ⇒ same as above
```

Parallel to the minibuffer prompt, a calendar is popped up[3]. When you exit the date prompt, either by clicking on a date in the calendar, or by pressing `RET`, the date selected in the calendar will be combined with the information entered at the prompt. You can control the calendar fully from the minibuffer:

`RET`	Choose date at cursor in calendar.
`mouse-1`	Select date by clicking on it.
`S-right/left`	One day forward/backward.
`S-down/up`	One week forward/backward.
`M-S-right/left`	One month forward/backward.
`> / <`	Scroll calendar forward/backward by one month.
`M-v / C-v`	Scroll calendar forward/backward by 3 months.

The actions of the date/time prompt may seem complex, but I assure you they will grow on you, and you will start getting annoyed by pretty much any other way of entering a date/time out there. To help you understand what is going on, the current interpretation of your input will be displayed live in the minibuffer[4].

8.2.2 Custom time format

Org-mode uses the standard ISO notation for dates and times as it is defined in ISO 8601. If you cannot get used to this and require another representation of date and time to keep you happy, you can get it by customizing the variables `org-display-custom-times` and `org-time-stamp-custom-formats`.

C-c C-x C-t `org-toggle-time-stamp-overlays`

Toggle the display of custom formats for dates and times.

Org-mode needs the default format for scanning, so the custom date/time format does not *replace* the default format—instead it is put *over* the default format using text properties. This has the following consequences:

- You cannot place the cursor onto a timestamp anymore, only before or after.
- The *S-up/down* keys can no longer be used to adjust each component of a timestamp. If the cursor is at the beginning of the stamp, *S-up/down* will change the stamp by one day, just like *S-left/right*. At the end of the stamp, the time will be changed by one minute.
- If the timestamp contains a range of clock times or a repeater, these will not be overlaid, but remain in the buffer as they were.

[3] If you don't need/want the calendar, configure the variable `org-popup-calendar-for-date-prompt`.

[4] If you find this distracting, turn the display of with `org-read-date-display-live`.

- When you delete a timestamp character-by-character, it will only disappear from the buffer after *all* (invisible) characters belonging to the ISO timestamp have been removed.
- If the custom timestamp format is longer than the default and you are using dates in tables, table alignment will be messed up. If the custom format is shorter, things do work as expected.

8.3 Deadlines and scheduling

A timestamp may be preceded by special keywords to facilitate planning:

DEADLINE

Meaning: the task (most likely a TODO item, though not necessarily) is supposed to be finished on that date.

On the deadline date, the task will be listed in the agenda. In addition, the agenda for *today* will carry a warning about the approaching or missed deadline, starting `org-deadline-warning-days` before the due date, and continuing until the entry is marked DONE. An example:

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

You can specify a different lead time for warnings for a specific deadlines using the following syntax. Here is an example with a warning period of 5 days `DEADLINE: <2004-02-29 Sun -5d>`.

SCHEDULED

Meaning: you are planning to start working on that task on the given date.

The headline will be listed under the given date[5]. In addition, a reminder that the scheduled date has passed will be present in the compilation for *today*, until the entry is marked DONE, i.e. the task will automatically be forwarded until completed.

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

Important: Scheduling an item in Org-mode should *not* be understood in the same way that we understand *scheduling a meeting*. Setting a date for a meeting is just a simple appointment, you should mark this entry with a simple plain timestamp, to get this item shown on the date where it applies. This is a frequent misunderstanding by Org users. In Org-mode, *scheduling* means setting a date when you want to start working on an action item.

You may use timestamps with repeaters in scheduling and deadline entries. Org-mode will issue early and late warnings based on the assumption that the timestamp represents the *nearest instance* of the repeater. However, the use of diary sexp entries like `<%%(diary-float t 42)>` in scheduling and deadline timestamps is limited. Org-mode does not know enough about the internals of each sexp function to issue early and late warnings. However, it will show the item on each day where the sexp entry matches.

[5] It will still be listed on that date after it has been marked DONE. If you don't like this, set the variable `org-agenda-skip-scheduled-if-done`.

8.3.1 Inserting deadlines or schedules

The following commands allow you to quickly insert[6] a deadline or to schedule an item:

C-c C-d `org-deadline`

Insert '`DEADLINE`' keyword along with a stamp. The insertion will happen in the line directly following the headline. When called with a prefix arg, an existing deadline will be removed from the entry. Depending on the variable `org-log-redeadline`[7], a note will be taken when changing an existing deadline.

C-c C-s `org-schedule`

Insert '`SCHEDULED`' keyword along with a stamp. The insertion will happen in the line directly following the headline. Any CLOSED timestamp will be removed. When called with a prefix argument, remove the scheduling date from the entry. Depending on the variable `org-log-reschedule`[8], a note will be taken when changing an existing scheduling time.

C-c C-x C-k `org-mark-entry-for-agenda-action`

Mark the current entry for agenda action. After you have marked the entry like this, you can open the agenda or the calendar to find an appropriate date. With the cursor on the selected date, press *k s* or *k d* to schedule the marked item.

C-c / d `org-check-deadlines`

Create a sparse tree with all deadlines that are either past-due, or which will become due within `org-deadline-warning-days`. With *C-u* prefix, show all deadlines in the file. With a numeric prefix, check that many days. For example, *C-1 C-c / d* shows all deadlines due tomorrow.

C-c / b `org-check-before-date`

Sparse tree for deadlines and scheduled items before a given date.

C-c / a `org-check-after-date`

Sparse tree for deadlines and scheduled items after a given date.

8.3.2 Repeated tasks

Some tasks need to be repeated again and again. Org-mode helps to organize such tasks using a so-called repeater in a DEADLINE, SCHEDULED, or plain timestamp. In the following example

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

the `+1m` is a repeater; the intended interpretation is that the task has a deadline on <2005-10-01> and repeats itself every (one) month starting from that time. If you need both a repeater and a special warning period in a deadline entry, the repeater should come first and the warning period last: `DEADLINE: <2005-10-01 Sat +1m -3d>`.

---

[6] The '`SCHEDULED`' and '`DEADLINE`' dates are inserted on the line right below the headline. Don't put any text between this line and the headline.

[7] with corresponding `#+STARTUP` keywords `logredeadline`, `lognoteredeadline`, and `nologredeadline`

[8] with corresponding `#+STARTUP` keywords `logreschedule`, `lognotereschedule`, and `nologreschedule`

Deadlines and scheduled items produce entries in the agenda when they are over-due, so it is important to be able to mark such an entry as completed once you have done so. When you mark a DEADLINE or a SCHEDULE with the TODO keyword DONE, it will no longer produce entries in the agenda. The problem with this is, however, that then also the *next* instance of the repeated entry will not be active. Org-mode deals with this in the following way: When you try to mark such an entry DONE (using `C-c C-t`), it will shift the base date of the repeating timestamp by the repeater interval, and immediately set the entry state back to TODO[9]. In the example above, setting the state to DONE would actually switch the date like this:

```
** TODO Pay the rent
   DEADLINE: <2005-11-01 Tue +1m>
```

A timestamp[10] will be added under the deadline, to keep a record that you actually acted on the previous instance of this deadline.

As a consequence of shifting the base date, this entry will no longer be visible in the agenda when checking past dates, but all future instances will be visible.

With the '`+1m`' cookie, the date shift will always be exactly one month. So if you have not paid the rent for three months, marking this entry DONE will still keep it as an overdue deadline. Depending on the task, this may not be the best way to handle it. For example, if you forgot to call your father for 3 weeks, it does not make sense to call him 3 times in a single day to make up for it. Finally, there are tasks like changing batteries which should always repeat a certain time *after* the last time you did it. For these tasks, Org-mode has special repeaters '`++`' and '`.+`'. For example:

```
** TODO Call Father
   DEADLINE: <2008-02-10 Sun ++1w>
   Marking this DONE will shift the date by at least one week,
   but also by as many weeks as it takes to get this date into
   the future.  However, it stays on a Sunday, even if you called
   and marked it done on Saturday.
** TODO Check the batteries in the smoke detectors
   DEADLINE: <2005-11-01 Tue .+1m>
   Marking this DONE will shift the date to one month after
   today.
```

You may have both scheduling and deadline information for a specific task—just make sure that the repeater intervals on both are the same.

An alternative to using a repeater is to create a number of copies of a task subtree, with dates shifted in each copy. The command `C-c C-x c` was created for this purpose, it is described in Section 2.5 [Structure editing], page 8.

---

[9] In fact, the target state is taken from, in this sequence, the `REPEAT_TO_STATE` property or the variable `org-todo-repeat-to-state`. If neither of these is specified, the target state defaults to the first state of the TODO state sequence.

[10] You can change this using the option `org-log-repeat`, or the `#+STARTUP` options `logrepeat`, `lognoterepeat`, and `nologrepeat`. With `lognoterepeat`, you will also be prompted for a note.

8.4 Clocking work time

Org-mode allows you to clock the time you spend on specific tasks in a project. When you start working on an item, you can start the clock. When you stop working on that task, or when you mark the task done, the clock is stopped and the corresponding time interval is recorded. It also computes the total time spent on each subtree of a project. And it remembers a history or tasks recently clocked, to that you can jump quickly between a number of tasks absorbing your time.

To save the clock history across Emacs sessions, use

```
(setq org-clock-persist 'history)
(org-clock-persistence-insinuate)
```

When you clock into a new task after resuming Emacs, the incomplete clock[11] will be found (see Section 8.4.3 [Resolving idle time], page 75) and you will be prompted about what to do with it.

8.4.1 Clocking commands

C-c C-x C-i `org-clock-in`

Start the clock on the current item (clock-in). This inserts the CLOCK keyword together with a timestamp. If this is not the first clocking of this item, the multiple CLOCK lines will be wrapped into a `:LOGBOOK:` drawer (see also the variable `org-clock-into-drawer`). When called with a *C-u* prefix argument, select the task from a list of recently clocked tasks. With two *C-u C-u* prefixes, clock into the task at point and mark it as the default task. The default task will always be available when selecting a clocking task, with letter *d*.

While the clock is running, the current clocking time is shown in the mode line, along with the title of the task. The clock time shown will be all time ever clocked for this task and its children. If the task has an effort estimate (see Section 8.5 [Effort estimates], page 76), the mode line displays the current clocking time against it[12] If the task is a repeating one (see Section 8.3.2 [Repeated tasks], page 70), only the time since the last reset of the task[13] will be shown. More control over what time is shown can be exercised with the `CLOCK_MODELINE_TOTAL` property. It may have the values `current` to show only the current clocking instance, `today` to show all time clocked on this tasks today (see also the variable `org-extend-today-until`), `all` to include all time, or `auto` which is the default[14].

Clicking with *mouse-1* onto the mode line entry will pop up a menu with clocking options.

C-c C-x C-o `org-clock-out`

Stop the clock (clock-out). This inserts another timestamp at the same location where the clock was last started. It also directly computes the resulting time in

[11] To resume the clock under the assumption that you have worked on this task while outside Emacs, use `(setq org-clock-persist t)`.

[12] To add an effort estimate "on the fly", hook a function doing this to `org-clock-in-prepare-hook`.

[13] as recorded by the `LAST_REPEAT` property

[14] See also the variable `org-clock-modeline-total`.

inserts it after the time range as '=> HH:MM'. See the variable org-log-note-clock-out for the possibility to record an additional note together with the clock-out timestamp[15].

C-c C-x C-e org-clock-modify-effort-estimate
Update the effort estimate for the current clock task.

C-c C-c or C-c C-y org-evaluate-time-range
Recompute the time interval after changing one of the timestamps. This is only necessary if you edit the timestamps directly. If you change them with S-cursor keys, the update is automatic.

C-c C-t org-todo
Changing the TODO state of an item to DONE automatically stops the clock if it is running in this same item.

C-c C-x C-x org-clock-cancel
Cancel the current clock. This is useful if a clock was started by mistake, or if you ended up working on something else.

C-c C-x C-j org-clock-goto
Jump to the headline of the currently clocked in task. With a C-u prefix arg, select the target task from a list of recently clocked tasks.

C-c C-x C-d org-clock-display
Display time summaries for each subtree in the current buffer. This puts overlays at the end of each headline, showing the total time recorded under that heading, including the time of any subheadings. You can use visibility cycling to study the tree, but the overlays disappear when you change the buffer (see variable org-remove-highlights-with-change) or press C-c C-c.

The l key may be used in the timeline (see Section 10.3.4 [Timeline], page 96) and in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 91) to show which tasks have been worked on or closed during a day.

8.4.2 The clock table

Org mode can produce quite complex reports based on the time clocking information. Such a report is called a *clock table*, because it is formatted as one or several Org tables.

C-c C-x C-r org-clock-report
Insert a dynamic block (see Section A.6 [Dynamic blocks], page 192) containing a clock report as an Org-mode table into the current file. When the cursor is at an existing clock table, just update it. When called with a prefix argument, jump to the first clock report in the current document and update it.

C-c C-c or C-c C-x C-u org-dblock-update
Update dynamic block at point. The cursor needs to be in the #+BEGIN line of the dynamic block.

C-u C-c C-x C-u
Update all dynamic blocks (see Section A.6 [Dynamic blocks], page 192). This is useful if you have several clock table blocks in a buffer.

[15] The corresponding in-buffer setting is: #+STARTUP: lognoteclock-out

S-left
S-right `org-clocktable-try-shift`
: Shift the current `:block` interval and update the table. The cursor needs to be in the `#+BEGIN: clocktable` line for this command. If `:block` is `today`, it will be shifted to `today-1` etc.

Here is an example of the frame for a clock table as it is inserted into the buffer with the *C-c C-x C-r* command:

```
#+BEGIN: clocktable :maxlevel 2 :emphasize nil :scope file
#+END: clocktable
```

The 'BEGIN' line and specify a number of options to define the scope, structure, and formatting of the report. Defaults for all these options can be configured in the variable `org-clocktable-defaults`.

First there are options that determine which clock entries are to be selected:

`:maxlevel` Maximum level depth to which times are listed in the table. Clocks at deeper levels will be summed into the upper level.

`:scope` The scope to consider. This can be any of the following:

`nil`	the current buffer or narrowed region
`file`	the full current buffer
`subtree`	the subtree where the clocktable is located
`treeN`	the surrounding level *N* tree, for example `tree3`
`tree`	the surrounding level 1 tree
`agenda`	all agenda files
`("file"..)`	scan these files
`file-with-archives`	current file and its archives
`agenda-with-archives`	all agenda files, including archives

`:block` The time block to consider. This block is specified either absolute, or relative to the current time and may be any of these formats:

`2007-12-31`	New year eve 2007
`2007-12`	December 2007
`2007-W50`	ISO-week 50 in 2007
`2007-Q2`	2nd quarter in 2007
`2007`	the year 2007
`today, yesterday, today-N`	a relative day
`thisweek, lastweek, thisweek-N`	a relative week
`thismonth, lastmonth, thismonth-N`	a relative month
`thisyear, lastyear, thisyear-N`	a relative year

Use *S-left/right* keys to shift the time interval.

`:tstart` A time string specifying when to start considering times.

`:tend` A time string specifying when to stop considering times.

`:step` `week` or `day`, to split the table into chunks. To use this, `:block` or `:tstart`, `:tend` are needed.

`:stepskip0` Do not show steps that have zero time.

`:fileskip0` Do not show table sections from files which did not contribute.

`:tags` A tags match to select entries that should contribute.

Then there are options which determine the formatting of the table. There options are interpreted by the function `org-clocktable-write-default`, but you can specify your own function using the `:formatter` parameter.

`:emphasize`	When `t`, emphasize level one and level two items.
`:lang`	Language[16] to use for descriptive cells like `"Task"`.
`:link`	Link the item headlines in the table to their origins.
`:narrow`	An integer to limit the width of the headline column in the org table. If you write it like '`50!`', then the headline will also be shortened in export.
`:indent`	Indent each headline field according to its level.
`:tcolumns`	Number of columns to be used for times. If this is smaller than `:maxlevel`, lower levels will be lumped into one column.
`:level`	Should a level number column be included?
`:compact`	Abbreviation for `:level nil :indent t :narrow 40! :tcolumns 1` All are overwritten except if there is an explicit `:narrow`
`:timestamp`	A timestamp for the entry, when available. Look for SCHEDULED, DEADLINE, TIMESTAMP and TIMESTAMP_IA, in this order.
`:formula`	Content of a `#+TBLFM` line to be added and evaluated. As a special case, '`:formula %`' adds a column with % time. If you do not specify a formula here, any existing formula below the clock table will survive updates and be evaluated.
`:formatter`	A function to format clock data and insert it into the buffer.

To get a clock summary of the current level 1 tree, for the current day, you could write

```
#+BEGIN: clocktable :maxlevel 2 :block today :scope tree1 :link t
#+END: clocktable
```

and to use a specific time range you could write[17]

```
#+BEGIN: clocktable :tstart "<2006-08-10 Thu 10:00>"
                    :tend "<2006-08-10 Thu 12:00>"
#+END: clocktable
```

A summary of the current subtree with % times would be

```
#+BEGIN: clocktable :scope subtree :link t :formula %
#+END: clocktable
```

A horizontally compact representation of everything clocked during last week would be

```
#+BEGIN: clocktable :scope agenda :block lastweek :compact t
#+END: clocktable
```

8.4.3 Resolving idle time

If you clock in on a work item, and then walk away from your computer—perhaps to take a phone call—you often need to "resolve" the time you were away by either subtracting it from the current clock, or applying it to another one.

[16] Language terms can be set through the variable `org-clock-clocktable-language-setup`.

[17] Note that all parameters must be specified in a single line—the line is broken here only to fit it into the manual.

By customizing the variable `org-clock-idle-time` to some integer, such as 10 or 15, Emacs can alert you when you get back to your computer after being idle for that many minutes[18], and ask what you want to do with the idle time. There will be a question waiting for you when you get back, indicating how much idle time has passed (constantly updated with the current amount), as well as a set of choices to correct the discrepancy:

`k` To keep some or all of the minutes and stay clocked in, press `k`. Org will ask how many of the minutes to keep. Press `RET` to keep them all, effectively changing nothing, or enter a number to keep that many minutes.

`K` If you use the shift key and press `K`, it will keep however many minutes you request and then immediately clock out of that task. If you keep all of the minutes, this is the same as just clocking out of the current task.

`s` To keep none of the minutes, use `s` to subtract all the away time from the clock, and then check back in from the moment you returned.

`S` To keep none of the minutes and just clock out at the start of the away time, use the shift key and press `S`. Remember that using shift will always leave you clocked out, no matter which option you choose.

`C` To cancel the clock altogether, use `C`. Note that if instead of canceling you subtract the away time, and the resulting clock amount is less than a minute, the clock will still be canceled rather than clutter up the log with an empty entry.

What if you subtracted those away minutes from the current clock, and now want to apply them to a new clock? Simply clock in to any task immediately after the subtraction. Org will notice that you have subtracted time "on the books", so to speak, and will ask if you want to apply those minutes to the next task you clock in on.

There is one other instance when this clock resolution magic occurs. Say you were clocked in and hacking away, and suddenly your cat chased a mouse who scared a hamster that crashed into your UPS's power button! You suddenly lose all your buffers, but thanks to auto-save you still have your recent Org mode changes, including your last clock in.

If you restart Emacs and clock into any task, Org will notice that you have a dangling clock which was never clocked out from your last session. Using that clock's starting time as the beginning of the unaccounted-for period, Org will ask how you want to resolve that time. The logic and behavior is identical to dealing with away time due to idleness; it's just happening due to a recovery event rather than a set amount of idle time.

You can also check all the files visited by your Org agenda for dangling clocks at any time using `M-x org-resolve-clocks`.

8.5 Effort estimates

If you want to plan your work in a very detailed way, or if you need to produce offers with quotations of the estimated work effort, you may want to assign effort estimates to entries.

[18] On computers using Mac OS X, idleness is based on actual user idleness, not just Emacs' idle time. For X11, you can install a utility program '`x11idle.c`', available in the UTILITIES directory of the Org git distribution, to get the same general treatment of idleness. On other systems, idle time refers to Emacs idle time only.

If you are also clocking your work, you may later want to compare the planned effort with the actual working time, a great way to improve planning estimates. Effort estimates are stored in a special property '`Effort`'[19]. You can set the effort for an entry with the following commands:

C-c C-x e `org-set-effort`
: Set the effort estimate for the current entry. With a numeric prefix argument, set it to the Nth allowed value (see below). This command is also accessible from the agenda with the `e` key.

C-c C-x C-e `org-clock-modify-effort-estimate`
: Modify the effort estimate of the item currently being clocked.

Clearly the best way to work with effort estimates is through column view (see Section 7.5 [Column view], page 60). You should start by setting up discrete values for effort estimates, and a `COLUMNS` format that displays these values together with clock sums (if you want to clock your time). For a specific buffer you can use

```
#+PROPERTY: Effort_ALL 0 0:10 0:30 1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00
#+COLUMNS: %40ITEM(Task) %17Effort(Estimated Effort){:} %CLOCKSUM
```

or, even better, you can set up these values globally by customizing the variables `org-global-properties` and `org-columns-default-format`. In particular if you want to use this setup also in the agenda, a global setup may be advised.

The way to assign estimates to individual items is then to switch to column mode, and to use *S-right* and *S-left* to change the value. The values you enter will immediately be summed up in the hierarchy. In the column next to it, any clocked time will be displayed.

If you switch to column view in the daily/weekly agenda, the effort column will summarize the estimated work effort for each day[20], and you can use this to find space in your schedule. To get an overview of the entire part of the day that is committed, you can set the option `org-agenda-columns-add-appointments-to-effort-sum`. The appointments on a day that take place over a specified time interval will then also be added to the load estimate of the day.

Effort estimates can be used in secondary agenda filtering that is triggered with the `/` key in the agenda (see Section 10.5 [Agenda commands], page 100). If you have these estimates defined consistently, two or three key presses will narrow down the list to stuff that fits into an available time slot.

8.6 Taking notes with a relative timer

When taking notes during, for example, a meeting or a video viewing, it can be useful to have access to times relative to a starting time. Org provides such a relative timer and make it easy to create timed notes.

C-c C-x . `org-timer`
: Insert a relative time into the buffer. The first time you use this, the timer will be started. When called with a prefix argument, the timer is restarted.

[19] You may change the property being used with the variable `org-effort-property`.

[20] Please note the pitfalls of summing hierarchical data in a flat list (see Section 10.8 [Agenda column view], page 114).

C-c C-x - `org-timer-item`
: Insert a description list item with the current relative time. With a prefix argument, first reset the timer to 0.

M-RET `org-insert-heading`
: Once the timer list is started, you can also use *M-RET* to insert new timer items.

C-c C-x , Pause the timer, or continue it if it is already paused (`org-timer-pause-or-continue`).

C-u C-c C-x ,
: Stop the timer. After this, you can only start a new timer, not continue the old one. This command also removes the timer from the mode line.

C-c C-x 0 `org-timer-start`
: Reset the timer without inserting anything into the buffer. By default, the timer is reset to 0. When called with a *C-u* prefix, reset the timer to specific starting offset. The user is prompted for the offset, with a default taken from a timer string at point, if any, So this can be used to restart taking notes after a break in the process. When called with a double prefix argument *C-u C-u*, change all timer strings in the active region by a certain amount. This can be used to fix timer strings if the timer was not started at exactly the right moment.

8.7 カウントダウンタイマ

Calling `org-timer-set-timer` from an Org-mode buffer runs a countdown timer. Use `;` from agenda buffers, `C-c C-x ;` everwhere else.

`org-timer-set-timer` prompts the user for a duration and displays a countdown timer in the modeline. `org-timer-default-timer` sets the default countdown value. Giving a prefix numeric argument overrides this default value.

9 Capture - Refile - Archive

An important part of any organization system is the ability to quickly capture new ideas and tasks, and to associate reference material with them. Org does this using a process called *capture*. It also can store files related to a task (*attachments*) in a special directory. Once in the system, tasks and projects need to be moved around. Moving completed project trees to an archive file keeps the system compact and fast.

9.1 Capture

Org's method for capturing new items is heavily inspired by John Wiegley excellent remember package. Up to version 6.36 Org used a special setup for '`remember.el`'. '`org-remember.el`' is still part of Org-mode for backward compatibility with existing setups. You can find the documentation for org-remember at `http://orgmode.org/org-remember.pdf`.

The new capturing setup described here is preferred and should be used by new users. To convert your `org-remember-templates`, run the command

M-x org-capture-import-remember-templates RET

and then customize the new variable with *M-x customize-variable org-capture-templates*, check the result, and save the customization. You can then use both remember and capture until you are familiar with the new mechanism.

Capture lets you quickly store notes with little interruption of your work flow. The basic process of capturing is very similar to remember, but Org does enhance it with templates and more.

9.1.1 Setting up capture

The following customization sets a default target file for notes, and defines a global key[1] for capturing new material.

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

9.1.2 Using capture

C-c c `org-capture`

Call the command `org-capture`. Note that this keybinding is global and not active by default - you need to install it. If you have templates defined see Section 9.1.3 [Capture templates], page 80, it will offer these templates for selection or use a new Org outline node as the default template. It will insert the template into the target file and switch to an indirect buffer narrowed to this new node. You may then insert the information you want.

C-c C-c `org-capture-finalize`

Once you have finished entering information into the capture buffer, *C-c C-c* will return you to the window configuration before the capture process, so that you can resume your work without further distraction. When called with a prefix arg, finalize and then jump to the captured item.

[1] Please select your own key, *C-c c* is only a suggestion.

`C-c C-w` `org-capture-refile`
: Finalize the capture process by refiling (see Section 9.5 [Refiling notes], page 86) the note to a different place. Please realize that this is a normal refiling command that will be executed—so the cursor position at the moment you run this command is important. If you have inserted a tree with a parent and children, first move the cursor back to the parent. Any prefix argument given to this command will be passed on to the `org-refile` command.

`C-c C-k` `org-capture-kill`
: Abort the capture process and return to the previous state.

You can also call `org-capture` in a special way from the agenda, using the `k c` key combination. With this access, any timestamps inserted by the selected capture template will default to the cursor date in the agenda, rather than to the current date.

To find the locations of the last stored capture, use `org-capture` with prefix commands:

`C-u C-c c`
: Visit the target location of a cpature template. You get to select the template in the usual way.

`C-u C-u C-c c`
: Visit the last stored capture item in its buffer.

9.1.3 Capture templates

You can use templates for different types of capture items, and for different target locations. The easiest way to create such templates is through the customize interface.

`C-c c C` Customize the variable `org-capture-templates`.

Before we give the formal description of template definitions, let's look at an example. Say you would like to use one template to create general TODO entries, and you want to put these entries under the heading '`Tasks`' in your file '`~/org/gtd.org`'. Also, a date tree in the file '`journal.org`' should capture journal entries. A possible configuration would look like:

```
(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")))
```

If you then press `C-c c t`, Org will prepare the template for you like this:

```
* TODO
  [[file:link to where you initiated capture]]
```

During expansion of the template, `%a` has been replaced by a link to the location from where you called the capture command. This can be extremely useful for deriving tasks from emails, for example. You fill in the task definition, press `C-c C-c` and Org returns you to the same place where you started the capture process.

To define special keys to capture to a particular template without going through the interactive template selection, you can create your key binding like this:

```
(define-key global-map "\C-cx"
   (lambda () (interactive) (org-capture nil "x")))
```

9.1.3.1 Template elements

Now lets look at the elements of a template definition. Each entry in `org-capture-templates` is a list with the following items:

keys The keys that will select the template, as a string, characters only, for example `"a"` for a template to be selected with a single key, or `"bt"` for selection with two keys. When using several keys, keys using the same prefix key must be sequential in the list and preceded by a 2-element entry explaining the prefix key, for example

```
("b" "Templates for marking stuff to buy")
```

If you do not define a template for the *C* key, this key will be used to open the customize buffer for this complex variable.

description
A short string describing the template, which will be shown during selection.

type The type of entry, a symbol. Valid values are:

`entry` An Org-mode node, with a headline. Will be filed as the child of the target entry or as a top-level entry. The target file should be an Org-mode file.

`item` A plain list item, placed in the first plain list at the target location. Again the target file should be an Org file.

`checkitem`
A checkbox item. This only differs from the plain list item by the default template.

`table-line`
a new line in the first table at the target location. Where exactly the line will be inserted depends on the properties `:prepend` and `:table-line-pos` (see below).

`plain` Text to be inserted as it is.

target Specification of where the captured item should be placed. In Org-mode files, targets usually define a node. Entries will become children of this node. Other types will be added to the table or list in the body of this node. Most target specifications contain a file name. If that file name is the empty string, it defaults to `org-default-notes-file`. A file can also be given as a variable, function, or Emacs Lisp form.

Valid values are:

`(file "path/to/file")`
Text will be placed at the beginning or end of that file.

`(id "id of existing org entry")`
Filing as child of this entry, or in the body of the entry.

`(file+headline "path/to/file" "node headline")`
Fast configuration if the target heading is unique in the file.

`(file+olp "path/to/file" "Level 1 heading" "Level 2" ...)`
: For non-unique headings, the full path is safer.

`(file+regexp "path/to/file" "regexp to find location")`
: Use a regular expression to position the cursor.

`(file+datetree "path/to/file")`
: Will create a heading in a date tree for today's date.

`(file+datetree+prompt "path/to/file")`
: Will create a heading in a date tree, but will prompt for the date.

`(file+function "path/to/file" function-finding-location)`
: A function to find the right location in the file.

`(clock)` File to the entry that is currently being clocked.

`(function function-finding-location)`
: Most general way, write your own function to find both file and location.

template The template for creating the capture item. If you leave this empty, an appropriate default template will be used. Otherwise this is a string with escape codes, which will be replaced depending on time and context of the capture call. The string with escapes may be loaded from a template file, using the special syntax `(file "path/to/template")`. See below for more details.

properties The rest of the entry is a property list of additional options. Recognized properties are:

`:prepend` Normally new captured information will be appended at the target location (last child, last table line, last list item...). Setting this property will change that.

`:immediate-finish`
: When set, do not offer to edit the information, just file it away immediately. This makes sense if the template only needs information that can be added automatically.

`:empty-lines`
: Set this to the number of lines to insert before and after the new item. Default 0, only common other value is 1.

`:clock-in`
: Start the clock in this item.

`:clock-keep`
: Keep the clock running when filing the captured entry.

`:clock-resume`
: If starting the capture interrupted a clock, restart that clock when finished with the capture. Note that `:clock-keep` has precedence over `:clock-resume`. When setting both to `t`, the current clock will run and the previous one will not be resumed.

`:unnarrowed`
: Do not narrow the target buffer, simply show the full buffer. Default is to narrow it so that you only see the new material.

`:kill-buffer`
: If the target file was not yet visited when capture was invoked, kill the buffer again after capture is completed.

9.1.3.2 テンプレートの拡張

In the template itself, special `%`-escapes[2] allow dynamic insertion of content:

`%^{`*prompt*`}`	prompt the user for a string and replace this sequence with it. You may specify a default value and a completion table with %^{prompt|default|completion2|completion3...} The arrow keys access a prompt-specific history.
`%a`	annotation, normally the link created with `org-store-link`
`%A`	like `%a`, but prompt for the description part
`%i`	initial content, the region when capture is called while the region is active. The entire text will be indented like `%i` itself.
`%t`	timestamp, date only
`%T`	timestamp with date and time
`%u, %U`	like the above, but inactive timestamps
`%^t`	like `%t`, but prompt for date. Similarly `%^T`, `%^u`, `%^U` You may define a prompt like `%^{Birthday}t`
`%n`	user name (taken from `user-full-name`)
`%c`	Current kill ring head.
`%x`	Content of the X clipboard.
`%^C`	Interactive selection of which kill or clip to use.
`%^L`	Like `%^C`, but insert as link.
`%k`	title of the currently clocked task
`%K`	link to the currently clocked task
`%f`	file visited by current buffer when org-capture was called
`%F`	like `%f`, but include full path
`%^g`	prompt for tags, with completion on tags in target file.
`%^G`	prompt for tags, with completion all tags in all agenda files.
`%^{`*prop*`}p`	Prompt the user for a value for property *prop*
`%:keyword`	specific information for certain link types, see below
`%[`*file*`]`	insert the contents of the file given by *file*
`%(`*sexp*`)`	evaluate Elisp *sexp* and replace with the result

For specific link types, the following keywords will be defined[3]:

Link type	Available keywords
bbdb	%:name %:company
irc	%:server %:port %:nick
vm, wl, mh, mew, rmail	%:type %:subject %:message-id %:from %:fromname %:fromaddress %:to %:toname %:toaddress %:date (message date header field) %:date-timestamp (date as active timestamp) %:date-timestamp-inactive (date as inactive timestamp) %:fromto (either "to NAME" or "from NAME")[4]

[2] If you need one of these sequences literally, escape the `%` with a backslash.

[3] If you define your own link types (see Section A.3 [Adding hyperlink types], page 186), any property you store with `org-store-link-props` can be accessed in capture templates in a similar way.

[4] This will always be the other, not the user. See the variable `org-from-is-user-regexp`.

```
gnus                    |  %:group, for messages also all email fields
w3, w3m                 |  %:url
info                    |  %:file %:node
calendar                |  %:date
```

To place the cursor after template expansion use:

`%?` After completing the template, position cursor here.

9.2 Attachments

It is often useful to associate reference material with an outline node/task. Small chunks of plain text can simply be stored in the subtree of a project. Hyperlinks (see Chapter 4 [Hyperlinks], page 33) can establish associations with files that live elsewhere on your computer or in the cloud, like emails or source code files belonging to a project. Another method is *attachments*, which are files located in a directory belonging to an outline node. Org uses directories named by the unique ID of each entry. These directories are located in the '`data`' directory which lives in the same directory where your Org file lives[5]. If you initialize this directory with `git init`, Org will automatically commit changes when it sees them. The attachment system has been contributed to Org by John Wiegley.

In cases where it seems better to do so, you can also attach a directory of your choice to an entry. You can also make children inherit the attachment directory from a parent, so that an entire subtree uses the same attached directory.

The following commands deal with attachments:

C-c C-a `org-attach`

The dispatcher for commands related to the attachment system. After these keys, a list of commands is displayed and you must press an additional key to select a command:

`a` `org-attach-attach`

Select a file and move it into the task's attachment directory. The file will be copied, moved, or linked, depending on `org-attach-method`. Note that hard links are not supported on all systems.

c/m/l Attach a file using the copy/move/link method. Note that hard links are not supported on all systems.

n `org-attach-new`

Create a new attachment as an Emacs buffer.

z `org-attach-sync`

Synchronize the current task with its attachment directory, in case you added attachments yourself.

o `org-attach-open`

Open current task's attachment. If there is more than one, prompt for a file name first. Opening will follow the rules set by `org-file-apps`. For more details, see the information on following hyperlinks (see Section 4.4 [Handling links], page 35).

[5] If you move entries or Org files from one directory to another, you may want to configure `org-attach-directory` to contain an absolute path.

O org-attach-open-in-emacs
Also open the attachment, but force opening the file in Emacs.

f org-attach-reveal
Open the current task's attachment directory.

F org-attach-reveal-in-emacs
Also open the directory, but force using `dired` in Emacs.

d org-attach-delete-one
Select and delete a single attachment.

D org-attach-delete-all
Delete all of a task's attachments. A safer way is to open the directory in `dired` and delete from there.

s org-attach-set-directory
Set a specific directory as the entry's attachment directory. This works by putting the directory path into the `ATTACH_DIR` property.

i org-attach-set-inherit
Set the `ATTACH_DIR_INHERIT` property, so that children will use the same directory for attachments as the parent does.

9.3 RSS フィード

Org can add and change entries based on information found in RSS feeds and Atom feeds. You could use this to make a task out of each new podcast in a podcast feed. Or you could use a phone-based note-creating service on the web to import tasks into Org. To access feeds, configure the variable `org-feed-alist`. The docstring of this variable has detailed information. Here is just an example:

```
(setq org-feed-alist
     '(("Slashdot"
         "http://rss.slashdot.org/Slashdot/slashdot"
         "~/txt/org/feeds.org" "Slashdot Entries")))
```

will configure that new items from the feed provided by `rss.slashdot.org` will result in new entries in the file '`~/org/feeds.org`' under the heading '`Slashdot Entries`', whenever the following command is used:

C-c C-x g org-feed-update-all
C-c C-x g Collect items from the feeds configured in `org-feed-alist` and act upon them.

C-c C-x G org-feed-goto-inbox
Prompt for a feed name and go to the inbox configured for this feed.

Under the same headline, Org will create a drawer '`FEEDSTATUS`' in which it will store information about the status of items in the feed, to avoid adding the same item several times. You should add '`FEEDSTATUS`' to the list of drawers in that file:

```
#+DRAWERS: LOGBOOK PROPERTIES FEEDSTATUS
```

For more information, including how to read atom feeds, see '`org-feed.el`' and the docstring of `org-feed-alist`.

9.4 Protocols for external access

You can set up Org for handling protocol calls from outside applications that are passed to Emacs through the 'emacsserver'. For example, you can configure bookmarks in your web browser to send a link to the current page to Org and create a note from it using capture (see Section 9.1 [Capture], page 79). Or you could create a bookmark that will tell Emacs to open the local source file of a remote website you are looking at with the browser. See `http://orgmode.org/worg/org-contrib/org-protocol.php` for detailed documentation and setup instructions.

9.5 Refiling notes

When reviewing the captured data, you may want to refile some of the entries into a different list, for example into a project. Cutting, finding the right location, and then pasting the note is cumbersome. To simplify this process, you can use the following special command:

C-c C-w `org-refile`
: Refile the entry or region at point. This command offers possible locations for refiling the entry and lets you select one with completion. The item (or all items in the region) is filed below the target heading as a subitem. Depending on `org-reverse-note-order`, it will be either the first or last subitem.
By default, all level 1 headlines in the current buffer are considered to be targets, but you can have more complex definitions across a number of files. See the variable `org-refile-targets` for details. If you would like to select a location via a file-path-like completion along the outline path, see the variables `org-refile-use-outline-path` and `org-outline-path-complete-in-steps`. If you would like to be able to create new nodes as new parents for refiling on the fly, check the variable `org-refile-allow-creating-parent-nodes`. When the variable `org-log-refile`[6] is set, a timestamp or a note will be recorded when an entry has been refiled.

C-u C-c C-w
: Use the refile interface to jump to a heading.

C-u C-u C-c C-w `org-refile-goto-last-stored`
: Jump to the location where `org-refile` last moved a tree to.

C-2 C-c C-w
: Refile as the child of the item currently being clocked.

C-0 C-c C-w or *C-u C-u C-u C-c C-w*
C-0 C-c C-w or *C-u C-u C-u C-c C-w* `org-refile-cache-clear`
: Clear the target cache. Caching of refile targets can be turned on by setting `org-refile-use-cache`. To make the command see new possible targets, you have to clear the cache with this command.

[6] with corresponding `#+STARTUP` keywords `logrefile`, `lognoterefile`, and `nologrefile`

9.6 Archiving

When a project represented by a (sub)tree is finished, you may want to move the tree out of the way and to stop it from contributing to the agenda. Archiving is important to keep your working files compact and global searches like the construction of agenda views fast.

`C-c C-x C-a` `org-archive-subtree-default`
: Archive the current entry using the command specified in the variable `org-archive-default-command`.

9.6.1 Moving a tree to the archive file

The most common archiving action is to move a project tree to another file, the archive file.

`C-c C-x C-s` or short `C-c $` `org-archive-subtree`
: Archive the subtree starting at the cursor position to the location given by `org-archive-location`.

`C-u C-c C-x C-s`
: Check if any direct children of the current headline could be moved to the archive. To do this, each subtree is checked for open TODO entries. If none are found, the command offers to move it to the archive location. If the cursor is *not* on a headline when this command is invoked, the level 1 trees will be checked.

The default archive location is a file in the same directory as the current file, with the name derived by appending '`_archive`' to the current file name. For information and examples on how to change this, see the documentation string of the variable `org-archive-location`. There is also an in-buffer option for setting this variable, for example[7]:

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

If you would like to have a special ARCHIVE location for a single entry or a (sub)tree, give the entry an `:ARCHIVE:` property with the location as the value (see Chapter 7 [Properties and Columns], page 57).

When a subtree is moved, it receives a number of special properties that record context information like the file from where the entry came, its outline path the archiving time etc. Configure the variable `org-archive-save-context-info` to adjust the amount of information added.

9.6.2 ファイル内部でのアーカイブ

If you want to just switch off (for agenda views) certain subtrees without moving them to a different file, you can use the `ARCHIVE tag`.

A headline that is marked with the ARCHIVE tag (see Chapter 6 [Tags], page 53) stays at its location in the outline tree, but behaves in the following way:

[7] For backward compatibility, the following also works: If there are several such lines in a file, each specifies the archive location for the text below it. The first such line also applies to any text before its definition. However, using this method is *strongly* deprecated as it is incompatible with the outline structure of the document. The correct method for setting multiple archive locations in a buffer is using properties.

- It does not open when you attempt to do so with a visibility cycling command (see Section 2.3 [Visibility cycling], page 6). You can force cycling archived subtrees with *C-TAB*, or by setting the option `org-cycle-open-archived-trees`. Also normal outline commands like `show-all` will open archived subtrees.
- During sparse tree construction (see Section 2.6 [Sparse trees], page 11), matches in archived subtrees are not exposed, unless you configure the option `org-sparse-tree-open-archived-trees`.
- During agenda view construction (see Chapter 10 [Agenda Views], page 89), the content of archived trees is ignored unless you configure the option `org-agenda-skip-archived-trees`, in which case these trees will always be included. In the agenda you can press `v a` to get archives temporarily included.
- Archived trees are not exported (see Chapter 12 [Exporting], page 124), only the headline is. Configure the details using the variable `org-export-with-archived-trees`.
- Archived trees are excluded from column view unless the variable `org-columns-skip-archived-trees` is configured to `nil`.

The following commands help manage the ARCHIVE tag:

C-c C-x a `org-toggle-archive-tag`

Toggle the ARCHIVE tag for the current headline. When the tag is set, the headline changes to a shadowed face, and the subtree below it is hidden.

C-u C-c C-x a

Check if any direct children of the current headline should be archived. To do this, each subtree is checked for open TODO entries. If none are found, the command offers to set the ARCHIVE tag for the child. If the cursor is *not* on a headline when this command is invoked, the level 1 trees will be checked.

C-TAB `org-force-cycle-archived`

Cycle a tree even if it is tagged with ARCHIVE.

C-c C-x A `org-archive-to-archive-sibling`

Move the current entry to the *Archive Sibling*. This is a sibling of the entry with the heading '`Archive`' and the tag '`ARCHIVE`'. The entry becomes a child of that sibling and in this way retains a lot of its original context, including inherited tags and approximate position in the outline.

10 アジェンダビュー

Org-mode で作業した結果、TODO アイテム、タイムスタンプのついたアイテム、タグの付いた見出しなどが、1 つのファイル、あるいはいくつものファイルにまたがって、撒き散らされることとなります。ある特定の日に重要な、実際に動いているアイテムやイベントの全体像を把握するためには、ひとつの管理された方法で、これらの情報を集めたり、並び替えたりしながら、表示することが必要です。

Org-mode では、いろいろな基準によってアイテムを選択することが可能であり、独立したバッファにそれらのアイテムを表示させることができます。7 つの異なるビューのタイプが用意されています。:

- アジェンダ　カレンダーのように指定した日付の情報を表示します、
- *TODO* リスト　未完了のアクションアイテムをカバーします、
- マッチビュー　関連づけられているタグやプロパティ、TODO の状態に基づいて見出しを表示します、
- タイムラインビュー　1 つの Org-mode のファイルの中に含まれている全てのイベントを時間順のビューに表示します、
- a テキストの検索ビュー　複数のファイルの中かから、指定したキーワードを含んでいるすべてのエントリーを表示します、
- a 詳細が未決定のプロジェクトビュー　現在作業が進んでいないプロジェクトを表示します。そして、
- カスタムビュー　特別な検索や異なるビューの組合せによるビューです。

抽出された情報は特別なアジェンダバッファに表示されます。このバッファはリードオンリーですが、オリジナルの Org-mode ファイルにジャンプしたり、オリジナルのファイルを間接的に編集することができます。

2 つの変数によって、アジェンダバッファをどのように表示するか、アジェンダが存在したときに、ウインドウの設定を元に戻すかどうかをコントロールします。; `org-agenda-window-setup`と`org-agenda-restore-windows-after-quit`.

10.1 Agenda files

表示される情報は、通常すべてのアジェンダファイルから収集されます。アジェンダファイルは`org-agenda-files`[1] 変数にリスト化されたファイルが対象となります。もしもこのリストの中にディレクトリ名が記載されていたら、そのディレクトリの中にある‘`.org`’という拡張子がついた全てのファイルが、アジェンダファイルの対象となります。

したがって、たとえあなたが 1 つの Org-mode ファイルでしか作業をしていなくても、このファイルをそのリスト[2] に記載したことになるでしょう。`org-agenda-files`をカスタマイズすることが可能で、しかも以下に述べるコマンドを通して簡単な方法で維持することができます。

---

[1] もしもその変数の値がリストではなく、単独のファイル名の場合には、その外部ファイルの中に記載されているアジェンダファイルの名前となります。

[2] コマンド選択画面を使用しているときに、コマンドを選択する前に、<を押すと、編集中のファイルに対するコマンドが制限されて、次のコマンド選択画面でコマンドが入力されるまで、`org-agenda-files`は無視されます。

C-c [org-agenda-file-to-front
: アジェンダファイルのリストに編集中のファイルを追加する。そのファイルは、リストの先頭に追加される。もしも既にリストに存在していたら、先頭に移動する。前置引数をつけることで、リストの最後に追加／移動する。

C-c] org-remove-file
: 編集中のファイルをアジェンダファイルのリストから削除する。

C-' org-cycle-agenda-files
C-, アジェンダファイルのリストに従って、1 つのファイルから次のファイルへと切り替える。

M-x org-iswitchb
: iswitchbと似たようなインターフェースで Org-mode のバッファの間を切り替えるコマンド。

Org-mode メニューには、現時点のファイルのリストが含まれており、その中のファイルに移動するのに役立ちます。

もしもこのリストに載っているファイルではなく、作業中のアジェンダファイルに焦点をあてたかったり、リストにあるファイルのまさにひとつのファイルに焦点をあてたかったり、はたまたあるファイルの中のあるサブツリーに焦点をあてたかったりしたいときは、いくつかの方法が用意されています。単一のアジェンダコマンドとして、コマンド選択画面上 (see Section 10.2 [Agenda dispatcher], page 90) で<を 1 回ないし数回押すとよいのです。アジェンダの対象をある限定した期間に絞り込むために以下のコマンドが用意されています。:

C-c C-x < org-agenda-set-restriction-lock
: アジェンダの対象を現在カーソルが置かれているサブツリーに固定的に制限します。前置引数をつけたり、ファイルの最初の見出しよりも前にカーソルが置かれているときには、アジェンダの対象範囲はファイル全体になります。この制約は *C-c C-x >*を実行して取り除くか、<または>をアジェンダのコマンド選択画面上で入力するまでは維持します。もしもウインドウ上にアジェンダビューが表示されているならば、あたらしい制約が即座に効果を及ぼします。

C-c C-x > org-agenda-remove-restriction-lock
: *C-c C-x <*で作成された固定する制限を削除します。

'speedbar.el'を併用しているときは、Speedbar のフレームの中で以下のコマンドを使用することができます。

< in the speedbar frame org-speedbar-set-agenda-restriction
: Speedbar のフレームの中で、1 つの Org-mode ファイルか、そのファイルのサブツリーの一つか、カーソルの置かれているアイテムに対応してアジェンダを恒久的に限定します。もしもアジェンダビューが表示されているウインドウがあるならば、限定箇所が変更されると即座に反映する。

> in the speedbar frame org-agenda-remove-restriction-lock
: 制限をふたたび解除する。

10.2 アジェンダのコマンド選択画面

グローバルなキーと結びついている、コマンド選択画面を通してそのビューは作成されます。—例えば、*C-c a* (see Section 1.2 [Installation], page 3) のように。以下のように、コマンド選択画面に

アクセスする方法として *C-c a*を想定しており、キーボードでコマンドにアクセスするためのリストが表示されています。*C-c a*を入力した後、コマンドを実行するために、次に入力する文字を要求します。コマンド選択画面では以下に記載するデフォルトのコマンドが提供されています。

`a` カレンダーのようなアジェンダを作成します。(see Section 10.3.1 [Weekly/daily agenda], page 91)

`t` / `T` すべてのTODOアイテムのリストを作成します。(see Section 10.3.2 [Global TODO list], page 93)

`m` / `M` タグの表記にマッチした見出しのリストを作成します。(see Section 10.3.3 [Matching tags and properties], page 94)

`L` カレントバッファ用のタイムラインのビューを作成します。(see Section 10.3.4 [Timeline], page 96)

`s` そのエントリーに存在するしないにかかわらず、and/orという正規表現によるキーワードの論理式で選択したエントリのリストを作成します。

`/` すべてのアジェンダファイルと`org-agenda-text-search-extra-files`の中でリスト化かれているファイルの中から正規表現を用いて検索します。これはEmacsの`multi-occur`というコマンドを使用します。前置引数をつけると、それぞれのマッチした行の状況の数をしていすることができます。デフォルトは1となっています。

`#` / `!` 詳細が未決定のプロジェクトのリストを作成します。(see Section 10.3.6 [Stuck projects], page 97)

`<` カレントバッファ[3]に対してアジェンダコマンドを制限します。`<`を入力したあと、コマンドを選択するために文字を入力する必要があります。

`< <` もしもアクティブなリージョンがあるときは、以下のようなアジェンダコマンドがそのリージョンに限定されます。一方、カレントのサブツリー[4]に限定することもできます。`< <`を入力したあと、コマンドを選択する文字を入力する必要があります。

あなたは、あたかもデフォルトのコマンドのように、コマンド選択画面でアクセスするカスタムコマンドを定義することもできます。複数のブロックを同時に含めた拡張されたアジェンダバッファを作成する可能性を含んでいます。例えば週のアジェンダ、グローバルなTODOリスト、そして多数の特定タグの検索など。See Section 10.6 [Custom agenda views], page 109.

10.3 agendaに組み込まれているビュー

このセクションではビルトインビューについて説明します。

10.3.1 1週間／1日のアジェンダ

1週間の／1日のアジェンダの目的は、その週あるいはその日のタスクをすべて表示して、紙のアジェンダのページのように、実行に移すことです。

---

[3] 逆の互換性として、*1*をカレントバッファを制限するために入力することもできます。

[4] 逆の互換性として、カレントリージョンまたはカレントサブツリーに限定するために *0*を入力することもできます。.

`C-c a a` `org-agenda-list`

Org-mode のファイルのリストの中からその週の予定を収集するものです。予定はそれぞれの日に表示されます。(`C-u 2 1 C-c a a`のように) 前置引数に数字をつけて[5] 表示する日数を設定することができます。

表示されるデフォルトの日数は、`org-agenda-span`(あるいは古くさくなってしまいましたが`org-agenda-ndays`) という変数で設定します。この変数は、アジェンダの中でデフォルトとして確認したい日数、あるいは、期間を示す`day`、`week`、`month`や`year`といった期間を示す名前をつけて設定します。

アジェンダバッファからリモートで編集するとは、例えば、アジェンダバッファの中でデッドラインやアポイントメントの日付を変更することができるという意味です。アジェンダバッファの中で利用できるコマンドは、Section 10.5 [Agenda commands], page 100 の中で一覧表にしています。

カレンダー／日記の統合

Emacs には、Edward M. Reingold によって開発されたカレンダーと日記の機能があります。カレンダーでは、国や文化の異なる祝祭日を備えた 3ヵ月分のカレンダーが表示されます。日記には記念日、月の満ち欠け、日の出日の入り、繰り返しの予定（隔週、隔月）などを記録しておくことができます。このような機能は、Org-mode に対して大変補完的な関係にあります。日記と Org-mode の出力を結びつけることは大変有益です。

Emacs の日記から Org-mode のアジェンダに項目を落とし込むために、あなたは次のように変数を設定するだけです。

```
(setq org-agenda-include-diary t)
```

After that, everything will happen automatically. All diary 祝祭日や記念日などを含むすべての項目は、Org-mode で作成されるジェンダバッファに取り込むことができます。日記に記録されている項目を編集するために、アジェンダバッファ上で`SPC`、`TAB`、及び`RET`を入力することで、日記のファイルにジャンプすることができます。その日に新しいエントリーを挿入する`i`というコマンドはアジェンダバッファ上で動作します。あたかも、日の出日の入りの時刻を表示したり、月の満ち欠けの状態を表示したり、他の暦に変換するための、`S`、`M`、および`C`というコマンドと同様です。`c`はカレンダーとアジェンダの間を行ったり来たりすることができます。

もしもあなたが日記を S 式項目と祝祭日だけで使用しているのならば、上のような設定をするよりも、Org-mode ファイルに直接コピーしたり移動したりしたほうが手っ取り早いです。Org-mode は日記形式の S 式項目を評価し、しかもより早く、というのは、最初にカレンダーを表示するという負荷がかからないからです。S 式項目は左端から記述し、式の前にスペースが入ってはいけないことに注意してください。たとえば、ある Org-mode ファイルについての、以下にのべるセグメントが処理され、項目がアジェンダの中に作成されます。

```
* Birthdays and similar stuff
#+CATEGORY: Holiday
%%(org-calendar-holiday)   ; special function for holiday names
#+CATEGORY: Ann
%%(diary-anniversary  5 14 1956)[6] Arthur Dent is %d years old
%%(diary-anniversary 10  2 1869) Mahatma Gandhi would be %d years old
```

[5] 逆方向の互換性のために、普遍的な前置引数`C-u`をつけることでアジェンダ（予定表）より上に、TODO リストを書き出すことができます。この機能は軽視されており、専用の TODO リストやブロックアジェンダ (see Section 10.6.2 [Block agenda], page 110). をその代わりに利用することが多いです。

[6] Note that the order of the arguments (month, day, year) depends on the setting of `calendar-date-style`.

Anniversaries from BBDB

もしも Big Brothers Database を使用して連絡先を管理しているのならば、あなたは先に述べたのと同様に、独立した Org-mode のファイルや日記のファイルに登録するよりも、BBDB の中に記念日を登録したいと考えるでしょう。Org-mode はこれもサポートしており、アジェンダの一部として BBDB の記念日を表示することができます。そのために必要なことは、以下のような記述をアジェンダファイルに行うことです。

```
* Anniversaries
  :PROPERTIES:
  :CATEGORY: Anniv
  :END:
%%(org-bbdb-anniversaries)
```

それから BBDB のデータレコードのための記念日の定義に取り掛かることができます。基本的には、BBDB のレコードの中にカーソルを置いて、*C-o anniversary RET*を実行し、それから日付を YYYY-MM-DDまたは MM-DDの形式で記入し、半角スペースに続けて記念日の種類 (‘birthday’、‘wedding’、または定型句) のクラスを記入します。もしもクラスを省略した場合は、デフォルトでは‘birthday’であるとみなします。いくつかの例を書いてみました。‘org-bbdb.el’ファイルの先頭のところにもう少し詳しい説明が書いてあります。

```
1973-06-22
06-22
1955-08-02 wedding
2008-04-14 %s released version 6.01 of org-mode, %d years ago
```

BBDB を変更したり、Emacs のセッションで最初にアジェンダを表示したとき後は、アジェンダの表示が少し遅くなるかもしれません。というのは Org-mode が記念日のハッシュデータを更新するからです。しかしながら、そのことについていうと非常に早いといえます。実際 Org-mode の日記ファイルに‘%%(diary-anniversary)’のエントリーを長々と書き連ねた場合よりもずっと早いと言えるでしょう。

Appointment reminders

Org-mode は Emacs の予定を通知する機能と連携しています。あなたのアジェンダファイルに含まれているすべてのアポイントを追加するために、org-agenda-to-apptコマンドを使います。このコマンドはあなたの予定のリストにフィルターをかけ、特別なカテゴリーに属しているものや正規表現の検索に合致したものを追加します。詳細はドキュメント文字列を参照してください。

10.3.2 The global TODO list

グローバルな TODO リストには、形式を整えられ、1 つの場所に集められたすべての未完了の TODO アイテムが含まれています。

C-c a t org-todo-list

グローバルな TODO リストを表示します。これはすべてのアジェンダファイル (see Chapter 10 [Agenda Views], page 89) から TODO アイテムを 1 つのバッファに集約します。デフォルトでは、このアイテムのリストは DONE という状態ではないアイテムです。そのバッファは agenda-modeとなり、そのバッファから TODO アイテムを直接調べたり操作したりするコマンドが用意されています (see Section 10.5 [Agenda commands], page 100)。上と似ていますが、指定した TODO キーワードと合致したものを表示します。同じことを前置引数をつけて *C-c a t*を実行することでも指定できます。キーワードの入力を促す指示が表示され、そして複数のキーワードを論理式 OR

という意味で‘|’で区切って指定することができます。数字付きの前置引数をつけるとorg-todo-keywordsの中のN番目のキーワードを選択することができます。*r*キーをアジェンダバッファで使用するとバッファの再構成が行われます。たとえば*3 r*というように、前置引数をつけてこのコマンドを実行すると選択したTODOキーワードが変更することができます。もしも特定のキーワードを使って検索することが多い場合は、カスタムコマンドを定義することもできます (see Section 10.2 [Agenda dispatcher], page 90)。
特定のTODOキーワードと合致するものを検索するのは、タグ検索の1機能として行うこともできます (see Section 6.3 [Tag searches], page 55)。

リモートでTODOアイテムを編集するということの意味は、1つのキーを入力することでTODOエントリーの状態を変更できるということです。TODOリストの中で利用できるコマンドはSection 10.5 [Agenda commands], page 100の記述を参考にしてください。

通常グローバルなTODOリストには、TODOキーワードのついたすべて見出しが表示されます。このリストは大変長いものになる場合もあります。それをコンパクトにするには2つの方法があります。

– TODOアイテムが、実行するために *scheduled* となっている、あるいは、もはや *open* となっている *deadline*(see Section 8.1 [Timestamps], page 65) を持っているかどうかを確認したい人もいるでしょう。org-agenda-todo-ignore-scheduled、org-agenda-todo-ignore-deadlines、org-agenda-todo-ignore-timestamp および／または org-agenda-todo-ignore-with-dateという変数を設定し、グローバルなTODOリストから取り除くことができます。
– TODOアイテムがサブタスクにブレイクダウンされた下位のレベルを持っているかもしれません。そういった場合は、最上位のTODOの見出しを表示すれば十分で、グローバルなリストからは下位のレベルの項目は省略してもよい場合があります。そういったときはorg-agenda-todo-list-sublevels変数を設定することで可能となります

10.3.3 Matching tags and properties

アジェンダファイルの中の見出しに *tags*(see Chapter 6 [Tags], page 53) がついていたり、あるいは属性 (see Chapter 7 [Properties and Columns], page 57) がついていたときは、このメタデータに基づいて見出しを選択し、アジェンダバッファに収集することができます。この項で述べている検索構文は *C-c / m*を用いたツリーの抽出を行うときも適用できます。

C-c a m org-tags-view
一組のタグのセットに合致したすべての見出しのリストを作成します。選択の基準の入力を指示するコマンドでタグのついた論理式による表現で記入します。例えば、‘+work+urgent-withboss’あるいは‘work|home’というように (see Chapter 6 [Tags], page 53)。もしも特定の検索をよく行うならばそのためのカスタムコマンドを定義することができます (see Section 10.2 [Agenda dispatcher], page 90)。

C-c a M org-tags-view
*C-c a m*と似ていますが、not-DONEの状態にあるTODOアイテムの見出しから選択するもので、自動的にサブアイテムもチェックします (org-tags-match-list-sublevels変数参照)。予定／期限のついたアイテムを除外するにはorg-agenda-tags-todo-honor-ignore-optionsの変数を参照してください。特定のTODOキーワードをタグの一致と一緒に指定することも可能です。Section 6.3 [Tag searches], page 55を参照してください。

タグのリストで利用できるコマンドは Section 10.5 [Agenda commands], page 100 のところで説明しています。

Match syntax

検索文字列では AND の意味で '&'、OR の意味で '|'という論理式を使うことができます。'&'は '|' よりも強く結びつけます。括弧 () は現在準備されていません。検索のどの要素も、タグそのものか、正規表現でマッチしたタグか、あるいは PROPERTY OPERATOR VALUEのような属性値にアクセスして比較操作のできる値のいずれかになります。どの要素も '-'を先頭につけてそれ以外のものを表現するか、'+'を先頭につけてポジティブな選択を行う、というような糖衣構文（簡便な構文）で表現します。'&'で AND を取り扱うことは '+'、'-'で表現できるもののオプションです。下にタグだけをつかったいくつかの例を挙げておきました。

'+work-boss'
: ':work:'というタグがついているが、':boss:'というタグがついていない見出しを選択します。

'work|laptop'
: ':work:'または ':laptop:'というタグがついたものを選択します。

'work|laptop+night'
: 前の文と同じですが、':laptop:'の行には、同時に ':night:'というタグが付いている必要があります。

タグの代わりに、大括弧でくくられた正規表現により指定をすることもできます。例えば、'work+{^boss.*}'と指定すると、':work:'というタグのついた見出しで 'boss'という単語で *starting* するタグがついているものに一致します。

タグとマッチするものを探すと同時に属性 (see Chapter 7 [Properties and Columns], page 57) の検索をすることも可能です。属性としては実際の属性のほかに、他のメタデータで表現された特別な属性 (see Section 7.2 [Special properties], page 58) にも対応しています。例えば、そのエントリーの中の TODO キーワードで表現された TODOという「属性」。あるいは、そのエントリーの階層を示す LEVELという「属性」などです。そのため、'+LEVEL=3+boss-TODO="DONE"'という検索式は、第 3 階層のすべての見出しの中で、'boss'というタグがついており、TODO キーワードが DONE では'ない'もののリストを表示します。org-odd-levels-onlyという設定がなされているバッファでは 'LEVEL'は * の数を数えるのではなく、'LEVEL=2'（2 番目）の階層は * が 3 つある階層が該当します。

いくつかの例を紹介します。

'work+TODO="WAITING"'
: ':work:'というタグがある TODO 行のうち、特に TODO キーワードが 'WAITING'となっている行を選択します。

'work+TODO="WAITING"|home+TODO="WAITING"'
: work と home というタグがついている Waiting となっているタスク

属性の検索では、多数の異なる操作で属性の値をテストすることができます。複雑な例を挙げます。

```
+work-boss+PRIORITY="A"+Coffee="unlimited"+Effort<2          \
      +With={Sarah\|Denny}+SCHEDULED>="<2008-10-11>"
```

比較のタイプは比較の値がどのように書かれているかによります。

– 比較する値が普通の数字ならば、数値の比較が行われ、'<'、'='、'>'、'<='、'>='、および '<>'という操作が可能です。

– 比較する対象がダブルクォーテーションで囲まれている場合は、文字列の比較が行われ、前項と同じ操作が可能です。
– もしも比較対象が、(‘DEADLINE<="<2008-12-24 18:30>"’のように)、ダブルクォーテーションおよび角括弧<>で囲まれていた場合は、両方の値が Org-mode 流の標準的な日付・時刻の指定であると仮定し、それにそって比較を行います。いくつかの特別な値があります。"<now>"は(時刻も含めた)現在を示し、"<today>"、"<tomorrow>"はそれらの日の 0:00 つまり、時刻の指定がないことを表します。同様に、"<+5d>"または"<-2m>"というような文字列は、それぞれ日、週、月、年を示す、d、w、m、yという単位がついているものとして使用されます。
– もしも比較対象が中括弧 {} でくくられていて、正規表現での比較がなされるときは、‘=’は一致していることを示し、‘<>’は一致していないことを示します。

そのため、例に掲げた検索文字列の意味は、‘:work:’というタグがつけられているが、‘:boss:’というタグはついておらず、また、優先順位の値が‘A’であり、‘:Coffee:’が‘unlimited’という値であり、‘Effort’属性が数値で 2 より小さく、‘:With:’の値が‘Sarah\|Denny’であり、スケジュールが 2008 年 10 月 11 日もしくはそれ以降に予約されたものを示しています。

TODO、LEVEL、CATEGORY を検索するときは短時間ですみます。それ以外の属性を検索するときはいささか時間がかかります。しかしながら、一度高い代償を払って 1 つのプロパティを検索したら、他の属性を追加して再び検索するときは安くあがります。

検索の際に Org-mode で属性の継承という機能を使用するように設定することができますが、相当検索スピードが落ちることを覚悟してください。詳細は Section 7.4 [Property inheritance], page 59 参照。

逆互換として、さらにまたタイプのスピードを上げるために、検索において TODO の状態をテストする別の方法があります。このためには、検索文字列(それは‘|’で結合された複数の用語が含まれていると思いますが)のタグ・属性検索の部分を‘/’を使って終了させ、TODO キーワードを論理式で結んで指定します。その構文はタグの検索で使用したのと似ていますが、よく考えて適用する必要があります。例えば、複数の TODO キーワードが存在することを検索するには論理式の AND で結びつけても意味がありません。しかしながら、*negative selection*(存在しないことを選択する場合)では「AND」で結合することは意味を持ちます。これを確かめるには、実際にいくつかの TODO キーワードで、*C-c a M*を用いて確認するだけです(そのほうがスピードアップできます)。あるいはスラッシュのあとに‘!’を記入して同時に TODO の部分を開始します。*C-c a M*または‘/!’を使用したときは、DONE の状態にある TODO キーワードを検索することはできません。例えば、

‘work/WAITING’
: ‘work+TODO="WAITING"’と同じ

‘work/!-WAITING-NEXT’
: ‘:work:’を選択。ただし TODO 行では‘WAITING’と‘NEXT’のどちらのタグもついていないもの

‘work/!+WAITING|+NEXT’
: ‘:work:’を選択。TODO 行に‘WAITING’か‘NEXT’かどちらかのタグがついているもの。

10.3.4 Timeline for a single file

タイムラインはひとつの Org-mode ファイルの中から *time-sorted view*(時間順のビュー)ですべてのタイムスタンプのついたアイテムをまとめてて表示します。このコマンドの主な目的は、あるプロジェクトに含まれているイベント全体の概要をつかむためにあります。

C-c a L org-timeline
すべてのタイムスタンプの付いたアイテムについて、Org-mode ファイルの中で時間順のビューを提供します。*C-u*という前置引数をつけて呼び出したときは、現在の日付の時点で、すべての未完了の TODO エントリー（予約されているものも、そうでないものも）を一覧にします。

タイムラインのバッファで利用できるコマンドは、Section 10.5 [Agenda commands], page 100 にリスト化されています。

10.3.5 Search view

アジェンダのビューでは Org-mode のエントリーに対する一般的なテキスト検索機能を持っています。これはノートを探すのに特に役に立ちます。

C-c a s org-search-view
このコマンドは特別な検索のためのもので、論理式を使って、文字列または特定の単語に合致するエントリーを選択します。

例えば、‘computer equipment’という検索文字列は、‘computer equipment’という 1 つの文字列が含まれているエントリーを検索するでしょう。もしも、2 つの単語が、1 つ以上のスペースまたは改行で分かれていても、依然として一致するものを検索するでしょう。検索ビューでは、エントリーの中にある特別なキーワードについて論理式を使って検索することもできます。‘+computer +wifi -ethernet -{8\.11[bg]}’という検索文字列では、次のようなノートエントリーを検索します。computerと wifiというキーワードを含んでおり、ethernetというキーワードは含まれておらず、8\.11[bg]という正規表現を含んでいない、すなわち 8.11b および 8.11g ともに含まれていないという意味ですが、エントリーを検索します。最初の ‘+’は単語検索を開始するために必要ですが、ほかの ‘+’はオプションです。詳しく知りたい場合は、org-search-viewというコマンドのドキュメント文字列を参照してください。

アジェンダファイルに加えて、このコマンドは org-agenda-text-search-extra-filesの中で一覧になっているファイルもまた検索するということに注意してください。

10.3.6 Stuck projects

もしもあなたが、以下に述べるような David Allen 氏の GTD のようなシステムであなたの仕事を管理しているならば、あなたが抱えている「義務」のひとつは、すべてのプロジェクトが進んでいるかを明確にするために、レビューを定期的に行うことです。詳細が未決定のプロジェクトは、次の行動が何も定義がされていないため、Org-mode が提示する TODO リストに、全く何も表示されることがないのです。レビューをする際に、そういったプロジェクトを明確にし、それらのプロジェクトための次の行動を定義することが必要です。

C-c a # org-agenda-list-stuck-projects
詳細が未決定のプロジェクトリスト

C-c a ! org-stuck-projectsの変数をカスタマイズすることで何が詳細が未決定のプロジェクトで、どうやったらそういうプロジェクトを発見できるかを定義することができます。

あなたは九分九厘このコマンドが機能するために、このビューを定義する必要があります。あらかじめビルトインされているデフォルトの設定では、すべてのあなたのプロジェクトは第 2 階層の見出しに記述されており、あるプロジェクトが未決定であるとはいえない状況とは、すくなくとも 1 つのエントリーに TODO または NEXT または NEXTACTION という印がつけられている場合です。

Org-mode を使う際に、あなた自身の方法でアプローチするとして、PROJECT というタグがあるものをプロジェクトと定義し、プロジェクトがまだ検討する段階にないということを示すために

TODO キーワードで MAYBE と書いているものと仮定しましょう。さらに TODO キーワードで DONE という印の付いたものは完了したプロジェクトであると仮定しましょう。そしてまた NEXT もしくは TODO と書かれたものは NextAction であると仮定しましょう。@SHOP というタグがついたときは NEXT というタグが付いていなくても、ショッピングに行くという次の行動を示しているとします。最終的に、もしもプロジェクトに IGNORE（無視）という特別なキーワードがどこかについていたら、それはリストに表示されないものとします。このようなケースの場合、タグ・TODO[7] が '+PROJECT/-MAYBE-DONE'とマッチし、さらにサブツリーに TODO、NEXT、@SHOP、および IGNORE というタグが付いているようなプロジェクトは、詳細が未決定のプロジェクトではないといえます。このようなカスタマイズを正しく定義するには、

```
(setq org-stuck-projects
      '("+PROJECT/-MAYBE-DONE" ("NEXT" "TODO") ("@SHOP")
                               "\\<IGNORE\\>"))
```

もしもあるプロジェクトが詳細が未決定のプロジェクトではないと定義されたならば、そのエントリーのサブツリーは依然として詳細が未決定のプロジェクトとして検索されるということに注意してください。

10.4 Presentation and sorting

アジェンダビューにアイテムが表示される前に、Org-mode ではそのアイテムを表示し並び替える準備を行っています。それぞれのアイテムは 1 行を占めます。その行にはその項目の *category* (see Section 10.4.1 [Categories], page 98) を含んだ *prefix* とそれ以外の重要な情報を含んでいます。あなたは `org-agenda-tags-column`を使って表示されるコラムタグをカスタマイズすることができます。`org-agenda-prefix-format`のオプションを使用して前置引数をカスタマイズすることができます。この前置引数は、そのアイテムに関連するアウトラインの見出しの最新のバージョンに従います。

10.4.1 Categories

カテゴリーとは、それぞれのアジェンダアイテムに割り当てられた幅の広いラベルです。デフォルトでは、カテゴリーはファイルの名前から単純に作成されます。しかし、バッファ上で特別な行を足すことでそれを指定することができます。[8]

```
#+CATEGORY: Thesis
```

もしもあなたが、1 つのエントリーもしくは 1 つの（サブ）ツリーに特別な CATEGORY を持たせたいと望むのならば、そのエントリーに、値として適用したいと思っている特別なカテゴリーを `:CATEGORY:`という属性に設定しなさい。

アジェンダバッファの表示は、そのカテゴリーが 10 文字以上長くしない方が見栄えが良いです。

あなたは `org-agenda-category-icon-alist`変数をカスタマイズすることで、カテゴリーにアイコンを設定することができます。

---

[7] See Section 6.3 [Tag searches], page 55.

[8] 逆に言うと、以下のような動作も生じます。もしも 1 つのファイルの中に、いくつかののそういう行が存在するならば、それよりも下の行にあるテキストに、そのカテゴリーをそれぞれ指定することになります。最初のカテゴリーは、その最初の CATEGORY の行はよりも前にあるどのテキストにも適用されます。しかしながら、*strongly* という手法を使うことは、文書のアウトライン構造と非互換であることを、強く非難することになります。複数のカテゴリーをバッファの中で設定する正しい方法は属性を使用することです。.

10.4.2 Time-of-day specifications

Org-mode は時刻の仕様に基づいて、それぞれのアジェンダアイテムをチェックします。時刻は、例えば、‘<2005-05-10 Tue 19:00>’ のように、アジェンダの中に含まれているものをトリガーとしたタイムスタンプの一部です。時間の幅は 2 つのタイムスタンプで指定され、例えば ‘<2005-05-10 Tue 20:30>--<2005-05-10 Tue 22:15>’ のように記載されます。

そのエントリー自身の見出しの中で、時刻（時間）はプレーンなテキストとして（‘12:45’や ‘8:30-1pm’) のように表示されます。もしもアジェンダが Emacs のダイアリー (see Section 10.3.1 [Weekly/daily agenda], page 91) と一体化されていたときは、ダイアリーのエントリーの中で指定した時間は、同様に認識されます。

アジェンダの表示のために、Org-mode は時間を引き出し、前置引数の一部として標準的な 24 時間のフォーマットでそれを表示します。前の段落に書かれた時間の例は、アジェンダの中で結局以下のように表示されます。

```
 8:30-13:00 Arthur Dent lies in front of the bulldozer
12:45...... Ford Prefect arrives and takes Arthur to the pub
19:00...... The Vogon reads his poem
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

もしもアジェンダが一日モードであるならば、あるいは今日を表示しているならば、時間設定されたエントリーは、次のような時間のグリッドに埋め込まれます。

```
 8:00...... ------------------
 8:30-13:00 Arthur Dent lies in front of the bulldozer
10:00...... ------------------
12:00...... ------------------
12:45...... Ford Prefect arrives and takes Arthur to the pub
14:00...... ------------------
16:00...... ------------------
18:00...... ------------------
19:00...... The Vogon reads his poem
20:00...... ------------------
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

時間のグリッドは、org-agenda-use-time-grid変数で表示したりしなかったさせることができます。そしてまた org-agenda-time-gridで設定をすることができます。

10.4.3 agenda の項目をソートする

ビューに書き出される前に、各アイテムは並び替えが行われます。この並び替えはビューのタイプによって決まります。

- 一日／一週間のアジェンダでは、それぞれの日の各アイテムは順番に並びます。デフォルトの順番は、明示的に日付と時刻の指定を含んでいるアイテムを、最初に集めます。これらのアイテムは、その日のスケジュールに応じて、リストの最初から順番に表示されます。その次に、各アイテムは org-agenda-filesによって決められた順番に、カテゴリーごとにグループ分けされます。それぞれのカテゴリーの中で、各アイテムは優先順位 (see Section 5.4 [Priorities], page 49) に従って並び替えられます。優先順位は基本的な優先順位で構成されます（優先順位 ‘A’ならば 2000、‘B’ならば 1000、‘C’ならば 0 として)。さらに、予定あるいはデッドラインを過ぎているアイテムのウエイトが追加されます。
- TODO リストでは、各アイテムはカテゴリーの順番に並び替えられますが、各カテゴリーの中では、優先順位 (see Section 5.4 [Priorities], page 49) によって並び替えられます。優先順位

は、優先順位の記号に従って並べ替えられます。さらに、アイテムが実行する日あるいは予約した日にどれだけ近いかということも考慮されます。

- タグでの一致については、項目は並び替えは行われず、アジェンダファイルの中で一致した項目が発見された順番に従って表示されるのみです。

並び替えは、`org-agenda-sorting-strategy`変数でカスタマイズすることができます。そして、並び替えはそのエントリーの工数の見積りに基づく評価も含まれます。

10.5 Commands in the agenda buffer

アジェンダバッファでのエントリーは、その項目が作成された Org-mode ファイルと日記ファイルの間でリンクされます。アジェンダバッファでは編集することはできませんが、コマンドを使って、そのエントリーがある場所を表示したり、ジャンプして、アジェンダバッファから「遠隔的に」Org-mode ファイルを編集することができます。この方法で、すべての情報は 1 度書き込めばよく、あなたがアジェンダとノートのファイルが別の情報になるというリスクを避けることができます。

いくつかのコマンドはアジェンダの行上でマウスをクリックすることで実行されます。それ以外のコマンドは、必要とされる行の中にカーソルが置かれている必要があります。

Motion

n `org-agenda-next-line`
次の行へ (up及び *C-p*と同じ)。

p `org-agenda-previous-line`
次の行へ (down及び *C-n*と同じ)。

View/Go to Org file

SPC or *mouse-3* `org-agenda-show-and-scroll-up`
そのアイテムのオリジナルの場所を別のウインドウで表示する。前置引数を使うことで、見出しだけでなく、アウトライン上にエントリー全体を明確に表示する。

L `org-agenda-recenter`
オリジナルの場所を表示し、ウインドウのセンターに再配置する。

TAB or *mouse-2* `org-agenda-goto`
別のウインドウでそのアイテムのオリジナルの場所に移動する。

RET `org-agenda-switch-to`
そのアイテムのオリジナルの場所に移動し、他のウインドウは削除する。

F `org-agenda-follow-mode`
Follow モードをトグルする。Follow モードではアジェンダバッファ上でカーソルを動かすと、Org-mode ファイルの中で、別のウインドウ上で対応する場所を表示する。新しいアジェンダバッファの中でこのモードの初期設定値は、`org-agenda-start-with-follow-mode`変数で設定することができる。

C-c C-x b `org-agenda-tree-to-indirect-buffer`
間接的なバッファの中で可憐とアイテムのサブツリー全体を表示する。数値付きの前置引数 N をつけると、第 N 階層まで階層を上がり、そのツリーを取得する。もしも N がマイナスならば、多くの階層まで上がる。*C-u*という前置引数を付けた場合は、既に使われた間接的バッファは消去されない。

C-c C-o org-agenda-open-link

エントリーの中にあるリンクをフォローする。この機能は、参照されている Org-mode のノードに属しているテキストの中に含まれているいくつかのリンクの中から選択するという機能を提供する。もしもリンクが 1 つしかない場合は、選択画面を表示せずに、そこにリンクを貼る。

Change display

o 他のウインドウを削除します。

v d or short *d* org-aganda-day-view
v w or short *w* org-aganda-day-view
v m org-agenda-month-view
v y org-agenda-month-year
v SPC org-agenda-reset-view

日／週／月／年のビューを切り替えます。日または週にビューを切り替えたときは、この設定は、それに続くアジェンダの更新ついてのデフォルトの設定となります。月および年のビューは、作成するために時間を要するので、デフォルトとはしていません。数字の付いた前置引数をつけると、その年、ISO の週、月、年の指定した日に直接ジャンプします。例えば *32 d*と書いたときは 2 月 1 日、*9 w*と書いたら ISO の週番号が 9 を指します。日、週あるいは月のビューを設定したときは、1 年は同様に前置引数の中でコード化されます。例えば、*200712 w*と書いたときは 2007 年の第 12 週にジャンプするでしょう。もしもそのような年の指定を、1 桁もしくは 2 桁の数字で行いたいたときは、1938 年から 2037 年の間に位置づけられます。*v SPC*によって、org-agenda-spanでの設定をリセットすることができます。

f org-agenda-later

時間を前の日付の表示へと遡ります。

. org-agenda-goto-today

今日へ移動します。

j org-agenda-goto-date

日付の選択画面でその日に移動します。

J org-agenda-clock-goto

アジェンダバッファの中で現在時間を計測中のタスクに移動します。

D org-agenda-toggle-diary

日記のエントリーに含めるかどうかトグルします。参照 Section 10.3.1 [Weekly/daily agenda], page 91.

v l or short *l* org-agenda-log-mode

Logbook mode にするかどうかをトグルします。Logbook mode の中では、ログの取得中に (変数 org-log-done)DONE と印が付けられたエントリーが、その日の時刻を持っているエントリーとして、アジェンダの中に表示されます。org-agenda-log-mode-items変数を用いて log モードに含まれるエントリーのタイプを設定することができます。*C-u*という前置引数をつけて呼び出すと、状態の変化を含め、すべてのおこりうる logbook のエントリーを表示できるでしょう。*C-u C-u*という 2 つの前置引数をつけて呼び出すと、ログの情報のみが表示され、それ以外は表示されません。*v L*は、*C-u v l*と等価です。

`v [` or short `[` `org-agenda-manipulate-query-add`
現在のビューに、不活性のタイムスタンプを含めます。週／日のアジェンダとタイムラインビューのみです。

`v a` `org-agenda-archives-mode`
`v A` `org-agenda-archives-mode 'files`
Archives モードをトグルします。Archives モードでは、ARCHIVEDと印されたツリーもまたアジェンダを作成するときにスキャンされます。大文字の *A*を使用したときは、全てのアーカイブファイルを含みます。archives mode から出るためには、再度 *v a*を押してください。

`v R` or short `R` `org-agenda-clockreport-mode`
Clockreport モードをトグルします。Clockreport モードでは、日／週のアジェンダは、時間軸のための時刻のついた表を表示し、カレントのアジェンダビューでカバーされる範囲をファイルします。新しいアジェンダバッファの中で、このモードの初期設定は、`org-agenda-start-with-clockreport-mode`変数で設定することができます。このモードをトグル (すなわち *C-u R*) している時に、前置引数を使用することで、アジェンダフィルター[9] によって隠されているエントリーからの情報を表示しないでしょう。

`v E` or short `E` `org-agenda-entry-text-mode`
entry text mode をトグルします。entry text mode では、アジェンダ行によって参照されている Org-mode のアウトラインのノードから、多数の行が、その行の下に表示されるでしょう。最大の行数は、`org-agenda-entry-text-maxlines`変数で指定します。数値付きの前置引数を付けて、このコマンドを呼び出すと、前置引数の値の数によって、即座に修正されます。

`G` `org-agenda-toggle-time-grid`
時間のグリッドの表示をトグルします。`org-agenda-use-time-grid`と `org-agenda-time-grid`変数を参照してください。

`r` `org-agenda-rodo`
アジェンダバッファを再構築する。例えば、*S-left*と *S-right*を使って、アイテムのタイムスタンプを改修したあと、その変更を反映するために。そのバッファがグローバルな TODO リストの場合は、指定した TODO キーワードを選択できるリストを作成するために、前置引数を解釈します。

`g` `org-agenda-rodo`
カレントの Emacs のセッションにおいて、すべての Org-mode のバッファを保存します。あわせて ID の場所も。

`C-c C-x C-c` `org-agenda-columns`
アジェンダバッファの中でカラムビュー (see Section 7.5 [Column view], page 60) を作成します。カラムビューのフォーマットは、その時点のエントリーから作成され、あるいは (もしも、その時点でエントリーが存在しないなら)、アジェンダビューの最初のエントリーから作成されます。そのエントリーのためのフォーマットが何であれ、(プロパティーから、`#+COLUMNS`という行から、あるいは `org-columns-default-format`変数のデフォルトから作成された) オリジナルのバッファに存在しているエントリーのフォーマットがアジェンダで使用されます。

[9] ここではタグフィルターだけが有効です。工数のフィルターは無視されます。

C-c C-x > org-agenda-remove-restriction-lock

もしもファイルまたはサブツリーをその時点で制限しているならば、アジェンダをロックする制限を取り除きます。(see Section 10.1 [Agenda files], page 89).

Secondary filtering and query editing

/ org-agenda-filter-by-tag

タグおよび（または）工数の見積りに対して、カレントのアジェンダビューにフィルターをかけます。これとカスタムなアジェンダコマンドとの間の差異は、このフィルターが非常に早いということです。このため、あなたは、アジェンダ（注 1）を再表示することなく、異なるフィルターの間を素早く切り替えることができます。[10]

タグ選択の文字を入力しましょう。SPCはタグの全てを意味しています。入力部分で TAB を押すと、選択するタグの補完機能を使用できます (すべてのタグに選択用の文字が指定されているとはかぎりません)。そして、そのコマンドは、このタグを含んでいないか継承していないエントリーを全て隠します。前置引数をつけて呼び出した場合は、そのタグを持っているエントリーを削除さえしてしまいます。入力部で 2 番目の/はフィルターを終了し、隠されているエントリーを再度出現させます。もしも最初に入力したキーが、+または-ならば、前のフィルターは、選択された新たなタグの要求あるいは禁止に応じて、幅を狭くします。/の後に、+あるいは-を入力する代わりに、\ コマンドを即座に使用することもできます。

工数見積のフィルターをかけるために、予め認められている汎用的な工数を設定すべきです。例えば

```
(setq org-global-properties
    '(("Effort_ALL". "0 0:10 0:30 1:00 2:00 3:00 4:00")))
```

あなたは、<、>および=のひとつの操作を最初に入力することで、工数のためのフィルターをかけることができます。それから、あらかじめ認められた値のリストの中で、工数見積りのインデックスの数字を入力します。そこでは *0*は１０番目の値を意味します。フィルターは選択された値よりも、以下、イコール、以上であるかによって限定されます。もしも 0-9 のキーがタグへのアクセスキーとして使用されていないならば、単純にあなたは操作コマンドを利用することなく、直接インデックスとなる数字を入力するだけです。この場合<が仮定されます。操作のアプリケーションのために、定義された工数がないエントリーでは、org-sort-agenda-noeffort-is-high変数の値に従って取り扱われます。工数の定義のないタスクにフィルターをかけるには、?を操作の値として入力します。

Org-mode はまた、コンテクストに対応したタグのフィルターを自動的にサポートしています。もしも、org-agenda-auto-exclude-function変数の値が、ユーザが定義した機能に設定されているときは、その機能によって、どのようなタグがアジェンダから自動的に排除されるかを決定します。一度この機能が設定されると、それによって、/ コマンドは、*RET* をサブのオプションキーとして受け付け、自動的に排除ロジックを走らせます。例えば、いってみれば、ネットワークへのアクセスを必要とするタスクを定義するために Netというタグ、街での用事のために Errandというタグ、電話を掛けなければならないときに Callというタグを使用しているとします。あなたは、インターネッ

[10] カスタムコマンドによって、オプションとして org-agenda-filter-preset変数と結びつけることで、フィルターを事前にセットすることができます。このフィルターは、ビューに適用されます。そして、リフレッシュや 2 番目のフィルターを通して、基本的なフィルターとして存続します。このフィルターは、アジェンダのブロックの中で、アジェンダビュー全体のグローバルなプロパティです。この設定を行うためには、個別のブロックのセクションではなく、グローバルオプションのセクションで行います。

トを利用できるかどうか、仕事時間外にあるかどうか、このような状況に基づいて、これらのタグを自動的に排除することができるのです。

```
(defun org-my-auto-exclude-function (tag)
  (and (cond
        ((string= tag "Net")
         (/= 0 (call-process "/sbin/ping" nil nil nil
                             "-c1" "-q" "-t1" "mail.gnu.org")))
        ((or (string= tag "Errand") (string= tag "Call"))
         (let ((hour (nth 2 (decode-time))))
           (or (< hour 8) (> hour 21)))))
       (concat "-" tag)))

(setq org-agenda-auto-exclude-function 'org-my-auto-exclude-function)
```

\ org-agenda-filter-by-tag-refine

Narrow the current agenda filter by an additional condition. When called with prefix arg, remove the entries that *do* have the tag, or that do match the effort criterion. You can achieve the same effect by pressing `+` or `-` as the first key after the `/` command.

[ ] { }

in *search view*

新しい検索の単語 (*[*と*]*)、あるいは新しい正規表現 (*{*と*}*) をクエリー文字列に追加する。開いた角括弧／大括弧は、'+'という接頭辞のついたポジティブな検索用語を追加する。この検索用語は、必ずそのエントリーに発生／合致しなければならないことを示す。閉じた角括弧／大括弧は、ネガティブな検索用語を追加し、それは、選択されているエントリーの中で、絶対に発生／合致しないということである。

Remote editing

0-9 Digit argument.

C-_ org-agenda-undo

外部の編集コマンドでの変更を元に戻す。この変更はアジェンダバッファと外部のバッファの両方を元に戻す。

t org-agenda-todo

アイテムの TODO のステータスを変更する。アジェンダファイルでもオリジナルの Org ファイルでも有効である。

C-S-right org-agenda-todo-nextset
C-S-left org-agenda-todo-previousset

次／前の TODO キーワードのセットへと切り替える。

C-k org-agenda-kill

オリジナルの Org ファイルの中で、そのアイテムが属しているサブツリー全体と共に、カレントのアジェンダアイテムを削除する。もしも外部ファイルの削除するテキストが 1 行以上ならば、削除を行うには、ユーザーが指定する必要がある。`org-agenda-confirm-kill`変数を参照のこと。

C-c C-w org-agenda-refile
その時点でそのエントリーを差し替える。

C-c C-x C-a or short a org-agenda-archive-default-with-confirmation
org-archive-default-commandに設定されたデフォルトのアーカイブコマンドを使用して、その時点でエントリーに対応したサブツリーをアーカイブする。aキーを使用したときは、承認が必要である。

C-c C-x a org-agenda-toggle-archive-tag
カレントの見出しのための ARCHIVE タグをトグルする。

C-c C-x A org-agenda-archive-to-archive-sibling
カレントエントリーに対応したサブツリーを、アーカイブファイルに移動する。

C-c C-x C-s or short *$* org-agenda-archive
カレントの見出しに対応したサブツリーをアーカイブする。これは、設定されたアーカイブの場所に、多くの場合それは異なるファイルであるが、エントリーを移動することを意味している。

T org-agenda-show-tags
カレントアイテムと関連づけられたすべてのタグを表示する。もしも、あなたたがorg-agenda-show-inherited-tags機能を停止しているにもかかわらず、依然として、たびたび見出しのすべてのタグを確認したいというときに役に立つ。

: org-agenda-set-tags
カレントの見出しにタグを設定する。もしもアジェンダの中にアクティブなリージョンがあるときは、そのリージョンの中ですべての見出し用としてタグを変更する。

, カレントアイテムに優先順位を設定する。(org-agenda-priority) Org-mode は優先順位を表す文字を指示します。もしも、SPCを使って返答すると、優先順位のクッキーがそのエントリーから取り除かれる。

P org-agenda-show-priority
カレントアイテムの優先順位の重み付けを表示する。

+ or *S-up* org-agenda-priority-up
カレントアイテムの優先順位を高くする。優先順位はオリジナルのバッファで変更される。しかしアジェンダ上では並び替えの更新は行われない。このためには、*r*キーを使用する。

- or *S-down* org-agenda-priority-down
カレントアイテムの優先順位を低くする。

z or *C-c C-z* org-agenda-add-note
そのエントリーにのノートを追加する。このノートは記録され、ノートが置かれている状態を変更した同じ場所にファイルされる。org-log-into-drawerによって、これは引き出しの中に入る。

C-c C-a org-attach
すべてのコマンドの選択画面は、付属するものに関連づけられる。

C-c C-s org-agenda-schedule
このアイテムを予約する。前置引数をつけると、予約のタイムスタンプが削除される。

C-c C-d org-agenda-deadline
このアイテムにデッドラインを設定する。前置引数をつけるとデッドラインが削除される。

k org-agenda-action
カーソルの置かれた日付に選択されたアイテムの日付を設定するための、アジェンダのアクション。このコマンドはカレンダーでも動作する！コマンドは追加されたキーで入力する。

m
その地点でアクションのためにエントリーにマークする。複数のエントリーに対しても可能である。
Org-mode では次を伴う *C-c C-x C-k*.
d その時点の日付でマークされたエントリーのデッドラインを設定する。
s その時点の日付でマークされたエントリーを予約する。
r
デフォルトの日付としてカーソルの日付とともに org-captureを呼び出す。

アジェンダを更新した後に、*r*を押すと、コマンドの効果を確認できる。

S-right org-agenda-do-date-later
カレント行に関連づけられたタイムスタンプを 1 日先に変更する。数値付きの前置引数をつけると、その数字の日数分だけ先に変更する。例えば、*3 6 5 S-right*と入力すると 1 年先に変更される。*C-u*という前置引数をつけると、1 時間ずつ時間を変更する。もしもあなたが、同じコマンドを即座に繰り返したいときは、前置変数を付けなくても 1 時間単位で変化し続けるでしょう。二重の *C-u C-u*という前置引数をつけると、同様に分単位で変更される。オリジナルの Org-mode ファイルの中でタイムスタンプは変更されるが、その変更はアジェンダバッファには直接は反映されない。バッファを更新するには、*r*または *g*を使用する。

S-left org-agenda-do-date-earlier
カレント行のに関連づけられたタイムスタンプを 1 日過去に変更する。

> org-agenda-date-prompt
カレント行に関連づけられたタイムスタンプを変更する。>キーが選択される。というのは、私のキーボード上では *S-.*と同じだからである。

I org-agenda-clock-in
カレントアイテムの時計をスタートする。もしもすでに時計が動いているのならば、まずそれが停止する。

O org-agenda-clock-out
すでにスタートした時計を停止する。

X org-agenda-clock-cancel
カレントで動いている時計をキャンセルする。

J org-agenda-clock-goto
別のウインドウの中の動いている時計にジャンプする。

Bulk remote editing selected entries

m org-agenda-bulk-mark
大量のアクションについて、その時点でエントリーにマークをつける。前置引数を付けると、多くの連続したエントリーにマークをつける。

U org-agenda-bulk-remove-all-marks
大量のアクションのマークを取り除く。

U org-agenda-bulk-remove-all-marks
大量のアクションのためにマークがつけられたエントリーのマークを取り除く。

B org-agenda-bulk-action
大量のアクション。アジェンダの中ですべてのマークをつかられたエントリーについて実行する。この機能では、適用されるアクションを選択するために、別のキーを入力する。*B*に前置引数をつけると、*s*や *d*のコマンドをパスして、これらの特別なタイムスタンプをまとめて取り除く。

r
1つのリフィル上のターゲットに入力しすべてのエントリーを移動する。そのエントリーは
アジェンダ上には表示されなくなる。再表示 (*g*) によって再度表示される。
$ 選択されているエントリーをすべてアーカイブする。
A エントリーをアーカイブし、それぞれを所定のアーカイブ先に移動する。
t TODO の状態を変更する。これは TODO キーワード 1 文字を入力し、そして

選択されたエントリーすべての状態を変更する。それはブロックしているのを無視し
ログのノートを抑え込んで（タイムスタンプは別です)。
+ 選択されたエントリーのすべてにタグを付加する。
- 選択されたエントリーのすべてから、タグのひとつを削除する。
s
すべてのアイテムに新しい日付で予約する。すでに予約がついていれば、日数分だけ

日付を更新する。入力欄でプラスを 2 つつけて何かの数字を最初に打つことで。
例えば、‘++8d’とか ‘++2w’のように。
S
N 日を指定して、それぞれをリスケジュールする。N は入力欄で指定する。前置引数
(*C-u B S*) をつけることで、平日のみに指定できる。
d 指定した日をデッドラインとして設定する。

Calendar commands

c org-agenda-goto-calendar
Emacs のカレンダーを開き、アジェンダのカーソルの置かれている日付に移動します。

c org-calendar-goto-agenda
すでにカレンダーの中にあるときは、カーソルの置かれている日付で計算し、Org-mode のアジェンダを表示します。

i org-agenda-diary-entry
カーソルの置かれている日付および（ブロックエントリーでは）マークされた日付を使って、新しいエントリーを日記に書き込みます。この機能では Emacs の日記ファイル

[11] に追加することになります。ある意味では、カレンダーの `i`コマンドと似た機能です。日記ファイルは別のウインドウにポップアップし、そこでエントリーを書き加えることができます。

もしも Org-mode ファイルに `org-agenda-diary-file` を指定したならば、Org-mode ではそのファイルの中に（Org-mode の構文を使って）日記の代わりに、エントリーを作成することができます。ほとんどのエントリーは、日付を元にしたアウトラインのツリーの中に記述されており、あとで過去の月／年の中から予定をアーカイブするのを簡単にします。そのツリーは、`DATE_TREE`属性か、最上位のエントリーとして、年という属性を持ったエントリーのもとに構築されています。Emacs でエントリーのテキストを入力するようプロンプトが表示されるでしょう。もしもあなたがそれを指示するならば、さらなる連携なく、`org-agenda-diary-file`にそのエントリーを作成することになるでしょう。テキストを入力することなく、その入力欄で直接 `RET`を入力したら、そのターゲットとなるファイルがその場でのエントリーを終了させ、別のウインドウが表示されるでしょう。*k r*コマンドを参照してください。

M `org-agenda-phases-of-moon`
: その日を中心として 3ヶ月間の月齢を表示する。

S `org-agenda-sunrise-sunset`
: 日の出と日の入りを表示する。地理上の場所によって、カレンダーの変数が設定される。Emacs の calendar の章を参照のこと。

C `org-agenda-convert-date`
: カーソルの置かれている日付によって、多くの他の文化的・歴史的なカレンダーに変換する。

H `org-agenda-holidays`
: カーソルのある日付を中心に 3ヶ月間の祝祭日を表示する。

M-x org-export-icalendar-combine-agenda-files
: すべてのアジェンダファイルからエントリーを含んだ iCalendar 形式のファイルにエクスポートする。これはグローバルに利用できるコマンドで、そしてまたアジェンダメニューの中で利用できるコマンドです。

Exporting to a file

C-x C-w `org-write-agenda`
: アジェンダビューを 1 つのファイルに書き出します。選択したファイル名の拡張子に従って、そのビューは、HTML (拡張子が ‘`.html`’または ‘`.htm`’)、Postscript (拡張子 ‘`.ps`’)、PDF (拡張子 ‘`.pdf`’)、そしてプレーンテキスト (その他の拡張子) などにエクスポートされます。*C-u*という前置引数を用いてコマンドを呼び出したならば、即座に新しく作成されたファイルが開きます。エクスポートの間に使用されている ‘`ps-print`’および ‘`htmlize`’のためのオプションを設定するために、`org-agenda-exporter-settings` 変数を使用します。

Quit and Exit

q `org-agenda-quit`
: アジェンダを終了し、アジェンダバッファを削除します s 。

[11] `org-agenda-include-diary`が設定されているときは、このファイルはアジェンダ用に解析されます。

`x` `org-agenda-exit`

アジェンダを終了し、アジェンダバッファとアジェンダを編集するために Emacs で読み込まれたすべてのバッファを削除する。Org-mode ファイルを読み込むためにユーザーによって作成されたバッファは削除されない。

10.6 Custom agenda views

カスタムアジェンダコマンドは 2 つの目的を提供する。ひとつは TODO とタグの検索を使用して、保存と素早く頻繁にアクセスするため。もうひとつは、特別に合成したアジェンダバッファを作成するため。カスタムなアジェンダコマンドはデフォルトのコマンドと同様に、コマンド選択画面ディスパッチャー (see Section 10.2 [Agenda dispatcher], page 90) を通して利用できる。

10.6.1 Storing searches

カスタム検索の最初のアプリケーションは、よく使われる検索式のためのキーボードショートカットを定義することです。それはアジェンダバッファの作成、またはツリーの抽出（後者は言うまでもなくカレントバッファのみをカバーする）のどちらに対してでも。カスタムコマンドは、`org-agenda-custom-commands`変数で設定されます。あなたはこの変数をカスタマイズできます。例えば、*C-c a C*というように。またあなたは‘`.emacs`’に Emacs の Lisp を記述して直接設定することもできます。以下に述べる例はすべての適正な検索タイプを含んでいます。

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING")
        ("W" todo-tree "WAITING")
        ("u" tags "+boss-urgent")
        ("v" tags-todo "+boss-urgent")
        ("U" tags-tree "+boss-urgent")
        ("f" occur-tree "\\<FIXME\\>")
        ("h" . "HOME+Name tags searches") ; description for "h" prefix
        ("hl" tags "+home+Lisa")
        ("hp" tags "+home+Peter")
        ("hk" tags "+home+Kim")))
```

それぞれのエントリーの頭文字は、コマンドにアクセスするために、コマンド選択画面を呼び出す *C-c a*というコマンドの後に、入力しなければならないキーを定義します。通常、これは 1 文字をあてますが、もしもあなたが似たようなコマンドをたくさん持っていたら、あなたは 2 文字の組合せで定義することができます。その場合、いくつかの組合せでは最初の文字が同じものとなり、前置引数[12]. と同じように提供されます。2 番目のパラメーターは検索の種類を示し、マッチさせるために使われる文字列や正規表現がそれに続きます。上の例ではそれゆえ以下のように定義します。

C-c a w TODO のキーワードとして、‘`WAITING`’となっている TODO エントリーのためのグローバルな検索として。す。

C-c a W 同じような検索であるが、カレントバッファにのみ適用され、ツリーの抽出として検索結果を表示する。

C-c a u ‘`:urgent:`’ではなく‘`:boss:`’というタグがつけられた見出しのための、グローバルなタグ検索を行う。

[12] あなたは前置引数と説明をつけて、コンソールのセルを挿入することで、前置引数のキーのための説明を表示することができます。

C-c a v *C-c a u*と同じ検索を行うが、TODO アイテムである見出しに対してのみ検索を行うという制限がある。

C-c a U *C-c a u*と同じ検索を行うが、カレントバッファに対してのみ検索を行い、結果をツリーの抽出として表示する。

C-c a f すべてのエントリーのうちで'FIXME'という言葉を含んでいるものを検索してツリーの抽出を行う（くどいかもしれませんが、カレントバッファだけが対象です)。

C-c a h HOME というタグ検索のためのコマンドの前置引数として、そこでは、タグ検索の追加として、一つの名前 (Lisa、Peter、または Kim) を選択するために、あなたはさらに (*l*、*p*、または *k*) というキーを追加入力する必要があります。

10.6.2 Block agenda

もう一つの可能性とは、アジェンダビューの構築です。そのビューは、様々なコマンドの結果で構成されており、それぞれのコマンドはアジェンダバッファの中の 1 つのブロックを作成します。利用できるコマンドは（*C-c a a*を実行して作成された）一日または週間アジェンダのための agenda、（*C-c a t*を実行して作成された）グローバルな todo リストのための alltodo、そして上で議論してきた todo、tags、tags-todoなどの検索コマンドに含まれています。２つの例を挙げます。

```
(setq org-agenda-custom-commands
      '(("h" "Agenda and Home-related tasks"
         ((agenda "")
          (tags-todo "home")
          (tags "garden")))
        ("o" "Agenda and Office-related tasks"
         ((agenda "")
          (tags-todo "work")
          (tags "office")))))
```

これによって、家で精を出さなければならない用事に対するマルチブロックのビューを作成するために、*C-c a h*を定義します。アジェンダバッファには結果として、その週の、'home'というタグが含まれているすべての TODO アイテムと、'garden'というタグがついたすべての行のためのアジェンダを含むことになります。最後に、*C-c a o*というコマンドで、同様に、オフィスの作業についてのビューを得ることができます。

10.6.3 Setting options for custom commands

Org-mode はたくさんのアジェンダの構築や表示について調整する変数を含んでいます。グローバルな変数では、カスタムコマンドも含めて、アジェンダの全てのコマンドの動作を定義することができます。しかしながら、もしもあるひとつのカスタムビューについて、いくつかの設定を変更したいならば、それも可能です。オプションの設定は変数名のリストに書き込むことが必要で、org-agenda-custom-commandsの中に、正しい位置に値を書き込む必要があります。例えば。

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING"
         ((org-agenda-sorting-strategy '(priority-down))
          (org-agenda-prefix-format "  Mixed: ")))
        ("U" tags-tree "+boss-urgent"
         ((org-show-following-heading nil)
          (org-show-hierarchy-above nil)))
        ("N" search ""
         ((org-agenda-files '("~org/notes.org"))
          (org-agenda-text-search-extra-files nil)))))
```

こう書き込むことによって、*C-c a w*というコマンドは、優先順位によってのみ収集したエントリーを並べ替えるでしょう。そのエントリーのカテゴリを設定する変わりに、例えば‘ Mixed: ’という文字を prefix の形で書くことで変更することができます。*C-c a U*というタグでツリーを抽出するコマンドは、この結果、超コンパクトとなるでしょう。なぜならば、検索に合致した項目の上の階層の見出しも、合致した項目の見出しもどちらも表示されないからです。*C-c a N*というコマンドは、1 つのファイルに制限されたテキスト検索を実行します。

ブロックアジェンダを作成するコマンドセットのために、org-agenda-custom-commandsではオプションの設定用に 2 つの別の場所を用意しています。その設定の中にたったひとつのコマンドに有効なオプションを付け加えることも、その設定の中にすべてのコマンドに有効なオプションを付け加えることもできます。前者のオプションは 1 つのコマンドエントリーを付け加える。後者のオプションは、コマンドエントリーのリストを書き込むことが必要です。ブロックアジェンダの例に戻ると (see Section 10.6.2 [Block agenda], page 110)、*C-c a h*というコマンドで、並べ替えの順序を優先順位の降順 priority-downに変更することができますし、その中で「GARDEN」というタグのついたものについては反対の順序、すなわち優先順位の昇順 priority-upに並べ替えることができるでしょう。このことは以下のように記述できます。

```
(setq org-agenda-custom-commands
      '(("h" "Agenda and Home-related tasks"
         ((agenda)
          (tags-todo "home")
          (tags "garden"
                ((org-agenda-sorting-strategy '(priority-up)))))
         ((org-agenda-sorting-strategy '(priority-down))))
        ("o" "Agenda and Office-related tasks"
         ((agenda)
          (tags-todo "work")
          (tags "office")))))
```

おわかりだと思いますが、変数とカッコで囲んでいる設定はやや複雑なところがあります。わかりにくいときは、カスタマイズのインターフェースとしてこの変数を設定してください。これはカスタマイズの構造を完全にサポートしています。注意しなければならないのは、このインターフェースでオプションを設定するときに、変数は、Lisp による表現をとっているということです。そのため、もしもその変数が 1 つの文字ならば、あなた自身でその変数の値に「”(ダブルクォート)」で囲む必要があるということです。

10.7 Exporting Agenda Views

もしもあなたが自分のコンピュータから離れているときは、いくつかのアジェンダのバージョンを印刷して持ち歩くことは大変役に立ちます。Org-mode はカスタムアジェンダビューをプレーンなテキスト、HTML[13]、Postscript、PDF[14]、iCalender ファイルとしてエキスポートすることができます。もしも、ときどきこのようなことを実行するのならばコマンドを使用しましょう。

C-x C-w `org-write-agenda`

アジェンダビューを 1 つのファイルに書き出します。選択したファイル名の拡張子により、そのビューは HTML（拡張子が‘`.html`’または‘`.htm`’）、Postscript（拡張子が‘`.ps`’）、iCalendar（拡張子が‘`.ics`’）、あるいはプレーンなテキスト（何かほかの拡張子）としてエクスポートされます。エクスポートの間に、‘`ps-print`’のため、および‘`htmlize`’のためにオプションを設定するには、`org-agenda-exporter-settings`変数を使用します。例えば

```
(setq org-agenda-exporter-settings
      '((ps-number-of-columns 2)
        (ps-landscape-mode t)
        (org-agenda-add-entry-text-maxlines 5)
        (htmlize-output-type 'css)))
```

もしも、あなたがアジェンダビューをたびたびエクスポートする必要があるのならば、アウトプットのファイルの名前[15] のリストに、いくつかのカスタムなアジェンダのコマンドを関連づけることができます。ここに一つの例があります。最初のものはアジェンダとグローバルな TODO リストに対するカスタムなコマンドを定義しており、それらをエクスポートするたくさんのファイルと一緒になっています。それから 2 つのブロックアジェンダコマンドを定義し、同様にそれらのためのファイル名を指定しています。ファイル名は、現在作業しているディレクトリに対して相対パスにすることも絶対パスにすることもできます。

```
(setq org-agenda-custom-commands
      '(("X" agenda "" nil ("agenda.html" "agenda.ps"))
        ("Y" alltodo "" nil ("todo.html" "todo.txt" "todo.ps"))
        ("h" "Agenda and Home-related tasks"
         ((agenda "")
          (tags-todo "home")
          (tags "garden"))
         nil
         ("~/views/home.html"))
        ("o" "Agenda and Office-related tasks"
         ((agenda)
          (tags-todo "work")
          (tags "office"))
         nil
         ("~/views/office.ps" "~/calendars/office.ics"))))
```

[13] あなたは Hrvoje Niksic 氏の‘`htmlize.el`’をインストールする必要があります。

[14] PDF の出力を作成するためには、Ghostscript の‘`ps2pdf`’ユーティリティがシステムにインストールされている必要があります。pdf ファイルを選択するとポストスクリプトファイルも作成されます。

[15] もしもあなたが週間アジェンダやグローバルな TODO リストなどのような標準的なビューを保存したいならば、ファイル名を指定することができるようにするために、それらのビューのためにカスタムなコマンドを定義する必要があります。

ファイル名の拡張子がエクスポートのタイプを決定します。もしも拡張子が‘.html’ならば、Org-mode は‘htmlize.el’パッケージを使用し、バッファを HTML に変換し、そのファイル名で保存します。もしも拡張子が‘.ps’ならば、ps-print-buffer-with-facesが Postscript の出力をするために使用されます。もしも拡張子が‘.ics’ならば、iCalendar のエクスポートは、アジェンダを構成しているすべてのファイルにわたってエクスポートを実行し、現在アジェンダの中ではリスト化されたエントリーのエクスポートに限定されます。ほかの拡張子がついた場合は、プレーンな ASCII テキストファイルが作成されます。

エクスポートファイルは、非常に負荷が高いので、これらのコマンドの一つを相互に影響するように使用している時は、出力されません。そのかわり、1 ステップですべての指定されたファイルを出力する特別なコマンドが用意されています。

C-c a e org-store-agenda-views
: アジェンダに関連するエクスポートファイル名を持つすべてのアジェンダビューをエクスポートします。

あなたは、エクスポートコマンドのためのオプションの設定をするために。カスタムアジェンダコマンドのオプションのセクションを使用することができます。例えば、

```
(setq org-agenda-custom-commands
      '(("X" agenda ""
         ((ps-number-of-columns 2)
          (ps-landscape-mode t)
          (org-agenda-prefix-format " [ ] ")
          (org-agenda-with-colors nil)
          (org-agenda-remove-tags t))
         ("theagenda.ps"))))
```

このコマンドは、Postscript のエクスポートのために、2つのオプションを設定します。横長のフォーマットで2段のプリントを作成するためです。出力されたページは、2つにカットして、紙のアジェンダとして使えるようになります。もうひとつの設定は、行頭のカテゴリーとスケジューリング情報を省き、その代わりにチェックのついてないチェックボックスの項目となるようにアジェンダを修正します。私たちは各行をコンパクトに表示するためにタグを省略したり、白黒プリンタのためにカラーを使わない用にすることもできます。org-agenda-exporter-settingsの中で指定する設定もできますが、org-agenda-custom-commandsでの設定が優先します。

コマンドラインで次のような設定を使用することができます。

```
emacs -f org-batch-store-agenda-views -kill
```

また、いくつかのパラメーター[16] を修正する必要があります。

```
emacs -eval '(org-batch-store-agenda-views                      \
              org-agenda-span month                             \
              org-agenda-start-day "2007-11-01"                 \
              org-agenda-include-diary nil                      \
              org-agenda-files (quote ("~/org/project.org")))'  \
      -kill
```

どちらも‘~/org/project.org’のファイルを対象として、日記のエントリーは除かれ、30 日以内に限定したアジェンダビューを作成します。

[16] 引用の方法はあなたの使用しているシステムに依存します。事例用の FAQ を確認してください。

あなたは、他のプログラムで将来の進行過程を認める方法で、アジェンダの情報を絞り込むことができます。詳細は Section A.8 [Extracting agenda information], page 195, のノートの情報を参照してください。

10.8 Using column view in the agenda

カラムビュー (see Section 7.5 [Column view], page 60) は、Org-mode ファイルの階層構造の中に組み込まれている属性を見たり編集したりするために通常は使われます。エントリーがある評価基準で収集されているアジェンダから、カラムビューを使用することは大変便利です。

C-c C-x C-c `org-agenda-columns`
: アジェンダの中でカラムビューに切り替えます。

この属性がどのようなものか理解するために、アジェンダのエントリーはもはや適切なアウトラインの環境ではなくなることを理解することが重要です。これによって以下のようなことが生じます。

1. Org-mode では、どの `COLUMNS`のフォーマットを使用するか、決定する必要があります。アジェンダの中のエントリーは、異なるファイルから集められるということと、ファイルが異なると `COLUMNS`のフォーマットも異なるということから、このことは些細な問題であるとはいえないのです。Org-mode は最初に、`org-overriding-columns-format`変数がカレントで設定されているかどうか、またそこからフォーマットを取り出すことができるかどうかチェックします。一方、アジェンダの最初のアイテムに関連したフォーマットを使用するか、もしもそのアイテムが特別なフォーマット（属性もしくはファイルの中で定義された）を持たないならば、`org-columns-default-format`を使用します。
2. もしも、どれかカラムに要約形式 (see Section 7.5.1.2 [Column attributes], page 60) が定義されているならば、アジェンダでカラムビューに切り替えるときに、すべての関連するアジェンダファイルを確認して、この属性の計算の更新を確実に行います。このことは、特別な `CLOCKSUM` の属性が真であると設定されているということです。Org-mode はアジェンダの中で表示された値を合計するでしょう。一日／週間アジェンダの中で、合計は 1 日をカバーしています。他のビューでは、ブロック全体をカバーするのです。アジェンダでは同じエントリーを 2 度表示したり（例えばスケジュールと期限というように)、同じ階層（例えば親と子）から 2 つのエントリーを表示したりするかもしれない、ということを理解することは重要なことです。これらの場合、アジェンダの中での要約は、いくつかの値が二重にカウントされるという間違った結果を導く可能性があります。
3. アジェンダの中のカラムビューが、`CLOCKSUM`を表示するときは、このアイテムのためにいつでも時間計測全体に対応します。そのため 1 日/週間アジェンダにおいて、カラムビューでリスト化された時間合計は、カレントのビューの外側の時間から発生することになるかもしれません。この機能によって、あるタスクについて、計画された総工数を 1 つのカラムにリストにして、その値を比較することができるので、優位性を持ちます。この機能はアジェンダのカラムビューにおける重要なアプリケーションのひとつです。もしもあなたが表示されている期間の中の作業時間についての情報を得たいならば、clock table mode（*R*をアジェンダの中で入力する）を使用してください。

11 Markup for rich export

Org-mode の文書をエクスポートする時，エクスポート機能は文書の構造をできるだけ正確に反映しようとします．HTML や LaTeX，DocBook，その他のリッチフォーマット等のエクスポートの対象について，Org-mode は文書をリッチエクスポートに変換するルールを持ちます．このセクションは Org-mode んバッファで使われるマークアップのルールについて説明します．

11.1 Structural markup elements

Document title

エクスポートされた文書のタイトルは専用の行で設定されます．

```
#+TITLE: これは文書のタイトルです．
```

もしこの行が存在しなければ，タイトルはバッファ中の最初の空でない，コメントでない行を用います．もしまだ何も存在していない，またはあなたが最初の見出しより前のテキストをエクスポートをしないよう設定していたら，タイトルは拡張子無しのファイル名となります．

もしあなたがリージョンでマークしたサブツリーのみをエクスポートしているなら，サブツリーの見出しは文書のタイトルとなるでしょう．もしサブツリーが `EXPORT_TITLE`プロパティを持っているなら，そのプロパティの値が優先して用いられるでしょう．

Headings and sections

Chapter 2 [Document Structure], page 6 で説明されているような文書のアウトライン構造はエクスポートされた文書のセクションの定義の基準を形成しています．しかしながら，アウトライン構造はまた (例えば) タスクのリストとしても使われているので，最初の 3 アウトラインレベルのみ見出しとして使われます．

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

Table of contents

目次は通常ファイルの最初の見出しの前に直接挿入されます．もしあなたが異なる場所に目次を挿入したいのなら，その場所に `[TABLE-OF-CONTENTS]`文字列を書いてください．目次の深さはデフォルトでは見出しのレベルの数と同じですが，`org-export-with-toc`変数を設定するか，ファイルに以下のように書くことによって，あなたはこれより小さな値に変更することも，目次を完全に表示させないようにすることも可能です．

```
#+OPTIONS: toc:2          (目次に表示するレベルを 2 までとする)
#+OPTIONS: toc:nil        (目次を表示しない)
```

最初の見出しより前のテキスト

Org-mode は通常最初の見出しの前にテキストをエクスポートし，最初の行を文書のタイトルにします．テキストは完全にマークアップされているでしょう．もしあなたが HTML や LaTeX，DocBook のような<リテラルを含めたい場合，独立したエクスポート機構のセクションで説明されている特別な構造を使います．

多くの人々は内部リンクの設定のためとそのために異なる方法でエクスポートされた最初の見出しの前のテキストを制御する最初の見出しの前に空白を使うことを好みます．あなたは `orgexport-skip-text-before-1st-heading`変数を `t`にすることで設定することができます．ファイル中に設定する場合，あなたは '`#+OPTIONS: skip:t`'とすることで同等の設定を行うことができます．

もし，あなたがまだ最初の見出しの前にテキストを置きたいのであれば，`#+TEXT`構造を使います:

```
#+OPTIONS: skip:t
#+TEXT: このテキストは*最初の*見出しの前に置かれます
#+TEXT: [TABLE-OF-CONTENTS]
#+TEXT: このテキストは目次と最初の見出しの間に置かれます
```

Lists

Section 2.7 [Plain lists], page 12 で説明されているプレーンリストは，バックエンドのリストに変換されます．多くのバックケンドがサポートしているのは記号付きリスト，番号付きリスト，見出し付きリストです．

段落，改行，引用

段落は最低 1 つの空白行で区切られます．もしあなたが強制的に段落の中で改行しないなら，‘\\\\’を行の末尾に書いてください．

リージョンで改行を保つためには，しかしそうでなければ通常のフォーマットが使われるなら，あなたはフォーマット技法として使われるこの構文を使うことができます．

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

     -- AlexSchroeder
#+END_VERSE
```

別の文書から一節を引用する時，段落の左右の余白を空けることが慣習となっています．あなたは以下を用いることで引用を Org-mode の文書に含めることができます:

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

もしあなたがテキストを中央寄せにしたいなら，以下を使うことができます:

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

Footnote markup

脚注は Section 2.10 [Footnotes], page 15 で説明されたように定義されていて，全てのバックエンドにエクスポートされます．Org-mode は同じノートに対しての複数の参照と異なるバックエンドをサポートします．

Emphasis and monospace

あなたは***bold***と*/italic/*，_underlined_，`=code=`，`~verbatim~`，そして必要なら‘`+strile-through+`’を単語に適用することができます．code と verbatim 文字列の中のテキストは Org-mode の明確な構文ではありません; それは verbatim にエクスポートされます．

Horizontal rules

少なくとも 5 文字のダッシュ文字のみで行成される線は水平線 (HTML では '`<hr/>`', LaTeX では `\\hrule`) にエクスポートされます.

コメント行

行頭の文字が '`#`'から始まる行はコメントとして扱われ, エクスポートされません. もしあなたがコメント行をインデントしたいのであれば, '`#+`'から行を開始してください. '`COMMENT`'ワードを持つサブツリーは, サブツリー全体がエクスポートされません. 最後に, '`#+BEGIN_COMMENT`'から '`END_COMMENT`'で囲まれた範囲はエクスポートされません.

C-c ; エントリー先頭の COMMENT キーワードをトグルします.

11.2 画像と表

Org-mode ネイティブなテーブル (see Chapter 3 [Tables], page 18) と '`table.el`'パッケージを用いたテーブルの両方が適切にエクスポートされます. Org-mode の表では, 最初の水平線の前の行が表のヘッダ行となります. あなたはキャプションと相互参照の指定を表の直前に, 参照のための `\\ref{tab:basic-data}`オブジェクトをテキストのどこかに書くことができます.

```
#+CAPTION: これは次の表 (またはリンク) のキャプションです
#+LABEL:   tbl:basic-data
   | ... | ...|
   |-----|----|
```

多くのバックエンド (HTML, LaTeX, DocBook) はエクスポートされた文書の中に直接画像を挿入することができまる. もし, 例えば, `[[./img/a.jpg]]`のような説明部分を持たない画像ファイルへのリンクがあるなら, Org-mode は画像の挿入を行います. もしあなたが画像のキャプションや内部相互参照のラベルを定義したいなら, 以下のように`#+CAPTION`と`#+LABEL`をリンクの前に書きます:

```
#+CAPTION: これは次の画像 (または表) のリンクのキャプションです.
#+LABEL:   fig:SED-HR4049
[[./img/a.jpg]]
```

あなたは画像に対する追加要素を定義するかもしれません. これはバックエンドの仕様なので, さらに情報が必要なら独立したバックエンドについてのセクションを見てください.

See Section 4.4 [Handling links], page 35.

11.3 Literal examples

あなたはマークアップに依存しないリテラルの例を含めることができます. そのような例に等幅のタイプセットがあり, それはソースコードやそれに似た例向きです.

```
#+BEGIN_EXAMPLE
テキストファイルからの例.
#+END_EXAMPLE
```

そのようなブロックはインデントされたテキストをうまく整列させるためと, 特にプレーンリスト構造 (See Section 2.7 [Plain lists], page 12.) のためにインデントされるでしょう. 小さな例を使う時, それを簡単にするために, あなたはコロンとそれに続く空白からなる例の行を使うことができる. それらはコロンの前に空白を追加することもできます.

```
ここに例を書きます
   : テキストファイルからの例
```

もし例がソースコードなら，もしくはEmacsでフォントロックによりマークアップされたテキストなら，あなたはEmacsバッファ[1]を要塞化するように要請することができます．あなたが例に色付けするために使うメジャーモードの名前を指定することが必要な時，‘src’ブロックを使います:

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

exampleとsrcスニペットでは，あなたはBEGINの行の最後に-nを追加することで，例の行番号を表示することができます．もしあなたが+nとすると，前のスニペットから現在のものに番号が引き継がれます．リテラルの例で，Org-modeは‘(ref:name)’をラベルとして解釈し，[[(name)]]のような特別なリンクによりそこを参照することができます (i.e. 参照名は1つの括弧に囲まれています)．HTMLでは，対応するコード行をマウスオーバーすると自動的にハイライト表示になり，少しクールです．

また，ソースコード[2]からラベルを消去するかどうかの切り替えのために-rを追加することもできます．-nで切り替えると，リンクされるそれらのリファレンスはコードリスティングの行番号によってラベルを付けられ，そうでなければ括弧無しのラベルにリンクされます．

```
#+BEGIN_SRC emacs-lisp -n -r
(save-excursion                  (ref:sc)
   (goto-char (point-min))       (ref:jump)
#+END_SRC
In line [[(sc)]] we remember the current position.  [[(jump)][Line (jump)]]
jumps to point-min.
```

もし，ラベルの構文が言語の構文と衝突した場合，-lを使うことで‘#+BEGIN_SRC pascal -n -r -l \"((%s))\"’のようにフォーマットを変更できます．org-coderef-label-format変数を見てください．

HTMLはエクスポート時にテキストエリア,See Section 12.5.7 [Text areas in HTML export], page 130. とすることができます

C-c ' カーソル位置のソースコード例をそのネイティブモードで編集します．これはソースコードを一時バッファに表示し，切り替えることで働きます．あなたは *C-c '* をもう一度押

[1] HTMLバックエンドに対しては，この作業は自動的に行われます (Org-modeと一緒に配布されている‘htmlize.el’のバージョン1.34が必要です)．LaTeXの要塞化されたコードの塊はリスティングか，minted (http://coe.google.com/p/minted) パッケージによってアーカイブされます．リスティングを使うには，org-export-latex-listings変数をオンにし，LaTeXのヘッダにリスティングパッケージが含まれているようにします (例: org-export-latex-packages-alistの設定とを使います)．色付きの出力を含む設定のオプションについて，リスティングのドキュメントを見てください．mintedを使うには，pygemnts (http://pygemnts.org) プログタムをインストールする必要があり，org-export-latex-mintedを追加で設定し，LaTeXのヘッダにmitedパッケージが含まれていることと-shell-escapeオプションが‘pdflatex’に引き継がれている (org-latex-to-pdf-processを見てください) ことを確認します．

[2] Org-modeの例で説明するのに便利なリンクに行番号を使う間，-kを-n -rに追加することでソースコードのラベルを維持します．

すことで編集を終了します[3] は artist-mode[4] によって編集されます．空行でこのコマンドを使うことで，新しい固定幅のリージョンを作成します．

C-c l *C-c '*によって作成した一時バッファでのソースコード例の編集中に org-store-link の呼び出しはラベルを指示します．現在のバッファがユニークであることを確認し，現在の行の最後に '(ref:label)'のように適切にフォーマットされたものが挿入されます．ラベルは '(label)'のようなリンクを記憶し，*C-c C-l*検索する．

11.4 Include files

エクスポート中，あなたは別のファイルの内容をインクルードすることができます．例えば，'.emacs' をインクルードするなら，あなたは次のようにします:

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

2 つ目のオプションは (e.g. 'quote'や 'example', 'src') のようなマークアップで，3 つ目はマークアップが 'src'ならコンテンツの言語を表します．マークアップはオプションです; もし与えられなければ，Org-mode フォーマットのテキストと仮定される．インクルードの行は最初の行とそれに続く行のプレフィクスの指定のための追加キーワードパラメータの:prefix1と:prefixを，Org-mode のコンテンツを指定したレベル下げるための:minilevelを，同様に選択したマークアップ固有のオプションを持ちます．例えば，ファイルをインクルードするには:

```
#+INCLUDE: "~/snippets/xx" :prefix1 "   + " :prefix "     "
```

:lineパラメータを使うことで，ファイルの指定した範囲の行のみをインクルードすることができます．範囲外の行はインクルードされません．範囲の開始と，または終了は明らかにデフォルトを使いません．

```
#+INCLUDE: "~/.emacs" :lines "5-10"    Include lines 5 to 10, 10 excluded
#+INCLUDE: "~/.emacs" :lines "-10"     Include lines 1 to 10, 10 excluded
#+INCLUDE: "~/.emacs" :lines "10-"     Include lines from 10 to EOF
```

C-c ' ポイント位置のインクルードされたファイルに移動します．

11.5 Index entries

あなたは公開した文書のインデックスに用いるエントリーを規定することができます．これは#+INDEX から始まる行により設定します．感嘆符を含むエントリーはサブアイテムを作るでしょう．さらなる情報を見るには Section 13.1.8 [Generating an index], page 148 を参照してください．

```
* Curriculum Vitae
#+INDEX: CV
#+INDEX: Application!CV
```

11.6 Macro replacement

あなたはこのようにして

```
#+MACRO: name   replacement text $1, $2 are arguments
```

which can be referenced anywhere in the document (even in code examples) with {{{name(arg1,arg2)}}}. In addition to defined macros, {{{title}}}, {{{author}}}, etc., will reference information set by the #+TITLE:, #+AUTHOR:, and similar lines. Also,

[3] 固定幅のリージョン (それぞれの行はコロンとスペースから始まります)

[4] あなたは異なる org-edit-fixed-width-region-mode変数により，モードを選ぶこともできます．

`{{{date(FORMAT)}}}` and `{{{modification-time(FORMAT)}}}` refer to current date time and to the modification time of the file being exported, respectively. *FORMAT* should be a format string understood by `format-time-string`.

マクロ展開はエクスポート中に行われ，一部の人々は複雑な HTML コードの構築に用いる．

11.7 Embedded LaTeX

プレーンな ASCII はほとんどの場合ノートをとるのに十分です．例外は数学の記号や時々出てくる数式を必要とする科学に関するノートのようなものです．LaTeX[5] は科学に関する文書の組版に広く使われています．多くの academics は LaTeX のソースコードの読み書きに使われていて，すぐに多くのエクスポートバックエンドに対応できるため，Org-mode は LaTeX コードのファイルへの組込みをサポートしています．

11.7.1 Special symbols

あなたは LaTeX マクロをギリシャ文字を表す '`\\alpha`'や矢印を表す '`\\tpo`'のような特殊記号の挿入に使うことができます．これらのマクロは補完が可能で，'`\\`'まで入力し，その後に何文字か入力して *M-TAB*を押すことで補完が可能です．LaTeX のコードとは違い，Org-mode は数学の区切り文字を囲まないようなマクロも使うことができます．以下に例を挙げます:

```
Angles are written as Geek letters \\alpha, \\beta and \\gamma.
```

エクスポート時，これらのシンボルはエクスポート先のネイティブフォーマットに変換されます．HTML では`\\alpha`のような文字列は`&alpha;`にエクスポートされ，LaTeX では`$\\alpha$`となります．同様に，`\\nbsp`は HTML では` `に，LaTeX では`~`となります．もしあなたが記号を単語の中に含めたいのであれば，次のようにします: '`\Aacute{}stor`'.

非常に多くのエンティティが提供されていて，HTML と LaTeX からその名前を引き継いでいます; 完全なリストは `org-entities`変数を見てください．'`\\-`'はシャイなハイフンとして扱われていて，'`--`'や '`---`'，'`...`'は異なる長さのハイフンかドットの集合を作成するための全て特殊コマンドに変換されます．

もしあなたが UTF-8 文字でエンティティを表示したいのなら，以下のコマンド[6]:

*C-c C-x * エンティティの UTF-8 での表示をトグルします．これはバッファの内容を変更せず，UTF-8 の文字を表示するためにオーバレイを用いています．

11.7.2 Subscripts and superscripts

LaTeX と同じように，'`^`'と '`_`'が下付き文字と上付き文字を示しています．さらに，それらは math-mode にの区切り文字に組込まずに使うことができます．ASCII テキストの可読性の向上のため，複数文字の下付き文字と上付き文字を波括弧で囲む必要はありません (囲んでもかまいませんが)．例

```
The mass of 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.
```

上付きテキスト，下付きテキストの説明を避けるため，あなたはバックスラッシュをつけた '`^`'と '`_`'を引用できます: '`\\^`'と '`\\_`'です．異なる文脈でしばしば使われるアンダーラインのテキストを書くなら，常にこれらの下付き文字として解釈する Org-mode の慣習はあなたのやり方で得ることが

[5] LaTeX はドナルド・クヌースの TeX システムを基としたマクロシステムです．"LaTeX" で説明される多くの機能は TeX からのものですが，違いはそれほどありません

[6] あなたは `org-pretty-entities`変数または`#+STARTUP`オプション `entitiespretty`にデフォルトを設定することができます

できます．この慣習を変更するには `org-export-with-sb-superscripts`変数を設定するか，ファイルに次のように書いてください．

```
#+OPTIONS: ^:{}
```

With this setting, ‘`a_b`’ will not be interpreted as a subscript, but ‘`a_{b}`’ will.

`C-c C-x \` さらに UTF-8 のエンティティを見るため，このコマンドは下付き文字と上付き文字を WYSISYM で形成する．

11.7.3 LaTeX の断片的なコード

シンボルと上付き，下付き，完全な式を越えることが必要です．Org-mode は LaTeX の数式を含むことができ，各エクスポート先への変換もサポートしています．LaTeX にエクスポートするとき，コードは明らかに残っています．HTML へエクスポートするとき，Org-mode は数式[7] の処理と描画のために MathJax library (`http://www.mathjax.org`) (see Section 12.5.6 [Math formatting in HTML export], page 129) を呼び出します．最後に，数式表現はブラウザか DocBook 文書で描画可能な画像[8] へと処理されます．

LaTeX のコード片は，特別なマークは全く必要ありません．以下のコード片は LaTeX のソースコードとして知られています:

- あらゆる種類の環境[9]．唯一必要なことは`\\begin`文は空白のみがある行に表示されることです．
- 通常の LaTeX の数学の区切り文字内部のテキスト．流通仕様との衝突を避けるために，囲まれたテキストに最大 2 つの改行が含まれている場合，‘`$`’文字は数学区切り文字のみとして認識され，‘`$`’文字がの間に空白がない，そして

For example:

```
\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 \[ a=-\sqrt{2} \].
```

もしあなたが他の目的に ASCII の区切り文字が必要なら，LaTeX コンバータに邪魔されえることを望まない文字を除外するために `org-format-latex-options`オプションを設定することができます．

LaTeX の処理は `org-export-with-LaTeX-fragments`変数を設定することができます．デフォルトの設定は `t`で，HTML には‘`MathJax`’を用い，DocBook と ASCII，LaTeX では処理しません．あなたはこの変数をファイルの冒頭部分に書くことで設定することもできます:

`#+OPTIONS: LaTeX:t`	Do the right thing automatically (MathJax)
`#+OPTIONS: LaTeX:dvipng`	Force using dvipng images
`#+OPTIONS: LaTeX:nil`	Do not process LaTeX fragments at all
`#+OPTIONS: LaTeX:verbatim`	Verbatim export, for jsMath or so

---

[7]

[8] これを行うには，あなたのシステムに LaTeX をインストールする必要があります．そしてまた，`http://sourceforge.net/projects/dvipng/`で入手できる‘`dvipng`’プログラムも必要です．

[9] ‘`MathJax`’が使われている時，‘`MathJax`’によって認識されている環境が処理されます．‘`dvipng`’を画像の生成に用いる時，LaTeX 環境が扱われます．

11.7.4 Previewing LaTeX fragments

もしあなたが‘dvipng’をインストールしているのであれば，LaTeX のコード片は出力された組版において画像として処理されます:

C-c C-x C-l
: ポイント位置の LaTeX コード片の画像プレビューの提供とソースコード上のオーバレイ．もしポイント位置にコード片がないのであれば，現在のエントリ (2 つの見出しの間) の全てのコード片を処理します．前置引数を付けて呼ばれた時は，サブツリー全体を処理します．前置引数を 2 つ付けて呼ばれた時，またはカーソルが最初の見出しの前にある時は，バッファ全体を処理します．

C-c C-c
: オーバレイされたプレビュー画像を消去します．

プレビューの外観を変更するために，あなたは`org-format-latex-optins`変数をカスタマイズすることができます．とりわけ，`:scale`(そして HTML へのエクスポートでは`:html-scale`) プロパティは画像のプレビューサイズの調整に使われます．

11.7.5 CDLaTeX を数学の入力に使う

CDLaTeX モードは環境や数学テンプレートの挿入をスピードアップするために AUCTeX に似たメジャーモードである LaTeX モードと併用して通常使われるマイナーモードです．Org-mode では，あなたは CDLaTeX モードのいくつかの機能を使用できます．あなたは`http://www.astro.uva.nl/~dominik/Tools/cdlatex`から‘`cdlatex.el`’と‘`texmathp.el`’(最近 AUCTeX に追加されました) をインストールする必要があります．Org-mode 中では CDLaTeX モード自身は使わないでください，代わりに Org-mode に一部である，より軽量なバージョンの`org-cdlatex-mode`を使ってください．`M-x org-cdlatex-mode`をカレントバッファで実行して有効にするか，全ての Org-mode ファイルで有効するために次の設定を行います:

```
(add-hook 'org-mode-hook 'turn-on-org-cdlatex)
```

このモードが有効である時，以下の機能が提供されます (詳細は CDLaTeX モードのドキュメントを参照してください)::

- *C-c {*による環境テンプレートの挿入．
- カーソルが LaTeX のコード片[10] の中にある場合，`TAB`キーはテンプレートの展開を行います．例えば，`TAB`は`fr`を`\\frac {}{}`に展開しカーソルを最初の括弧に移動します．もう一度`TAB`を押すと 2 つ目の括弧にカーソルが移動します．コード片の外だと，`TAB`は行の先頭にある環境の略語を展開します．例えば，もしあなたが行頭に‘`equ`’と書いていて`TAB`を押すと，この略語は`equation`環境に展開されます．全ての略語を見るには，*M-x cdlatex-command-help*をタイプしてください．
- LaTeX コード片の中で`_`と`^`を押すと，それらの文字が括弧のペアと一緒に挿入されます．もしあなたが`TAB`を括弧から抜け出すために使うなら，また括弧が 1 文字の文字かマクロのみを囲っているなら，それらは再び消去されます (`cdlatex-simplify-sub-super-script`変数に依存します)．
- Pressing the backquote ‘ followed by a character inserts math macros, also outside LaTeX fragments. If you wait more than 1.5 seconds after the backquote, a help window will pop up.

[10] カーソルがコード片の中にあるときに Org-mode はテストを行うためのメソッドを持ちます．詳細は`org-inside-LaTeX-fragment-p`関数のドキュメントを参照してください．

- 別の文字に続いてシングルクォート ’を押すと，強調やフォントでポイント前のシンボルが変更されます．もしシングルクォートを入力した後 1.5 秒以上待つと，ヘルプウィンドウがポップアップします．文字の変更は LaTeX コード片の中でのみ働きます; それ以外ではクォートは通常通りの働きをします．

12 Exporting

org-mode のドキュメントは様々なフォーマットにエクスポートすることができます。ノートを共有し印刷するには ASCII 形式でエクスポートすることで Org ファイルの読みやすく、シンプルなものが得られます。HTML のエクスポートではノートをウェブに公開できるようになりますし, XOXO フォーマットは他の様々なアプリケーションでやりとりするうえで確かな基礎となります。 LaTeX のエクスポートでは、org-mode とその構造化された編集機能を使って、容易に LaTeX のファイルを出力することができます。DocBook のエクスポートでは、Org ファイルを DocBook のツールを使った様々なフォーマットに変換することが可能です。プロジェクトの管理では、TaskJuggler 形式のエクスポートを使って、ガントリソースチャートを作成することができます。デッドラインや予約のような時間と関連のあるエントリーを iCal のようなデスクトップカレンダーに取り込むために org-mode は iCalendar 形式で抽出することもできます。現在、Org-mode はエクスポートのみをサポートしており、他の異なるフォーマットからインポートすることはできません。

org-mode は、 `transient-mark-mode` がオンの時 (Emacs 23 ではデフォルト)、は選択したリージョンをエクスポートをすることができます。

12.1 Selective export

エクスポートしたいドキュメントのある部分を選択、または除外する時にタグを使うことができます。その挙動は、`org-export-select-tags` と `org-export-exclude-tags` の二つの変数により決まります。

org-mode はまず最初に *select* タグがバッファにないかチェックします。 あった場合は, タグがない全てのツリーは除外されます。もし選択したツリーがサブツリーだった場合、それより上の階層はエクスポートされるものとして選択されますが, それより下の階層は選択されません。

もし、選択されたタグがなかった場合, バッファにある全ての内容がエクスポートされるものとして選択されるでしょう。

最後に, *exclude* タグでマークされていない全てのサブツリーはエクスポートバッファから除かれるでしょう。

12.2 Export options

エクスポートする際にはバッファにある特別な行が読みこまれます。 その行には追加的な情報が含まれており、ファイルの中でどこにでも書くことができます。*`C-c C-e t`* と入力することで、バッファにそのような行をセットで挿入することができます。それぞれの行で '`#+`' と入力した後に *`M-TAB`* による補完を行ない、 (see Section 15.1 [Completion], page 174) キーワードが正しいか、確認してみると良いでしょう。エクスポートと関連のない、バッファ内の設定の概要については Section 15.6 [In-buffer settings], page 176 を参照してください。特に、 `#+SETUPFILE` を使うことによって含めることができる別のファイルの中でよく使われる (エクスポートの) オプションを指定できることに注意してください

`C-c C-e t` `org-insert-export-options-template`

エクスポートオプションのテンプレートを挿入します。下の例を見てください。

`#+TITLE:` 表示されるタイトル (デフォルトはバッファ名)
`#+AUTHOR:` 著者 (デフォルトは `user-full-name` の値)
`#+DATE:` `format-time-string` で解釈される固定された日付の文字列
`#+EMAIL:` 彼/彼女のメールアドレス (デフォルトは `user-mail-address` の値)

`#+DESCRIPTION:` ページの説明, e.g. XHTML のメタタグで使われる。
`#+KEYWORDS:` ページのキーワード, e.g. XHTML のメタタグで使われる。
`#+LANGUAGE:` HTML で指定される言語 e.g. 'en' (org-export-default-language)
`#+TEXT:` 冒頭に挿入される説明的な文章
`#+TEXT:` 複数の行に書くことができます。
`#+OPTIONS:` H:2 num:t toc:t \n:nil @:t ::t |:t ^:t f:t TeX:t ...
`#+BIND:` lisp-var lisp-val, e.g.: org-export-latex-low-levels itemize

これらを確認するか, org-export-allow-BIND を設定すること
`#+LINK_UP:` 出力したページにおける ``up'' のリンク先
`#+LINK_HOME:` 出力したページにおける ``home'' のリンク先
`#+LATEX_HEADER:` LaTeX のヘッダーで使われる \usepackage{xyz} のような余分な行
`#+EXPORT_SELECT_TAGS:` エクスポートするツリーを示すタグ
`#+EXPORT_EXCLUDE_TAGS:` エクスポートから除外するツリーを示すタグ
`#+XSLT:` FO ファイルを生成するのに DocBook のエクスポート機能が使う XSLT のスタイルシート

OPTIONS 行は 以下のようなエクスポートの設定を示すコンパクトな式です。[1]

`H:` エクスポートする見出しの階層数
`num:` セクション番号の有無
`toc:` 目次の有無, または階層数の上限 (整数)
`\n:` 改行を維持するかどうか (うまく動作しない)
`@:` HTML の引用タグの有無
`::` 固定幅の段落の有無
`|:` 表の有無
`^:` 上付き、下付き文字を示す TEX のようなシンタックスの有無
"^:{}" は a_{b} 解釈されるが、
簡潔な a_b はそのままとなるでしょう。
`-:` 特別な文字列を変換するかどうか
`f:` this[1] のような脚注を用いるかどうか
`todo:` TODO キーワードを出力した文字列に含めるかどうか
`pri:` クッキーを優先するかどうか
`tags:` タグの有無, not-in-toc となるかもしれません。
`<:` DEADLINES のような時間/日付の有無
`*:` 強調テキストの有無 (太字, イタリック, アンダーライン)
`TeX:` テキスト中のシンプルな TEX マクロの有無
`LaTeX:` LATEX 出力の設定 デフォルトは auto
`skip:` 最初見出しの前にある文章をスキップするかどうか
`author:` 著者の名前/e-mail を出力するかどうか
`email:` 著者の e-mail を出力するかどうか
`creator:` 作者を出力するかどうか
`timestamp:` 作成した日付を出力するかどうか
`d:` drawer を出力するかどうか

これらのオプションは HTML、LATEX の両方のエクスポートに影響します。TeX と LaTeX のオプションを除き LATEX のエクスポートをするのに、それぞれ t 、または nil となります。

[1] もし、このように多くのオプションを設定したい時は, それぞれオプション行を作りことができます。

`org-export-html-pre/postamble` を `t` とすると HTML にエクスポートする時に `author` 、 `email` 及び `creator` の値は上書きされるでしょう。 代わりに `org-export-html-pre/postamble-format` が用いられます。

このようなオプションの初期値は変数のセットで与えられます。 そのような変数は、OPTIONS のキーと公開するキーにも対応しています。(see Section 13.1.1 [Project alist], page 144), `org-export-plist-vars` の定数を見てください。

エクスポートのコマンドを呼びだす前に、 `@C-c` で選択した単一のサブツリーをエクスポートする時、そのサブツリーは、`EXPORT_FILE_NAME` 、 `EXPORT_TITLE` 、 `EXPORT_TEXT` 、`EXPORT_AUTHOR` 、 `EXPORT_DATE` 、 そして `EXPORT_OPTIONS` プロパティでエクスポートの設定を無視することができます。

12.3 The export dispatcher

全てのエクスポートコマンドはエクスポートコマンド選択画面から選ぶことができます。コマンド選択画面では、コマンドを特定するための追加的なキーの入力を促されます。通常、ファイルの全ての内容がエクスポートされますが、もしアクティブなリージョンに一つのアウトラインツリーが含まれていた場合, まず、見出しがドキュメントのタイトルとして扱われ、サブツリーがエクスポートされます。

`C-c C-e` `org-export`
: エクスポート、または公開のコマンド選択画面です。エクスポート、または公開のコマンドを起動するのに必要なキーがヘルプウィンドウに表示されます。前置引数として、入力すると、直接エクスポート機能となります。二重の前置引数 `C-u C-u` を入力することで、コマンドは別の Emacs プロセスにおいてバックグラウンドで実行されます。[2].

`C-c C-e v` `org-export-visible`
: `C-c C-e` のように動作しますが、 今見えている文章だけがエクスポートされます。(i.e. アウトライン表示により、隠されていない文章).

`C-u C-u C-c C-e` `org-export`
: エクスポート機能が呼ばれますが、 `org-export-run-in-background` の設定と逆の挙動となります。 i.e. 動いていないバックグラウンドプロセスを呼びだしたり, 現在の Emacs のプロセスで強制的に実行したりします。

12.4 ASCII/Latin-1/UTF-8 export

ASCII 形式へのエクスポートは、org-mode のファイルを ASCII のみが含まれる、シンプルで読みやすい形に書き出します。Latin-1 及び UTF-8 でのエクスポートでは特殊な文字やシンボルをそれらのエンコードで出力します。

`C-c C-e a` `org-export-as-ascii`
: ASCII 形式のファイルをエクスポートします。 Org ファイルを '`myfile.org`' だとすると、ASCII 形式のファイルは '`myfile.txt`' となるでしょう。 そのファイルは警告なしに上書きされます。もしアクティブなリージョン[3] があった場合, そのリージョンのみがエクスポートされます。 選択したリージョンが一つのツリー @[4] を含んでいた場合、

[2] このような挙動をデフォルトにするには、 `org-export-run-in-background`変数を設定してください.

[3] `transient-mark-mode` が有効である必要があります。

[4] 現在のサブツリーの選択するには、 `C-c` と入力してください。

そのツリーの見出しがドキュメントのタイトルとなるでしょう．見出しがあるか、または EXPORT_FILE_NAME プロパティを継承していた場合、 エクスポートする際にはその名前が使われるでしょう。

C-c C-e A org-export-as-ascii-to-buffer
一時的なバッファに出力し、ファイルを作成しません。

C-c C-e n org-export-as-latin1
C-c C-e N org-export-as-latin1-to-buffer
上に示したコマンドのような動作をしますが, Latin-1 でエンコーディングされたものが出力されます。

C-c C-e u org-export-as-utf8
C-c C-e U org-export-as-utf8-to-buffer
上に示したコマンドのような動作をしますが, UTF-8 でエンコーディングされたものが出力されます。

C-c C-e v a/n/u
文書の中で、バッファで表示されている部分だけを出力する。

エクスポートされたものでは、 最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層はアイテムのリストとしてエクスポートされます。 この違いを異なる階層に変えたい場合は、 前置引数で、その階層を指定します。 例えば、

```
C-1 C-c C-e a
```

は 最初の階層のみを見出しとし、それ以外はアイテムとなります。見出しがアイテムに変更された時、見出し後の文章のインデントは、アイテムの下にうまく調和するように変更されます。この変更は、最初の本文が全体のインデントを示しているという仮定のもとで実行されます。これよりも大きなインデントは、最初の文章との相対的なレイアウトを維持するように調整されます。最初の行より少ないインデントであれば, 左寄せします。

次の見出しの前にあるリンクは脚注のような形でエクスポートされます。その脚注は、次の見出しの前に項目名とリンクがエクスポートされます。詳しい内容と他のオプションについては、変数 org-export-ascii-links-to-notes を見てください。

12.5 HTML export

org-mode には多くの HTML のフォーマットに対応した HTML (XHTML 1.0 準拠) エクスポート機能があります。それは、John Gruber が開発した *markdown* 言語に似ていますが、org-mode ではさらにテーブルもサポートしています。

12.5.1 HTML エクスポートのコマンド

C-c C-e h org-export-as-html
HTML ファイル ‘myfile.html’をエクスポートします。Org ファイル ‘myfile.org’ をエクスポートすると、ASCII 形式のファイルは ‘myfile.html’となるでしょう。そのファイルは警告なしに上書きされます。もしアクティブなリージョン [5] があった場合, そのリージョンのみがエクスポートされます。 選択したリージョンが一つのツリー[6] を含んでいた場合、そのツリーの見出しがドキュメントのタイトルとなるでしょう．見出

[5] transient-mark-mode が有効である必要があります。
[6] 現在のサブツリーの選択するには、 *C-c @* と入力してください。

しがあるか、または EXPORT_FILE_NAME プロパティを継承していた場合、 エクスポートする際にはその名前が使われるでしょう。

C-c C-e b org-export-as-html-and-open
HTML ファイルをエクスポートし、そのファイルをブラウザで開きます。

C-c C-e H org-export-as-html-to-buffer
一時的なバッファに出力し、ファイルを作成しません。

C-c C-e R org-export-region-as-html
アクティブなリージョンを一時的なバッファに出力します。前置引数があるとヘッダーとフッターを出力せずに、リージョンの HTML のみを出力します。これはカットアンドペーストで編集する際に便利です

C-c C-e v h/b/H/R
文書の中で、バッファで表示されている部分だけを出力する。

M-x org-export-region-as-html
org-mode の記法が使われているという前提でリージョンを HTML に変換します。これはどのバッファでも起動するグローバルなコマンドです。

M-x org-replace-region-by-HTML
org-mode の記法が使われているという前提でアクティブなリージョンを HTML に変換します。

エクスポートされたものでは、 最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層はアイテムのリストとしてエクスポートされます。 この違いを異なる階層に変えたい場合は、 前置引数で、その階層を指定します。 例えば、

C-2 C-c C-e b

この場合 2 番目のレベルまでを見出しとして取り扱い、それ以外は項目として取り扱います。

12.5.2 Quoting HTML tags

HTML にエクスポートする際、プレインな '<' and '>'は常に '<'と '>'に変換されます。もし単純な HTML タグをそのまま含めたい時は, @@'bold text'のように ma@''でマークします。これは単純な HTML タグでしか動作しませんので注意してください。エクスポートするファイルにさらに広範囲な HTML をそのままコピーするには次のようなブロックが使えます。

#+HTML: エクスポートする HTML コード

or

マーカー間の全ての行は文字どおり出力されます。

12.5.3 Links in HTML export

内部リンク (see Section 4.2 [Internal links], page 33) エクスポートされ HTML でも同様に動作します。これには、ラジオターゲット (see Section 4.2.1 [Radio targets], page 34) により生成された自動リンクも含まれます。もしターゲットとなるファイルが公開される Org ファイルを示す同じ相対パス上にあっても、リンクは外部リンクとして動作するでしょう。他の '.org'ファイルへのリンクは、HTML にエクスポートされたものにも同じ相対パスでリンクされたファイルがある、という前提で、リンクに変換されます。'id:'リンクはファイル間で特定のエントリーにジャンプするのに使われます。リンクするファイル、公開ディレクトリでの公開に関する情報については、Section 13.1.6 [Publishing links], page 147 参照してください。

リンクの属性を記述したい時は、特別な#+ATTR_HTML行を用いることができます。この行は、<a>タグや<img>タグを追加する属性を定義するために使われます。以下の例では、リンクに titleと styleの属性を設定しています。

```
#+ATTR_HTML: title="The Org-mode homepage" style="color:red;"
[[http://orgmode.org]]
```

12.5.4 Tables

org-modeの表は、org-export-html-table-tagで定義されているテーブルのタグを使ってHTMLにエクスポートされます。デフォルトの設定では、セルの罫線とフレームがない状態でテーブルが出力されます。 個々のテーブルでその設定を変えたい場合は, 次のような行をテーブルの前に記述してください。

```
#+CAPTION: これはセルの周囲に線が引かれた表です。
#+ATTR_HTML: border="2" rules="all" frame="all"
```

12.5.5 Images in HTML export

HTML のエクスポートでは Org ファイルにリンクがある画像をインライン表示することができます。その画像はリンクされているクリック可能な部分として扱われます。デフォルトでは、[7], リンクに description がなければ、画像はインライン表示されます。 つまり、'[[file:myimg.jpg]]'はインライン表示されますが、'[[file:myimg.jpg][the image]]'はが画像にリンクされる 'the image' というテキストリンクが作られます。description の部分が file:リンクか画像を示す http:の URL の場合, 画像はインラインに表示され、画像がクリックされると活性化されます。例えば、リンク先に高解像度の画像があるサムネイルを追加したい場合、次のように書くと良いでしょう。

```
[[file:highres.jpg][file:thumb.jpg]]
```

インライン画像に属性を追加したい場合は、#+ATTR_HTMLを使います。次の例では、テキストでの見やすさとアクセスのしやすさを考慮して altと titleの属性を指定して、align を右にしています。

```
#+CAPTION: A black cat stalking a spider
#+ATTR_HTML: alt="cat/spider image" title="Action!" align="right"
[[./img/a.jpg]]
```

httpのアドレスも使うことができます。

12.5.6 Math formatting in HTML export

LaTeX の数学用スニペット (see Section 11.7.3 [LaTeX fragments], page 121) は二つの異なる方法で HTML に表示される。デフォルトでは org-mode をインストールすると、すぐに MathJax system (http://www.mathjax.org) が使えるようになっています。http://orgmode.orgは 'MathJax' が Org-mode ユーザ、小さなアプリケーション、そしてテストにとって便利だと考えているからです。

もし特定のページで、あるいは常に **'MathJax'**を使うのであれば、私達のサーバでの読みこみを減らすために **MathJax** をあなたのサーバにインストール[8] してください。'MathJax'について設定するには、org-export-html-mathjax-optionsを使うか、バッファに次のような行を挿入してください。

---

[7] ただし、org-export-html-inline-imagesを確認してください。

[8] インストール方法については、MathJax のウェブサイトにあります。http://www.mathjax.org/resources/docs/?instal
を参照してください。

```
#+MATHJAX: align:"left" mathml:t path:"/MathJax/MathJax.js"
```

See the docstring of the variable この行の各パラメータの意味の知るための `org-export-html-mathjax-options`

望むのであれば、LaTeX を小さな画像に変換してブラウザ上のページに挿入することもできます。MathJax が有用である前には、これが org-mode でのデフォルトの方法でした。この方法を用いるには、あなたのシステムで‘`dvipng`’プログラムが利用できる状態である必要があります。この方法は以下のような行を追加することでも有効になります。

```
#+OPTIONS: LaTeX:dvipng
```

12.5.7 Text areas in HTML export

コードサンプルを HTML にして公開する方法として、テキストエリアを使う方法があります。何かのアプリケーションに貼りつける前であれば、そのコードサンプルは編集することができます。`example` ブロックか `src`ブロックに`-t`スイッチが付加されることでテキストエリアに変換されます。このスイッチを使うことで、シンタックス、ラベルのハイライト、行番号に関するオプションが無効になります。`-h`と`-w`を使うことがあるかもしれません。それらのスイッチはテキストエリアの高さと幅を特定するもので、デフォルトでは高さが example ブロックの行数で幅は 80 となります。設定は、例えば以下のようになります。

```
#+BEGIN_EXAMPLE -t -w 40
  (defun org-xor (a b)
     "Exclusive or."
     (if a (not b) b))
#+END_EXAMPLE
```

12.5.8 CSS support

エクスポートするファイルには、スタイルに関する情報を含めることができます。HTML エクスポート機能には、文章のパーツを適切に表示するために次に示す特別な CSS クラス

[9] があります。見出しやテーブルなどの標準的なクラスに加えて、それら特別な CSS クラスも変更することができます。

`p.author`	著者の情報、email 含む
`p.date`	公開日
`p.creator`	作成情報, org-mode のバージョン
`.title`	文章のタイトル
`.todo`	DONE となっていない TODO キーワード
`.done`	DONE キーワード、DONE と扱われる全てのキーワードが対象
`.WAITING`	
各 TODO キーワードはその名前のクラス名も用いることができる	
`.timestamp`	タイムスタンプ
`.timestamp-kwd`	SCHEDULED 等のタイムスタンプに関連するキーワード
`.timestamp-wrapper`	SCHEDULED 等のキーワードとタイムスタンプ全体
`.tag`	見出し中のタグ
`._HOME`	各タグはその名前のクラス名も用いることができる (`"`は`"_"`に置き換えられる)

[9] TODO キーワードやタグに CSS が適用されるとコンフリクトを起こします。`org-export-html-todo-kwd-class-prefix`と `org-export-html-tag-class-prefix`を使って、それらをユニークにしてください。

.target	リンクのターゲット
.linenr	コード中の行番号
.code-highlighted	参照されコード行のハイライト
div.outline-N	深さレベル N の div 要素 (見出しとテキスト)
div.outline-text-N	深さレベル N のテキスト部分の div 要素
.section-number-N	深さレベル N の見出しの番号。各レベルで異なる
div.figure	インライン画像のフォーマット方法
pre.src	ソースコードブロックのフォーマット方法
pre.example	例示ブロック
p.verse	verse ブロック
div.footnotes	脚注の見出し
p.footnote	脚注定義の文章、脚注を含む
.footref	脚注の参照番号 (常に<sup>となる)
.footnum	脚注定義中の番号 (常に<sup>となる)

エクスポートされたファイルは、基礎的な方法で定義されたコンパクトなスタイル[10] が含まれています。この設定は上書きされるかもしれませんし、`org-export-html-style` (Org-wide の設定に使われます) や `org-export-html-style-extra` (ファイルごとの設定のような詳細な設定に使われます。) を使って追加されるかもしれません。後者の変数をファイルごとに設定するには、次のように行ないます。

```
#+STYLE: <link rel="stylesheet" type="text/css" href="stylesheet.css" />
```

長いスタイルの定義には複数行で記述することもできます。外部ファイルを参照せずに`<style>` `</style>`セクションに直接記述してください。

サブツリーにスタイルを追加するには, ツリーにクラスを適用する`:HTML_CONTAINER_CLASS:` プロパティを使います。個々の見出しに CSS スタイルを適用するには、`:CUSTOM_ID:`プロパティで指定される ID を使うことができます。

12.5.9 ウェブページの表示に関する JavaScript のサポート

Sebastian Rose は、org-mode が生成した HTML ファイルに関するウェブエクスペリエンスを拡張するためにデザインされた Javascript プログラムを書きました。このプログラムを使うことで、異なる二つの方法で大きなファイルを見ることができます。一つめは *Info* のようなモードで、それぞれの章は別々に表示され、*n*キーと *p*キーで操作できます。(他のキーでも操作できます。利用できるキーの概要を知るには、*?*を入力してください。)。二つめは、org-mode が Emacs で提供するような折りたたまれたスタイルです。このスクリプトは、`http://orgmode.org/org-info.js`で利用できます。ドキュメントについては、`http://orgmode.org/worg/code/org-info-js/`にあります。このスクリプトは私達のサイトでホスティングしていますが、何度も使う場合は、`orgmode.org`にあるものを使わずにあなたのサーバにコピーしたものを使う方を選択するかもしれません。

このスクリプトを使うには、'`org-jsinfo.el`'がロードされているか、確認する必要があります。デフォルトでは、ロードされるようになっていますが、*M-x customize-variable RET org-modules RET*と入力して、確かにロードされている確認してください。このプログラムを使えるようにするには、次のような行を Org ファイルに追加するだけです。

```
#+INFOJS_OPT: view:info toc:nil
```

ファイル中にこの行が見つかると、HTML のヘッダーは自動的にこのスクリプトを起動させるのに必要なコードを自動的に追加します。以上のような行を使うと、次のようなオプションを設定できます。

[10] このスタイルは `org-export-html-style-default`で定義されており、変更できません。この初期設定を無効にするには `org-export-html-style-include-default`を修正してください。

`path:` スクリプトのパス。デフォルトでは、`http://orgmode.org/org-info.js`

を使うようになっていますが、ローカルにコピーしたものを使いたい場合は
‘`../scripts/org-info.js`’のようなパスを使ってください。
`view:` ウェブサイトを最初に開いた時の表示。可能な値は次のとおり:
`info`
一つのページに一つのセクションが表示される Info のようなインターフェイス
`overview`
最初はトップレベルのみが表示される折りたたみインターフェイス
`content` 全ての見出しが見える状態の折りたたみインターフェイス
`showall`
全ての見出しと文章が見える状態の折りたたみインターフェイス
`sdepth:` info や折りたたみモードで独立して表示されるセクションの
最大の見出しレベル。デフォルトでは `org-export-headline-levels`
(= `#+OPTIONS`の中の `H`スイッチ) の値が使われる。
もし、`org-export-headline-levels`の値より小さかった場合、
info/折りたたみ のセクションは小見出しまで含まれます。
`toc:` 目次表示の有無
`nil`としても、`i`を入力することで目次は表示されます。
`tdepth:` 目次の深さ。デフォルトでは、`org-export-headline-levels`
`org-export-with-toc`の値が用いられます。
`ftoc:` CSS によって、目次の場所を指定するかどうか。
「yes」の場合は、セクションとして表示されなくなります。
`ltoc:` それぞれのセクションにショートコンテンツを設置するかどうか。

セクションの冒頭にショートコンテンツを設置する場合は値を `above`とします。
`mouse:` マウスを見出しの上に移動させた時にハイライトさせます。

‘`underline`’ (default) か、‘`#cccccc`’のように背景色が指定できます。
`buttons:`
ビューの変更をトグルさせるボタンを様々なところに設置するかどうか。
`nil`の場合は、(デフォルト)、ボタンが一つだけ表示されます。

`org-infojs-options`を変更することで、これらのオプションの初期値を変更することができます。このスクリプトを常にページに適用させたい場合は、`org-export-html-use-infojs`を変更してください。

12.6 LaTeX と PDF のエクスポート

org-mode には、Bastien Guerry によって書かれた LaTeX のエクスポート機能があります。追加的な処理と合わせて、[11], このバックエンドは PDF の出力にも使われています。LaTeX の出力は、リンクと相互参照の実装に ‘`hyperref`’を使っているので、出力された PDF ファイルは完全にリンクされているでしょう。セクションの階層に合わせて正しく出力されるためには、org ファイルは適切に構造化されていないといけないので注意してください。

[11] デフォルトの LaTeX 出力は、pdftex または latex により出力されるよう設計されています。それには、xetex や恐らく luatex と互換性のないパッケージが含まれています。`org-export-latex-default-packages-alist`や `org-export-latex-packages-alist`を参照してください。

12.6.1 LaTeX エクスポートのコマンド

C-c C-e l org-export-as-latex
: LaTeX ファイル 'myfile.tex'を出力します。Org ファイルに対して 'myfile.org',、ASCII ファイルは 'myfile.tex'となるでしょう。そのファイルは警告なしに上書きされます。アクティブなリージョン[12] があれば、そのリージョンのみが出力されるでしょう。選択したリージョンが一つのツリー@[13] であった場合、ツリーの見出しがタイトルになります。ツリーの見出しのエントリーがEXPORT_FILE_NAMEプロパティを継承、または持っている場合、エクスポートされる際には、その名前が使われるでしょう。

C-c C-e L org-export-as-latex-to-buffer
: 一時バッファに出力します。ファイルを作りません。

C-c C-e v l/L
: 文書の中で、バッファで表示されている部分だけを出力する。

M-x org-export-region-as-latex
: Org-mode の記法が使われているという前提でリージョンを LaTeX に変換します。これはどのバッファでも起動するグローバルなコマンドです。

M-x org-replace-region-by-latex
: アクティブなリージョンを (Org-mode の記法が使われている前提で)LaTeX コードに置き変えます。

C-c C-e p org-export-as-pdf
: LaTeX に出力し、PDF にも変換します。

C-c C-e d org-export-as-pdf-and-open
: LaTeX に出力し、PDF にも変換します。その際出力された PDF ファイルを開きます。

エクスポートされたものでは、 最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層は概要のリストとしてエクスポートされます。エクスポート機能では、org-latex-low-levelsを変更することで、この設定を無視、または変更することができます。

この違いを異なる階層で変えたい場合は、 前置引数で、その階層を指定します。 例えば、

C-2 C-c C-e l

この場合 2 番目のレベルまでを見出しとして取り扱い、それ以外は項目として取り扱います。

12.6.2 見出しと構造の分割

デフォルトでは、LaTeX の出力にはarticleクラスが使われます。

クラスはorg-export-latex-default-classの値を変更することで、全体的に変更することもできますし、ファイル中に org-export-latex-default-classのようなオプションを追加することで、局所的に変更することもできます。:LaTeX_CLASS:プロパティを使えば、エクスポートするリージョンにそのツリー (サブツリー) のみが含まれていた場合にクラスを指定できます。クラスは、org-export-latex-classesにリストアップされてます。この変数は、各クラス[14] の見出しテンプレートを定義し、各クラスの構造の分割について定義します。クラス自体についても定義されま

[12] transient-mark-modeが有効である必要があります。
[13] 現在のサブツリーを選択するには、*C-c* を入力してください。
[14] org-export-latex-default-packages-alistと org-export-latex-packages-alistが接合されたものです。

す。#+LaTeX_CLASS_OPTIONS、またはLaTeX_CLASS_OPTIONSプロパティは\documentclassマクロのオプションを指定します。見出しに#+LATEX_HEADER: \usepackage{xyz}を追加して同様のことをすることもできます。詳しい情報については、org-export-latex-classesのドキュメント文字列を参照してください。

12.6.3 LaTeX コードの引用

Section 11.7 [Embedded LaTeX], page 120 で記述された埋め込まれた LaTeX は、LaTeX に正しく挿入されます。図の相互参照を生成するために、'\ref{LABEL}'のようなシンプルなマクロが含まれます。さらに、次のような行を追加することで、LaTeX エクスポートの際に表示だけしてほしい特別なコードを追加することができます。

```
#+LaTeX: エクスポートする際に文字のまま、出力される LaTeX code
```

or

```
#+BEGIN_LaTeX
マーカの間にある全ての行は文字がそのまま出力されます。
#+END_LaTeX
```

12.6.4 LaTeX エクスポートにおける表

LaTeX で表を出力する際に、ラベルと表題を指定することができます (see Section 11.2 [Images and tables], page 117)、ATTR_LaTeX行を使うことで、表に関する longtable環境を呼び出すこともできます。複数のページにまたがる表や、デフォルトの表の環境を tableから table*にするため、またはデフォルトの内部 tabular 環境を tabularxや tabularyにしたい時にも ATTR_LaTeX行は使われます。つまり、文字の配置や (tabularxや tabularyを使って) 幅を次のようにして設定できます。:

```
#+CAPTION: A long table
#+LABEL: tbl:long
#+ATTR_LaTeX: longtable align=l|lp{3cm}r|l
| ..... | ..... |
| ..... | ..... |
```

tabularyを使って、複数のセルにまたがる表を指定することもできます。

```
#+CAPTION: A wide table with tabulary
#+LABEL: tbl:wide
#+ATTR_LaTeX: table* tabulary width=\textwidth
| ..... | ..... |
| ..... | ..... |
```

12.6.5 LaTeX エクスポートにおける画像

Images that are linked to without a description part in the link, like '[[file:img.jpg]]' or '[[./img.jpg]]' will be inserted into the PDF output file resulting from LaTeX processing. Org will use an \includegraphics macro to insert the image. If you have specified a caption and/or a label as described in Section 11.2 [Images and tables], page 117, the figure will be wrapped into a figure environment and thus become a floating element. You can use an #+ATTR_LaTeX: line to specify the various options that can be used in the optional argument of the \includegraphics macro. To modify the placement option of the figure environment, add something like 'placement=[h!]' to the Attributes.

If you would like to let text flow around the image, add the word 'wrap' to the #+ATTR_LaTeX: line, which will make the figure occupy the left half of the page. To fine-tune,

the `placement` field will be the set of additional arguments needed by the `wrapfigure` environment. Note that if you change the size of the image, you need to use compatible settings for `\includegraphics` and `wrapfigure`.

```
#+CAPTION:    The black-body emission of the disk around HR 4049
#+LABEL:      fig:SED-HR4049
#+ATTR_LaTeX: width=5cm,angle=90
[[./img/sed-hr4049.pdf]]

#+ATTR_LaTeX: width=0.38\textwidth wrap placement={r}{0.4\textwidth}
[[./img/hst.png]]
```

If you need references to a label created in this way, write '`\ref{fig:SED-HR4049}`' just like in LaTeX.

12.6.6 Beamer クラスのエクスポート

The LaTeX class '`beamer`' allows production of high quality presentations using LaTeX and pdf processing. Org-mode has special support for turning an Org-mode file or tree into a '`beamer`' presentation.

When the LaTeX class for the current buffer (as set with `#+LaTeX_CLASS: beamer`) or subtree (set with a `LaTeX_CLASS` property) is `beamer`, a special export mode will turn the file or tree into a beamer presentation. Any tree with not-too-deep level nesting should in principle be exportable as a beamer presentation. By default, the top-level entries (or the first level below the selected subtree heading) will be turned into frames, and the outline structure below this level will become itemize lists. You can also configure the variable `org-beamer-frame-level` to a different level—then the hierarchy above frames will produce the sectioning structure of the presentation.

A template for useful in-buffer settings or properties can be inserted into the buffer with *`M-x org-insert-beamer-options-template`*. Among other things, this will install a column view format which is very handy for editing special properties used by beamer.

You can influence the structure of the presentation using the following properties:

`BEAMER_env`
: The environment that should be used to format this entry. Valid environments are defined in the constant `org-beamer-environments-default`, and you can define more in `org-beamer-environments-extra`. If this property is set, the entry will also get a `:B_environment:` tag to make this visible. This tag has no semantic meaning, it is only a visual aid.

`BEAMER_envargs`
: The beamer-special arguments that should be used for the environment, like `[t]` or `[<+->]` of `<2-3>`. If the `BEAMER_col` property is also set, something like `C[t]` can be added here as well to set an options argument for the implied `columns` environment. `c[t]` or `c<2->` will set an options for the implied `column` environment.

`BEAMER_col`
: The width of a column that should start with this entry. If this property is set, the entry will also get a `:BMCOL:` property to make this visible. Also this tag

is only a visual aid. When this is a plain number, it will be interpreted as a fraction of `\textwidth`. Otherwise it will be assumed that you have specified the units, like '`3cm`'. The first such property in a frame will start a `columns` environment to surround the columns. This environment is closed when an entry has a `BEAMER_col` property with value 0 or 1, or automatically at the end of the frame.

`BEAMER_extra`
: Additional commands that should be inserted after the environment has been opened. For example, when creating a frame, this can be used to specify transitions.

Frames will automatically receive a `fragile` option if they contain source code that uses the verbatim environment. Special '`beamer`' specific code can be inserted using `#+BEAMER:` and `#+BEGIN_beamer...#+end_beamer` constructs, similar to other export backends, but with the difference that `#+LaTeX:` stuff will be included in the presentation as well.

Outline nodes with `BEAMER_env` property value '`note`' or '`noteNH`' will be formatted as beamer notes, i,e, they will be wrapped into `\note{...}`. The former will include the heading as part of the note text, the latter will ignore the heading of that node. To simplify note generation, it is actually enough to mark the note with a *tag* (either `:B_note:` or `:B_noteNH:`) instead of creating the `BEAMER_env` property.

You can turn on a special minor mode `org-beamer-mode` for editing support with

```
#+STARTUP: beamer
```

C-c C-b `org-beamer-select-environment`
: In `org-beamer-mode`, this key offers fast selection of a beamer environment or the `BEAMER_col` property.

Column view provides a great way to set the environment of a node and other important parameters. Make sure you are using a COLUMN format that is geared toward this special purpose. The command *M-x org-insert-beamer-options-template* defines such a format.

Here is a simple example Org document that is intended for beamer export.

```
#+LaTeX_CLASS: beamer
#+TITLE: Example Presentation
#+AUTHOR: Carsten Dominik
#+LaTeX_CLASS_OPTIONS: [presentation]
#+BEAMER_FRAME_LEVEL: 2
#+BEAMER_HEADER_EXTRA: \usetheme{Madrid}\usecolortheme{default}
#+COLUMNS: %35ITEM %10BEAMER_env(Env) %10BEAMER_envargs(Args) %4BEAMER_col(Col) %8BEAMER_extra(Ex)

* This is the first structural section

** Frame 1 \\ with a subtitle
*** Thanks to Eric Fraga                                          :BMCOL:B_block:
    :PROPERTIES:
    :BEAMER_env: block
    :BEAMER_envargs: C[t]
    :BEAMER_col: 0.5
    :END:
    for the first viable beamer setup in Org
*** Thanks to everyone else                                       :BMCOL:B_block:
```

```
    :PROPERTIES:
    :BEAMER_col: 0.5
    :BEAMER_env: block
    :BEAMER_envargs: <2->
    :END:
    for contributing to the discussion
**** This will be formatted as a beamer note                    :B_note:
** Frame 2 \\ where we will not use columns
*** Request                                                     :B_block:
    Please test this stuff!
    :PROPERTIES:
    :BEAMER_env: block
    :END:
```

For more information, see the documentation on Worg.

12.7 DocBook export

Org contains a DocBook exporter written by Baoqiu Cui. Once an Org file is exported to DocBook format, it can be further processed to produce other formats, including PDF, HTML, man pages, etc., using many available DocBook tools and stylesheets.

Currently DocBook exporter only supports DocBook V5.0.

12.7.1 DocBook export commands

C-c C-e D `org-export-as-docbook`

Export as DocBook file. For an Org file, '`myfile.org`', the DocBook XML file will be '`myfile.xml`'. The file will be overwritten without warning. If there is an active region[15], only the region will be exported. If the selected region is a single tree@[16], the tree head will become the document title. If the tree head entry has, or inherits, an `EXPORT_FILE_NAME` property, that name will be used for the export.

C-c C-e V `org-export-as-docbook-pdf-and-open`

Export as DocBook file, process to PDF, then open the resulting PDF file.

Note that, in order to produce PDF output based on exported DocBook file, you need to have XSLT processor and XSL-FO processor software installed on your system. Check variables `org-export-docbook-xslt-proc-command` and `org-export-docbook-xsl-fo-proc-command`.

The stylesheet argument `%s` in variable `org-export-docbook-xslt-proc-command` is replaced by the value of variable `org-export-docbook-xslt-stylesheet`, which needs to be set by the user. You can also overrule this global setting on a per-file basis by adding an in-buffer setting `#+XSLT:` to the Org file.

C-c C-e v D

Export only the visible part of the document.

[15] This requires `transient-mark-mode` to be turned on

[16] To select the current subtree, use *C-c .*

12.7.2 Quoting DocBook code

You can quote DocBook code in Org files and copy it verbatim into exported DocBook file with the following constructs:

```
#+DOCBOOK: Literal DocBook code for export
```

or

```
#+BEGIN_DOCBOOK
All lines between these markers are exported by DocBook exporter
literally.
#+END_DOCBOOK
```

For example, you can use the following lines to include a DocBook warning admonition. As to what this warning says, you should pay attention to the document context when quoting DocBook code in Org files. You may make exported DocBook XML files invalid by not quoting DocBook code correctly.

```
#+BEGIN_DOCBOOK
<warning>
  <para>You should know what you are doing when quoting DocBook XML code
  in your Org file.  Invalid DocBook XML may be generated by
  DocBook exporter if you are not careful!</para>
</warning>
#+END_DOCBOOK
```

12.7.3 Recursive sections

DocBook exporter exports Org files as articles using the `article` element in DocBook. Recursive sections, i.e. `section` elements, are used in exported articles. Top level headlines in Org files are exported as top level sections, and lower level headlines are exported as nested sections. The entire structure of Org files will be exported completely, no matter how many nested levels of headlines there are.

Using recursive sections makes it easy to port and reuse exported DocBook code in other DocBook document types like `book` or `set`.

12.7.4 Tables in DocBook export

Tables in Org files are exported as HTML tables, which have been supported since DocBook V4.3.

If a table does not have a caption, an informal table is generated using the `informaltable` element; otherwise, a formal table will be generated using the `table` element.

12.7.5 Images in DocBook export

Images that are linked to without a description part in the link, like '`[[file:img.jpg]]`' or '`[[./img.jpg]]`', will be exported to DocBook using `mediaobject` elements. Each `mediaobject` element contains an `imageobject` that wraps an `imagedata` element. If you have specified a caption for an image as described in Section 11.2 [Images and tables], page 117, a `caption` element will be added in `mediaobject`. If a label is also specified, it will be exported as an `xml:id` attribute of the `mediaobject` element.

Image attributes supported by the `imagedata` element, like `align` or `width`, can be specified in two ways: you can either customize variable `org-export-docbook-default-image-attributes` or use the `#+ATTR_DOCBOOK:` line. Attributes specified in variable `org-export-docbook-default-image-attributes` are applied to all inline images in the Org file to be exported (unless they are overridden by image attributes specified in `#+ATTR_DOCBOOK:` lines).

The `#+ATTR_DOCBOOK:` line can be used to specify additional image attributes or override default image attributes for individual images. If the same attribute appears in both the `#+ATTR_DOCBOOK:` line and variable `org-export-docbook-default-image-attributes`, the former takes precedence. Here is an example about how image attributes can be set:

```
#+CAPTION:    The logo of Org-mode
#+LABEL:      unicorn-svg
#+ATTR_DOCBOOK: scalefit="1" width="100%" depth="100%"
[[./img/org-mode-unicorn.svg]]
```

By default, DocBook exporter recognizes the following image file types: '`jpeg`', '`jpg`', '`png`', '`gif`', and '`svg`'. You can customize variable `org-export-docbook-inline-image-extensions` to add more types to this list as long as DocBook supports them.

12.7.6 DocBook 出力における特殊文字

Special characters that are written in TEX-like syntax, such as `\alpha`, `\Gamma`, and `\Zeta`, are supported by DocBook exporter. These characters are rewritten to XML entities, like `&alpha;`, `&Gamma;`, and `&Zeta;`, based on the list saved in variable `org-entities`. As long as the generated DocBook file includes the corresponding entities, these special characters are recognized.

You can customize variable `org-export-docbook-doctype` to include the entities you need. For example, you can set variable `org-export-docbook-doctype` to the following value to recognize all special characters included in XHTML entities:

```
"<!DOCTYPE article [
<!ENTITY % xhtml1-symbol PUBLIC
\"-//W3C//ENTITIES Symbol for HTML//EN//XML\"
\"http://www.w3.org/2003/entities/2007/xhtml1-symbol.ent\"
>
%xhtml1-symbol;
]>
"
```

12.8 TaskJuggler export

TaskJuggler (`http://www.taskjuggler.org/`) is a project management tool. It provides an optimizing scheduler that computes your project time lines and resource assignments based on the project outline and the constraints that you have provided.

The TaskJuggler exporter is a bit different from other exporters, such as the HTML and LaTeX exporters for example, in that it does not export all the nodes of a document or strictly follow the order of the nodes in the document.

Instead the TaskJuggler exporter looks for a tree that defines the tasks and a optionally tree that defines the resources for this project. It then creates a TaskJuggler file based on these trees and the attributes defined in all the nodes.

12.8.1 TaskJuggler export commands

`C-c C-e j` `org-export-as-taskjuggler`
: Export as TaskJuggler file.

`C-c C-e J` `org-export-as-taskjuggler-and-open`
: Export as TaskJuggler file and then open the file with TaskJugglerUI.

12.8.2 Tasks

Create your tasks as you usually do with Org-mode. Assign efforts to each task using properties (it's easiest to do this in the column view). You should end up with something similar to the example by Peter Jones in `http://www.contextualdevelopment.com/static/artifacts/articles/2008/project-planning/project-planning.org`. Now mark the top node of your tasks with a tag named `:taskjuggler_project:` (or whatever you customized `org-export-taskjuggler-project-tag` to). You are now ready to export the project plan with `C-c C-e J` which will export the project plan and open a gantt chart in TaskJugglerUI.

12.8.3 Resources

Next you can define resources and assign those to work on specific tasks. You can group your resources hierarchically. Tag the top node of the resources with `:taskjuggler_resource:` (or whatever you customized `org-export-taskjuggler-resource-tag` to). You can optionally assign an identifier (named '`resource_id`') to the resources (using the standard Org properties commands, see Section 7.1 [Property syntax], page 57) or you can let the exporter generate identifiers automatically (the exporter picks the first word of the headline as the identifier as long as it is unique—see the documentation of `org-taskjuggler-get-unique-id`). Using that identifier you can then allocate resources to tasks. This is again done with the '`allocate`' property on the tasks. Do this in column view or when on the task type `C-c C-x p allocate RET <resource_id> RET`.

Once the allocations are done you can again export to TaskJuggler and check in the Resource Allocation Graph which person is working on what task at what time.

12.8.4 Export of properties

The exporter also takes TODO state information into consideration, i.e. if a task is marked as done it will have the corresponding attribute in TaskJuggler ('`complete 100`'). Also it will export any property on a task resource or resource node which is known to TaskJuggler, such as '`limits`', '`vacation`', '`shift`', '`booking`', '`efficiency`', '`journalentry`', '`rate`' for resources or '`account`', '`start`', '`note`', '`duration`', '`end`', '`journalentry`', '`milestone`', '`reference`', '`responsible`', '`scheduling`', etc for tasks.

12.8.5 Dependencies

The exporter will handle dependencies that are defined in the tasks either with the '`ORDERED`' attribute (see Section 5.2.7 [TODO dependencies], page 45), with the '`BLOCKER`' attribute

(see 'org-depend.el') or alternatively with a 'depends' attribute. Both the 'BLOCKER' and the 'depends' attribute can be either 'previous-sibling' or a reference to an identifier (named 'task_id') which is defined for another task in the project. 'BLOCKER' and the 'depends' attribute can define multiple dependencies separated by either space or comma. You can also specify optional attributes on the dependency by simply appending it. The following examples should illustrate this:

```
* Preparation
  :PROPERTIES:
  :task_id:  preparation
  :ORDERED:  t
  :END:
* Training material
  :PROPERTIES:
  :task_id:  training_material
  :ORDERED:  t
  :END:
** Markup Guidelines
   :PROPERTIES:
   :Effort:   2.0
   :END:
** Workflow Guidelines
   :PROPERTIES:
   :Effort:   2.0
   :END:
* Presentation
  :PROPERTIES:
  :Effort:   2.0
  :BLOCKER:  training_material { gapduration 1d } preparation
  :END:
```

12.8.6 Reports

TaskJuggler can produce many kinds of reports (e.g. gantt chart, resource allocation, etc). The user defines what kind of reports should be generated for a project in the TaskJuggler file. The exporter will automatically insert some default reports in the file. These defaults are defined in `org-export-taskjuggler-default-reports`. They can be modified using customize along with a number of other options. For a more complete list, see *M-x customize-group RET org-export-taskjuggler RET*.

For more information and examples see the Org-taskjuggler tutorial at `http://orgmode.org/worg/org-tutorials/org-taskjuggler.html`.

12.9 Freemind export

The Freemind exporter was written by Lennart Borgman.

C-c C-e m `org-export-as-freemind`

Export as Freemind mind map 'myfile.mm'.

12.10 XOXO export

Org-mode contains an exporter that produces XOXO-style output. Currently, this exporter only handles the general outline structure and does not interpret any additional Org-mode features.

C-c C-e x `org-export-as-xoxo`
: Export as XOXO file '`myfile.html`'.

C-c C-e v x
: Export only the visible part of the document.

12.11 iCalendar エクスポート

Some people use Org-mode for keeping track of projects, but still prefer a standard calendar application for anniversaries and appointments. In this case it can be useful to show deadlines and other time-stamped items in Org files in the calendar application. Org-mode can export calendar information in the standard iCalendar format. If you also want to have TODO entries included in the export, configure the variable `org-icalendar-include-todo`. Plain timestamps are exported as VEVENT, and TODO items as VTODO. It will also create events from deadlines that are in non-TODO items. Deadlines and scheduling dates in TODO items will be used to set the start and due dates for the TODO entry[17]. As categories, it will use the tags locally defined in the heading, and the file/tree category[18]. See the variable `org-icalendar-alarm-time` for a way to assign alarms to entries with a time.

The iCalendar standard requires each entry to have a globally unique identifier (UID). Org creates these identifiers during export. If you set the variable `org-icalendar-store-UID`, the UID will be stored in the `:ID:` property of the entry and re-used next time you report this entry. Since a single entry can give rise to multiple iCalendar entries (as a timestamp, a deadline, a scheduled item, and as a TODO item), Org adds prefixes to the UID, depending on what triggered the inclusion of the entry. In this way the UID remains unique, but a synchronization program can still figure out from which entry all the different instances originate.

C-c C-e i `org-export-icalendar-this-file`
: Create iCalendar entries for the current file and store them in the same directory, using a file extension '`.ics`'.

C-c C-e I `org-export-icalendar-all-agenda-files`
: Like *C-c C-e i*, but do this for all files in `org-agenda-files`. For each of these files, a separate iCalendar file will be written.

C-c C-e c `org-export-icalendar-combine-agenda-files`
: Create a single large iCalendar file from all files in `org-agenda-files` and write it to the file given by `org-combined-agenda-icalendar-file`.

[17] See the variables `org-icalendar-use-deadline` and `org-icalendar-use-scheduled`.

[18] To add inherited tags or the TODO state, configure the variable `org-icalendar-categories`.

The export will honor SUMMARY, DESCRIPTION and LOCATION[19] properties if the selected entries have them. If not, the summary will be derived from the headline, and the description from the body (limited to `org-icalendar-include-body` characters).

How this calendar is best read and updated, depends on the application you are using. The FAQ covers this issue.

[19] The LOCATION property can be inherited from higher in the hierarchy if you configure `org-use-property-inheritance` accordingly.

13 Publishing

Org includes a publishing management system that allows you to configure automatic HTML conversion of *projects* composed of interlinked org files. You can also configure Org to automatically upload your exported HTML pages and related attachments, such as images and source code files, to a web server.

You can also use Org to convert files into PDF, or even combine HTML and PDF conversion so that files are available in both formats on the server.

Publishing has been contributed to Org by David O'Toole.

13.1 Configuration

Publishing needs significant configuration to specify files, destination and many other properties of a project.

13.1.1 The variable `org-publish-project-alist`

Publishing is configured almost entirely through setting the value of one variable, called `org-publish-project-alist`. Each element of the list configures one project, and may be in one of the two following forms:

```
("project-name" :property value :property value ...)
  i.e. a well-formed property list with alternating keys and values
or
("project-name" :components ("project-name" "project-name" ...))
```

In both cases, projects are configured by specifying property values. A project defines the set of files that will be published, as well as the publishing configuration to use when publishing those files. When a project takes the second form listed above, the individual members of the `:components` property are taken to be sub-projects, which group together files requiring different publishing options. When you publish such a "meta-project", all the components will also be published, in the sequence given.

13.1.2 Sources and destinations for files

Most properties are optional, but some should always be set. In particular, Org needs to know where to look for source files, and where to put published files.

`:base-directory`	Directory containing publishing source files
`:publishing-directory`	Directory where output files will be published. You can directly publish to a webserver using a file name syntax appropriate for the Emacs '`tramp`' package. Or you can publish to a local directory and use external tools to upload your website (see Section 13.2 [Uploading files], page 148).
`:preparation-function`	Function or list of functions to be called before starting the publishing process, for example, to run `make` for updating files to be published. The project property list is scoped into this call as the variable `project-plist`.

`:completion-function` Function or list of functions called after finishing the publishing process, for example, to change permissions of the resulting files. The project property list is scoped into this call as the variable `project-plist`.

13.1.3 Selecting files

By default, all files with extension '`.org`' in the base directory are considered part of the project. This can be modified by setting the properties

`:base-extension` Extension (without the dot!) of source files. This actually is a regular expression. Set this to the symbol `any` if you want to get all files in `:base-directory`, even without extension.

`:exclude` Regular expression to match file names that should not be published, even though they have been selected on the basis of their extension.

`:include` List of files to be included regardless of `:base-extension` and `:exclude`.

`:recursive` Non-nil means, check base-directory recursively for files to publish.

13.1.4 Publishing action

Publishing means that a file is copied to the destination directory and possibly transformed in the process. The default transformation is to export Org files as HTML files, and this is done by the function `org-publish-org-to-html` which calls the HTML exporter (see Section 12.5 [HTML export], page 127). But you also can publish your content as PDF files using `org-publish-org-to-pdf`, or as `ascii`, `latin1` or `utf8` encoded files using the corresponding functions. If you want to publish the Org file itself, but with *archived*, *commented*, and *tag-excluded* trees removed, use `org-publish-org-to-org` and set the parameters `:plain-source` and/or `:htmlized-source`. This will produce '`file.org`' and '`file.org.html`' in the publishing directory[1]. Other files like images only need to be copied to the publishing destination; for this you may use `org-publish-attachment`. For non-Org files, you always need to specify the publishing function:

`:publishing-function` Function executing the publication of a file. This may also be a list of functions, which will all be called in turn.
`:plain-source` Non-nil means, publish plain source.
`:htmlized-source` Non-nil means, publish htmlized source.

The function must accept three arguments: a property list containing at least a `:publishing-directory` property, the name of the file to be published, and the path to the publishing directory of the output file. It should take the specified file, make the necessary transformation (if any) and place the result into the destination folder.

[1] '`file-source.org`' and '`file-source.org.html`' if source and publishing directories are equal. Note that with this kind of setup, you need to add `:exclude "-source\\.org"` to the project definition in `org-publish-project-alist` to prevent the published source files from being considered as new org files the next time the project is published.

13.1.5 Options for the HTML/LaTeX exporters

The property list can be used to set many export options for the HTML and LaTeX exporters. In most cases, these properties correspond to user variables in Org. The table below lists these properties along with the variable they belong to. See the documentation string for the respective variable for details.

`:link-up`	`org-export-html-link-up`
`:link-home`	`org-export-html-link-home`
`:language`	`org-export-default-language`
`:customtime`	`org-display-custom-times`
`:headline-levels`	`org-export-headline-levels`
`:section-numbers`	`org-export-with-section-numbers`
`:section-number-format`	`org-export-section-number-format`
`:table-of-contents`	`org-export-with-toc`
`:preserve-breaks`	`org-export-preserve-breaks`
`:archived-trees`	`org-export-with-archived-trees`
`:emphasize`	`org-export-with-emphasize`
`:sub-superscript`	`org-export-with-sub-superscripts`
`:special-strings`	`org-export-with-special-strings`
`:footnotes`	`org-export-with-footnotes`
`:drawers`	`org-export-with-drawers`
`:tags`	`org-export-with-tags`
`:todo-keywords`	`org-export-with-todo-keywords`
`:priority`	`org-export-with-priority`
`:TeX-macros`	`org-export-with-TeX-macros`
`:LaTeX-fragments`	`org-export-with-LaTeX-fragments`
`:latex-listings`	`org-export-latex-listings`
`:skip-before-1st-heading`	`org-export-skip-text-before-1st-heading`
`:fixed-width`	`org-export-with-fixed-width`
`:timestamps`	`org-export-with-timestamps`
`:author`	`user-full-name`
`:email`	`user-mail-address` : `addr;addr;..`
`:author-info`	`org-export-author-info`
`:email-info`	`org-export-email-info`
`:creator-info`	`org-export-creator-info`
`:tables`	`org-export-with-tables`
`:table-auto-headline`	`org-export-highlight-first-table-line`
`:style-include-default`	`org-export-html-style-include-default`
`:style-include-scripts`	`org-export-html-style-include-scripts`
`:style`	`org-export-html-style`
`:style-extra`	`org-export-html-style-extra`
`:convert-org-links`	`org-export-html-link-org-files-as-html`
`:inline-images`	`org-export-html-inline-images`
`:html-extension`	`org-export-html-extension`
`:html-preamble`	`org-export-html-preamble`
`:html-postamble`	`org-export-html-postamble`
`:xml-declaration`	`org-export-html-xml-declaration`

```
:html-table-tag           org-export-html-table-tag
:expand-quoted-html       org-export-html-expand
:timestamp                org-export-html-with-timestamp
:publishing-directory     org-export-publishing-directory
:select-tags              org-export-select-tags
:exclude-tags             org-export-exclude-tags
:latex-image-options      org-export-latex-image-default-option
```

Most of the `org-export-with-*` variables have the same effect in both HTML and LaTeX exporters, except for `:TeX-macros` and `:LaTeX-fragments` options, respectively `nil` and `t` in the LaTeX export. See `org-export-plist-vars` to check this list of options.

When a property is given a value in `org-publish-project-alist`, its setting overrides the value of the corresponding user variable (if any) during publishing. Options set within a file (see Section 12.2 [Export options], page 124), however, override everything.

13.1.6 Links between published files

To create a link from one Org file to another, you would use something like '`[[file:foo.org][The foo]]`' or simply '`file:foo.org.`' (see Chapter 4 [Hyperlinks], page 33). When published, this link becomes a link to '`foo.html`'. In this way, you can interlink the pages of your "org web" project and the links will work as expected when you publish them to HTML. If you also publish the Org source file and want to link to that, use an `http:` link instead of a `file:` link, because `file:` links are converted to link to the corresponding '`html`' file.

You may also link to related files, such as images. Provided you are careful with relative file names, and provided you have also configured Org to upload the related files, these links will work too. See Section 13.3.2 [Complex example], page 149, for an example of this usage.

Sometimes an Org file to be published may contain links that are only valid in your production environment, but not in the publishing location. In this case, use the property

`:link-validation-function` Function to validate links

to define a function for checking link validity. This function must accept two arguments, the file name and a directory relative to which the file name is interpreted in the production environment. If this function returns `nil`, then the HTML generator will only insert a description into the HTML file, but no link. One option for this function is `org-publish-validate-link` which checks if the given file is part of any project in `org-publish-project-alist`.

13.1.7 Generating a sitemap

The following properties may be used to control publishing of a map of files for a given project.

`:auto-sitemap` When non-nil, publish a sitemap during `org-publish-current-project` or `org-publish-all`.

`:sitemap-filename` Filename for output of sitemap. Defaults to '`sitemap.org`' (which becomes '`sitemap.html`').

`:sitemap-title`	Title of sitemap page. Defaults to name of file.
`:sitemap-function`	Plug-in function to use for generation of the sitemap. Defaults to `org-publish-org-sitemap`, which generates a plain list of links to all files in the project.
`:sitemap-sort-folders`	Where folders should appear in the sitemap. Set this to `first` (default) or `last` to display folders first or last, respectively. Any other value will mix files and folders.
`:sitemap-sort-files`	How the files are sorted in the site map. Set this to `alphabetically` (default), `chronologically` or `anti-chronologically`. `chronologically` sorts the files with older date first while `anti-chronologically` sorts the files with newer date first. `alphabetically` sorts the files alphabetically. The date of a file is retrieved with `org-publish-find-date`.
`:sitemap-ignore-case`	Should sorting be case-sensitive? Default `nil`.
`:sitemap-file-entry-format`	With this option one can tell how a sitemap's entry is formated in the sitemap. This is a format string with some escape sequences: `%t` stands for the title of the file, `%a` stands for the author of the file and `%d` stands for the date of the file. The date is retrieved with the `org-publish-find-date` function and formated with `org-publish-sitemap-date-format`. Default `%t`.
`:sitemap-date-format`	Format string for the `format-time-string` function that tells how a sitemap entry's date is to be formated. This property bypasses `org-publish-sitemap-date-format` which defaults to `%Y-%m-%d`.

13.1.8 Generating an index

Org-mode can generate an index across the files of a publishing project.

`:makeindex`	When non-nil, generate in index in the file '`theindex.org`' and publish it as '`theindex.html`'.

The file will be created when first publishing a project with the `:makeindex` set. The file only contains a statement `#+include: "theindex.inc"`. You can then build around this include statement by adding a title, style information, etc.

13.2 Uploading files

For those people already utilizing third party sync tools such as `rsync` or `unison`, it might be preferable not to use the built in *remote* publishing facilities of Org-mode which rely

heavily on Tramp. Tramp, while very useful and powerful, tends not to be so efficient for multiple file transfer and has been known to cause problems under heavy usage.

Specialized synchronization utilities offer several advantages. In addition to timestamp comparison, they also do content and permissions/attribute checks. For this reason you might prefer to publish your web to a local directory (possibly even *in place* with your Org files) and then use '`unison`' or '`rsync`' to do the synchronization with the remote host.

Since Unison (for example) can be configured as to which files to transfer to a certain remote destination, it can greatly simplify the project publishing definition. Simply keep all files in the correct location, process your Org files with `org-publish` and let the synchronization tool do the rest. You do not need, in this scenario, to include attachments such as '`jpg`', '`css`' or '`gif`' files in the project definition since the 3rd party tool syncs them.

Publishing to a local directory is also much faster than to a remote one, so that you can afford more easily to republish entire projects. If you set `org-publish-use-timestamps-flag` to `nil`, you gain the main benefit of re-including any changed external files such as source example files you might include with `#+INCLUDE`. The timestamp mechanism in Org is not smart enough to detect if included files have been modified.

13.3 Sample configuration

Below we provide two example configurations. The first one is a simple project publishing only a set of Org files. The second example is more complex, with a multi-component project.

13.3.1 Example: simple publishing configuration

This example publishes a set of Org files to the '`public_html`' directory on the local machine.

```
(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\"/>")))
```

13.3.2 Example: complex publishing configuration

This more complicated example publishes an entire website, including Org files converted to HTML, image files, Emacs Lisp source code, and style sheets. The publishing directory is remote and private files are excluded.

To ensure that links are preserved, care should be taken to replicate your directory structure on the web server, and to use relative file paths. For example, if your Org files are kept in '`~/org`' and your publishable images in '`~/images`', you would link to an image with

```
file:../images/myimage.png
```

On the web server, the relative path to the image should be the same. You can accomplish this by setting up an "images" folder in the right place on the web server, and publishing images to it.

```
(setq org-publish-project-alist
      '(("orgfiles"
          :base-directory "~/org/"
          :base-extension "org"
          :publishing-directory "/ssh:user@host:~/html/notebook/"
          :publishing-function org-publish-org-to-html
          :exclude "PrivatePage.org"   ;; regexp
          :headline-levels 3
          :section-numbers nil
          :table-of-contents nil
          :style "<link rel=\"stylesheet\"
                  href=\"../other/mystyle.css\" type=\"text/css\"/>"
          :html-preamble t)

         ("images"
          :base-directory "~/images/"
          :base-extension "jpg\\|gif\\|png"
          :publishing-directory "/ssh:user@host:~/html/images/"
          :publishing-function org-publish-attachment)

         ("other"
          :base-directory "~/other/"
          :base-extension "css\\|el"
          :publishing-directory "/ssh:user@host:~/html/other/"
          :publishing-function org-publish-attachment)
         ("website" :components ("orgfiles" "images" "other"))))
```

13.4 公開の開始

Once properly configured, Org can publish with the following commands:

C-c C-e X `org-publish`
: Prompt for a specific project and publish all files that belong to it.

C-c C-e P `org-publish-current-project`
: Publish the project containing the current file.

C-c C-e F `org-publish-current-file`
: Publish only the current file.

C-c C-e E `org-publish-all`
: Publish every project.

Org uses timestamps to track when a file has changed. The above functions normally only publish changed files. You can override this and force publishing of all files by giving a prefix argument to any of the commands above, or by customizing the variable `org-`

`publish-use-timestamps-flag`. This may be necessary in particular if files include other files via `#+SETUPFILE:` or `#+INCLUDE:`.

14 ソースコードとの連携

Source code can be included in Org-mode documents using a '`src`' block, e.g.

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

Org-mode provides a number of features for working with live source code, including editing of code blocks in their native major-mode, evaluation of code blocks, converting code blocks into source files (known as *tangling* in literate programming), and exporting code blocks and their results in several formats. This functionality was contributed by Eric Schulte and Dan Davison, and was originally named Org-babel.

The following sections describe Org-mode's code block handling facilities.

14.1 Structure of code blocks

The structure of code blocks is as follows:

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

Switches and header arguments are optional. Code can also be embedded in text inline using

```
src_<language>{<body>}
```

or

```
src_<language>[<header arguments>]{<body>}
```

`<name>` This name is associated with the code block. This is similar to the '`#+tblname`' lines that can be used to name tables in Org-mode files. Referencing the name of a code block makes it possible to evaluate the block from other places in the file, other files, or from Org-mode table formulas (see Section 3.5 [The spreadsheet], page 23).

`<language>`
The language of the code in the block.

`<switches>`
Optional switches controlling exportation of the code block (see switches discussion in Section 11.3 [Literal examples], page 117)

`<header arguments>`
Optional header arguments control many aspects of evaluation, export and tangling of code blocks. See the Section 14.8 [Header arguments], page 156 section. Header arguments can also be set on a per-buffer or per-subtree basis using properties.

`<body>` The source code.

14.2 Editing source code

Use `C-c '` to edit the current code block. This brings up a language major-mode edit buffer containing the body of the code block. Saving this buffer will write the new contents back to the Org buffer. Use `C-c '` again to exit.

The `org-src-mode` minor mode will be active in the edit buffer. The following variables can be used to configure the behavior of the edit buffer. See also the customization group `org-edit-structure` for further configuration options.

`org-src-lang-modes`
: If an Emacs major-mode named `<lang>-mode` exists, where `<lang>` is the language named in the header line of the code block, then the edit buffer will be placed in that major-mode. This variable can be used to map arbitrary language names to existing major modes.

`org-src-window-setup`
: Controls the way Emacs windows are rearranged when the edit buffer is created.

`org-src-preserve-indentation`
: This variable is especially useful for tangling languages such as Python, in which whitespace indentation in the output is critical.

`org-src-ask-before-returning-to-edit-buffer`
: By default, Org will ask before returning to an open edit buffer. Set this variable to nil to switch without asking.

To turn on native code fontification in the *Org* buffer, configure the variable `org-src-fontify-natively`.

14.3 Exporting code blocks

It is possible to export the *contents* of code blocks, the *results* of code block evaluation, *neither*, or *both*. For most languages, the default exports the contents of code blocks. However, for some languages (e.g. `ditaa`) the default exports the results of code block evaluation. For information on exporting code block bodies, see Section 11.3 [Literal examples], page 117.

The `:exports` header argument can be used to specify export behavior:

Header arguments:

`:exports code`
: The default in most languages. The body of the code block is exported, as described in Section 11.3 [Literal examples], page 117.

`:exports results`
: The code block will be evaluated and the results will be placed in the Org-mode buffer for export, either updating previous results of the code block located anywhere in the buffer or, if no previous results exist, placing the results immediately after the code block. The body of the code block will not be exported.

`:exports both`
: Both the code block and its results will be exported.

`:exports none`
: Neither the code block nor its results will be exported.

It is possible to inhibit the evaluation of code blocks during export. Setting the `org-export-babel-evaluate` variable to `nil` will ensure that no code blocks are evaluated as part of the export process. This can be useful in situations where potentially untrusted Org-mode files are exported in an automated fashion, for example when Org-mode is used as the markup language for a wiki.

14.4 Extracting source code

Creating pure source code files by extracting code from source blocks is referred to as "tangling"—a term adopted from the literate programming community. During "tangling" of code blocks their bodies are expanded using `org-babel-expand-src-block` which can expand both variable and "noweb" style references (see Section 14.10 [Noweb reference syntax], page 171).

Header arguments

`:tangle no`
: The default. The code block is not included in the tangled output.

`:tangle yes`
: Include the code block in the tangled output. The output file name is the name of the org file with the extension '`.org`' replaced by the extension for the block language.

`:tangle filename`
: Include the code block in the tangled output to file '`filename`'.

Functions

`org-babel-tangle`
: Tangle the current file. Bound to *C-c C-v t*.

`org-babel-tangle-file`
: Choose a file to tangle. Bound to *C-c C-v f*.

Hooks

`org-babel-post-tangle-hook`
: This hook is run from within code files tangled by `org-babel-tangle`. Example applications could include post-processing, compilation or evaluation of tangled code files.

14.5 Evaluating code blocks

Code blocks can be evaluated[1] and the results placed in the Org-mode buffer. By default, evaluation is only turned on for `emacs-lisp` code blocks, however support exists for evaluating blocks in many languages. See Section 14.7 [Languages], page 156 for a list of supported

[1] Whenever code is evaluated there is a potential for that code to do harm. Org-mode provides a number of safeguards to ensure that it only evaluates code with explicit confirmation from the user. For information on these safeguards (and on how to disable them) see Section 15.4 [Code evaluation security], page 175.

languages. See Section 14.1 [Structure of code blocks], page 152 for information on the syntax used to define a code block.

There are a number of ways to evaluate code blocks. The simplest is to press `C-c C-c` or `C-c C-v e` with the point on a code block[2]. This will call the `org-babel-execute-src-block` function to evaluate the block and insert its results into the Org-mode buffer.

It is also possible to evaluate named code blocks from anywhere in an Org-mode buffer or an Org-mode table. `#+call` (or synonymously `#+function` or `#+lob`) lines can be used to remotely execute code blocks located in the current Org-mode buffer or in the "Library of Babel" (see Section 14.6 [Library of Babel], page 155). These lines use the following syntax.

```
#+call: <name>(<arguments>) <header arguments>
#+function: <name>(<arguments>) <header arguments>
#+lob: <name>(<arguments>) <header arguments>
```

`<name>` The name of the code block to be evaluated.

`<arguments>`
: Arguments specified in this section will be passed to the code block. These arguments should relate to `:var` header arguments in the called code block expressed using standard function call syntax. For example if the original code block named `double` has the header argument `:var n=2`, then the call line passing the number four to that block would be written as `#+call: double(n=2)`.

`<header arguments>`
: Header arguments can be placed after the function invocation. See Section 14.8 [Header arguments], page 156 for more information on header arguments.

All header arguments placed in the `<header arguments>` section described above will be applied to the evaluation of the `#+call:` line, however it is sometimes desirable to specify header arguments to be passed to the code block being evaluated.

This is possible through the use of the following optional extended syntax.

```
#+call: <name>[<block header arguments>](<arguments>) <header arguments>
```

Any header argument placed between the square brackets in the `<block header arguments>` section will be applied to the evaluation of the named code block. For more examples of passing header arguments to `#+call:` lines see [Header arguments in function calls], page 158.

14.6 Library of Babel

The "Library of Babel" is a library of code blocks that can be called from any Org-mode file. The library is housed in an Org-mode file located in the '`contrib`' directory of Org-mode. Org-mode users can deposit functions they believe to be generally useful in the library.

Code blocks defined in the "Library of Babel" can be called remotely as if they were in the current Org-mode buffer (see Section 14.5 [Evaluating code blocks], page 154 for information on the syntax of remote code block evaluation).

[2] The `org-babel-no-eval-on-ctrl-c-ctrl-c` variable can be used to remove code evaluation from the `C-c C-c` key binding.

Code blocks located in any Org-mode file can be loaded into the “Library of Babel” with the `org-babel-lob-ingest` function, bound to *C-c C-v i*.

14.7 Languages

Code blocks in the following languages are supported.

Language	Identifier	Language	Identifier
Asymptote	asymptote	Emacs Calc	calc
C	C	C++	C++
Clojure	clojure	CSS	css
ditaa	ditaa	Graphviz	dot
Emacs Lisp	emacs-lisp	gnuplot	gnuplot
Haskell	haskell	Javascript	js
LaTeX	latex	Ledger	ledger
Lisp	lisp	MATLAB	matlab
Mscgen	mscgen	Objective Caml	ocaml
Octave	octave	Org-mode	org
Oz	oz	Perl	perl
Plantuml	plantuml	Python	python
R	R	Ruby	ruby
Sass	sass	Scheme	scheme
GNU Screen	screen	shell	sh
SQL	sql	SQLite	sqlite

Language-specific documentation is available for some languages. If available, it can be found at `http://orgmode.org/worg/org-contrib/babel/languages`.

The `org-babel-load-languages` controls which languages are enabled for evaluation (by default only `emacs-lisp` is enabled). This variable can be set using the customization interface or by adding code like the following to your emacs configuration.

The following disables `emacs-lisp` evaluation and enables evaluation of `R` code blocks.

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((emacs-lisp . nil)
   (R . t)))
```

It is also possible to enable support for a language by loading the related elisp file with `require`.

The following adds support for evaluating `clojure` code blocks.

```
(require 'ob-clojure)
```

14.8 Header arguments

Code block functionality can be configured with header arguments. This section provides an overview of the use of header arguments, and then describes each header argument in detail.

14.8.1 Using header arguments

The values of header arguments can be set in six different ways, each more specific (and having higher priority) than the last.

System-wide header arguments

System-wide values of header arguments can be specified by customizing the `org-babel-default-header-args` variable:

```
:session    => "none"
:results    => "replace"
:exports    => "code"
:cache      => "no"
:noweb      => "no"
```

For example, the following example could be used to set the default value of `:noweb` header arguments to `yes`. This would have the effect of expanding `:noweb` references by default when evaluating source code blocks.

```
(setq org-babel-default-header-args
(cons '(:noweb . "yes")
(assq-delete-all :noweb org-babel-default-header-args)))
```

Language-specific header arguments

Each language can define its own set of default header arguments. See the language-specific documentation available online at `http://orgmode.org/worg/org-contrib/babel`.

Buffer-wide header arguments

Buffer-wide header arguments may be specified through the use of a special line placed anywhere in an Org-mode file. The line consists of the `#+BABEL:` keyword followed by a series of header arguments which may be specified using the standard header argument syntax.

For example the following would set `session` to `*R*`, and `results` to `silent` for every code block in the buffer, ensuring that all execution took place in the same session, and no results would be inserted into the buffer.

```
#+BABEL: :session *R* :results silent
```

Header arguments in Org-mode properties

Header arguments are also read from Org-mode properties (see Section 7.1 [Property syntax], page 57), which can be set on a buffer-wide or per-heading basis. An example of setting a header argument for all code blocks in a buffer is

```
#+property: tangle yes
```

When properties are used to set default header arguments, they are looked up with inheritance, so the value of the `:cache` header argument will default to `yes` in all code blocks in the subtree rooted at the following heading:

```
* outline header
:PROPERTIES:
:cache:    yes
```

```
:END:
```

Properties defined in this way override the properties set in `org-babel-default-header-args`. It is convenient to use the `org-set-property` function bound to *C-c C-x p* to set properties in Org-mode documents.

Code block specific header arguments

The most common way to assign values to header arguments is at the code block level. This can be done by listing a sequence of header arguments and their values as part of the `#+begin_src` line. Properties set in this way override both the values of `org-babel-default-header-args` and header arguments specified as properties. In the following example, the `:results` header argument is set to `silent`, meaning the results of execution will not be inserted in the buffer, and the `:exports` header argument is set to `code`, meaning only the body of the code block will be preserved on export to HTML or LaTeX.

```
#+source: factorial
#+begin_src haskell :results silent :exports code :var n=0
fac 0 = 1
fac n = n * fac (n-1)
#+end_src
```

Similarly, it is possible to set header arguments for inline code blocks:

```
src_haskell[:exports both]{fac 5}
```

Code block header arguments can span multiple lines using =#+header:= or =#+headers:= lines preceding a code block or nested in between the name and body of a named code block.

Multi-line header arguments on an un-named code block:

```
#+headers: :var data1=1
#+begin_src emacs-lisp :var data2=2
  (message "data1:%S, data2:%S" data1 data2)
#+end_src

#+results:
: data1:1, data2:2
```

Multi-line header arguments on a named code block:

```
#+source: named-block
#+header: :var data=2
#+begin_src emacs-lisp
  (message "data:%S" data)
#+end_src

#+results: named-block
: data:2
```

Header arguments in function calls

At the most specific level, header arguments for "Library of Babel" or function call lines can be set as shown in the two examples below. For more information on the structure of `#+call:` lines see Section 14.5 [Evaluating code blocks], page 154.

The following will apply the `:exports results` header argument to the evaluation of the `#+call:` line.

```
#+call: factorial(n=5) :exports results
```

The following will apply the `:session special` header argument to the evaluation of the `factorial` code block.

```
#+call: factorial[:session special](n=5)
```

14.8.2 Specific header arguments

The following header arguments are defined:

14.8.2.1 `:var`

The `:var` header argument is used to pass arguments to code blocks. The specifics of how arguments are included in a code block vary by language; these are addressed in the language-specific documentation. However, the syntax used to specify arguments is the same across all languages. The values passed to arguments can be literal values, values from org-mode tables and literal example blocks, the results of other code blocks, or Emacs Lisp code—see the "Emacs Lisp evaluation of variables" heading below.

These values can be indexed in a manner similar to arrays—see the "indexable variable values" heading below.

The following syntax is used to pass arguments to code blocks using the `:var` header argument.

```
:var name=assign
```

where `assign` can take one of the following forms

- literal value either a string `"string"` or a number `9`.
- reference a table name:

```
#+tblname: example-table
| 1 |
| 2 |
| 3 |
| 4 |

#+source: table-length
#+begin_src emacs-lisp :var table=example-table
(length table)
#+end_src

#+results: table-length
: 4
```

  a code block name, as assigned by `#+srcname:`, followed by parentheses:

```
#+begin_src emacs-lisp :var length=table-length()
(* 2 length)
#+end_src

#+results:
```

```
: 8
```

In addition, an argument can be passed to the code block referenced by `:var`. The argument is passed within the parentheses following the code block name:

```
#+source: double
#+begin_src emacs-lisp :var input=8
(* 2 input)
#+end_src

#+results: double
: 16

#+source: squared
#+begin_src emacs-lisp :var input=double(input=1)
(* input input)
#+end_src

#+results: squared
: 4
```

Alternate argument syntax

It is also possible to specify arguments in a potentially more natural way using the `#+source:` line of a code block. As in the following example arguments can be packed inside of parenthesis, separated by commas, following the source name.

```
#+source: double(input=0, x=2)
#+begin_src emacs-lisp
(* 2 (+ input x))
#+end_src
```

Indexable variable values

It is possible to reference portions of variable values by "indexing" into the variables. Indexes are 0 based with negative values counting back from the end. If an index is separated by `,`s then each subsequent section will index into the next deepest nesting or dimension of the value. Note that this indexing occurs *before* other table related header arguments like `:hlines`, `:colnames` and `:rownames` are applied. The following example assigns the last cell of the first row the table `example-table` to the variable `data`:

```
#+results: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |

#+begin_src emacs-lisp :var data=example-table[0,-1]
  data
#+end_src

#+results:
```

```
: a
```

Ranges of variable values can be referenced using two integers separated by a `:`, in which case the entire inclusive range is referenced. For example the following assigns the middle three rows of `example-table` to `data`.

```
#+results: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |
| 5 | 3 |

#+begin_src emacs-lisp :var data=example-table[1:3]
  data
#+end_src

#+results:
| 2 | b |
| 3 | c |
| 4 | d |
```

Additionally, an empty index, or the single character `*`, are both interpreted to mean the entire range and as such are equivalent to `0:-1`, as shown in the following example in which the entire first column is referenced.

```
#+results: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |

#+begin_src emacs-lisp :var data=example-table[,0]
  data
#+end_src

#+results:
| 1 | 2 | 3 | 4 |
```

It is possible to index into the results of code blocks as well as tables. Any number of dimensions can be indexed. Dimensions are separated from one another by commas, as shown in the following example.

```
#+source: 3D
#+begin_src emacs-lisp
  '(((1  2  3)  (4  5  6)  (7  8  9))
    ((10 11 12) (13 14 15) (16 17 18))
    ((19 20 21) (22 23 24) (25 26 27)))
#+end_src

#+begin_src emacs-lisp :var data=3D[1,,1]
```

```
  data
#+end_src

#+results:
| 11 | 14 | 17 |
```

Emacs Lisp evaluation of variables

Emacs lisp code can be used to initialize variable values. When a variable value starts with `(`, `[`, `'` or `` ` `` it will be evaluated as Emacs Lisp and the result of the evaluation will be assigned as the variable value. The following example demonstrates use of this evaluation to reliably pass the file-name of the org-mode buffer to a code block—note that evaluation of header arguments is guaranteed to take place in the original org-mode file, while there is no such guarantee for evaluation of the code block body.

```
#+begin_src sh :var file-name=(buffer-file-name) :exports both
  wc -w $file
#+end_src
```

Note that values read from tables and lists will not be evaluated as Emacs Lisp, as shown in the following example.

```
#+results: table
| (a b c) |

#+headers: :var data=table[0,0]
#+begin_src perl
  $data
#+end_src

#+results:
: (a b c)
```

14.8.2.2 `:results`

There are three classes of `:results` header argument. Only one option per class may be supplied per code block.

- **collection** header arguments specify how the results should be collected from the code block
- **type** header arguments specify what type of result the code block will return—which has implications for how they will be inserted into the Org-mode buffer
- **handling** header arguments specify how the results of evaluating the code block should be handled.

Collection

The following options are mutually exclusive, and specify how the results should be collected from the code block.

- `value` This is the default. The result is the value of the last statement in the code block. This header argument places the evaluation in functional mode. Note that in

some languages, e.g., Python, use of this result type requires that a `return` statement be included in the body of the source code block. E.g., `:results value`.

- `output` The result is the collection of everything printed to STDOUT during the execution of the code block. This header argument places the evaluation in scripting mode. E.g., `:results output`.

Type

The following options are mutually exclusive and specify what type of results the code block will return. By default, results are inserted as either a table or scalar depending on their value.

- `table`, `vector` The results should be interpreted as an Org-mode table. If a single value is returned, it will be converted into a table with one row and one column. E.g., `:results value table`.
- `list` The results should be interpreted as an Org-mode list. If a single scalar value is returned it will be converted into a list with only one element.
- `scalar`, `verbatim` The results should be interpreted literally—they will not be converted into a table. The results will be inserted into the Org-mode buffer as quoted text. E.g., `:results value verbatim`.
- `file` The results will be interpreted as the path to a file, and will be inserted into the Org-mode buffer as a file link. E.g., `:results value file`.
- `raw`, `org` The results are interpreted as raw Org-mode code and are inserted directly into the buffer. If the results look like a table they will be aligned as such by Org-mode. E.g., `:results value raw`.
- `html` Results are assumed to be HTML and will be enclosed in a `begin_html` block. E.g., `:results value html`.
- `latex` Results assumed to be LaTeX and are enclosed in a `begin_latex` block. E.g., `:results value latex`.
- `code` Result are assumed to be parseable code and are enclosed in a code block. E.g., `:results value code`.
- `pp` The result is converted to pretty-printed code and is enclosed in a code block. This option currently supports Emacs Lisp, Python, and Ruby. E.g., `:results value pp`.
- `wrap` The result is wrapped in a `begin_result` block. This can be useful for inserting `raw` or `org` syntax results in such a way that their extend is known and they can be automatically removed or replaced.

Handling

The following results options indicate what happens with the results once they are collected.

- `silent` The results will be echoed in the minibuffer but will not be inserted into the Org-mode buffer. E.g., `:results output silent`.
- `replace` The default value. Any existing results will be removed, and the new results will be inserted into the Org-mode buffer in their place. E.g., `:results output replace`.

- `append` If there are pre-existing results of the code block then the new results will be appended to the existing results. Otherwise the new results will be inserted as with `replace`.
- `prepend` If there are pre-existing results of the code block then the new results will be prepended to the existing results. Otherwise the new results will be inserted as with `replace`.

14.8.2.3 `:file`

The header argument `:file` is used to specify an external file in which to save code block results. After code block evaluation an Org-mode style `[[file:]]` link (see Section 4.1 [Link format], page 33) to the file will be inserted into the Org-mode buffer. Some languages including R, gnuplot, dot, and ditaa provide special handling of the `:file` header argument automatically wrapping the code block body in the boilerplate code required to save output to the specified file. This is often useful for saving graphical output of a code block to the specified file.

The argument to `:file` should be either a string specifying the path to a file, or a list of two strings in which case the first element of the list should be the path to a file and the second a description for the link.

14.8.2.4 `:dir` and remote execution

While the `:file` header argument can be used to specify the path to the output file, `:dir` specifies the default directory during code block execution. If it is absent, then the directory associated with the current buffer is used. In other words, supplying `:dir path` temporarily has the same effect as changing the current directory with *M-x cd path*, and then not supplying `:dir`. Under the surface, `:dir` simply sets the value of the Emacs variable `default-directory`.

When using `:dir`, you should supply a relative path for file output (e.g. `:file myfile.jpg` or `:file results/myfile.jpg`) in which case that path will be interpreted relative to the default directory.

In other words, if you want your plot to go into a folder called 'Work' in your home directory, you could use

```
#+begin_src R :file myplot.png :dir ~/Work
matplot(matrix(rnorm(100), 10), type="l")
#+end_src
```

Remote execution

A directory on a remote machine can be specified using tramp file syntax, in which case the code will be evaluated on the remote machine. An example is

```
#+begin_src R :file plot.png :dir /dand@yakuba.princeton.edu:
plot(1:10, main=system("hostname", intern=TRUE))
#+end_src
```

Text results will be returned to the local Org-mode buffer as usual, and file output will be created on the remote machine with relative paths interpreted relative to the remote directory. An Org-mode link to the remote file will be created.

So, in the above example a plot will be created on the remote machine, and a link of the following form will be inserted in the org buffer:

```
[[file:/scp:dand@yakuba.princeton.edu:/home/dand/plot.png][plot.png]]
```

Most of this functionality follows immediately from the fact that `:dir` sets the value of the Emacs variable `default-directory`, thanks to tramp. Those using XEmacs, or GNU Emacs prior to version 23 may need to install tramp separately in order for these features to work correctly.

Further points

- If `:dir` is used in conjunction with `:session`, although it will determine the starting directory for a new session as expected, no attempt is currently made to alter the directory associated with an existing session.
- `:dir` should typically not be used to create files during export with `:exports results` or `:exports both`. The reason is that, in order to retain portability of exported material between machines, during export links inserted into the buffer will *not* be expanded against `default directory`. Therefore, if `default-directory` is altered using `:dir`, it is probable that the file will be created in a location to which the link does not point.

14.8.2.5 `:exports`

The `:exports` header argument specifies what should be included in HTML or LaTeX exports of the Org-mode file.

- `code` The default. The body of code is included into the exported file. E.g., `:exports code`.
- `results` The result of evaluating the code is included in the exported file. E.g., `:exports results`.
- `both` Both the code and results are included in the exported file. E.g., `:exports both`.
- `none` Nothing is included in the exported file. E.g., `:exports none`.

14.8.2.6 `:tangle`

The `:tangle` header argument specifies whether or not the code block should be included in tangled extraction of source code files.

- 抽出The code block is exported to a source code file named after the basename (name w/o extension) of the Org-mode file. E.g., `:tangle yes`.
- `no` The default. The code block is not exported to a source code file. E.g., `:tangle no`.
- other Any other string passed to the `:tangle` header argument is interpreted as a file basename to which the block will be exported. E.g., `:tangle basename`.

14.8.2.7 `:mkdirp`

The `:mkdirp` header argument can be used to create parent directories of tangled files when missing. This can be set to `yes` to enable directory creation or to `no` to inhibit directory creation.

14.8.2.8 :comments

By default code blocks are tangled to source-code files without any insertion of comments beyond those which may already exist in the body of the code block. The `:comments` header argument can be set as follows to control the insertion of extra comments into the tangled code file.

- `no` The default. No extra comments are inserted during tangling.
- `link` The code block is wrapped in comments which contain pointers back to the original Org file from which the code was tangled.
- `yes` A synonym for "link" to maintain backwards compatibility.
- `org` Include text from the org-mode file as a comment.

  The text is picked from the leading context of the tangled code and is limited by the nearest headline or source block as the case may be.
- `both` Turns on both the "link" and "org" comment options.
- `noweb` Turns on the "link" comment option, and additionally wraps expanded noweb references in the code block body in link comments.

14.8.2.9 :no-expand

By default, code blocks are expanded with `org-babel-expand-src-block` during tangling. This has the effect of assigning values to variables specified with `:var` (see Section 14.8.2.1 [var], page 159), and of replacing "noweb" references (see Section 14.10 [Noweb reference syntax], page 171) with their targets. The `:no-expand` header argument can be used to turn off this behavior.

14.8.2.10 :session

The `:session` header argument starts a session for an interpreted language where state is preserved.

By default, a session is not started.

A string passed to the `:session` header argument will give the session a name. This makes it possible to run concurrent sessions for each interpreted language.

14.8.2.11 :noweb

The `:noweb` header argument controls expansion of "noweb" style (see Section 14.10 [Noweb reference syntax], page 171) references in a code block. This header argument can have one of three values: `yes` `no` or `tangle`.

- `yes` All "noweb" syntax references in the body of the code block will be expanded before the block is evaluated, tangled or exported.
- `no` The default. No "noweb" syntax specific action is taken on evaluating code blocks, However, noweb references will still be expanded during tangling.
- 抽出All "noweb" syntax references in the body of the code block will be expanded before the block is tangled, however "noweb" references will not be expanded when the block is evaluated or exported.

Noweb prefix lines

Noweb insertions are now placed behind the line prefix of the `<<reference>>`. This behavior is illustrated in the following example. Because the `<<example>>` noweb reference appears behind the SQL comment syntax, each line of the expanded noweb reference will be commented.

This code block:

```
-- <<example>>
```

expands to:

```
-- this is the
-- multi-line body of example
```

Note that noweb replacement text that does not contain any newlines will not be affected by this change, so it is still possible to use inline noweb references.

14.8.2.12 `:cache`

The `:cache` header argument controls the use of in-buffer caching of the results of evaluating code blocks. It can be used to avoid re-evaluating unchanged code blocks. This header argument can have one of two values: `yes` or `no`.

- `no` The default. No caching takes place, and the code block will be evaluated every time it is called.
- `yes` Every time the code block is run a SHA1 hash of the code and arguments passed to the block will be generated. This hash is packed into the `#+results:` line and will be checked on subsequent executions of the code block. If the code block has not changed since the last time it was evaluated, it will not be re-evaluated.

Code block caches notice if the value of a variable argument to the code block has changed. If this is the case, the cache is invalidated and the code block is re-run. In the following example, `caller` will not be re-run unless the results of `random` have changed since it was last run.

```
#+srcname: random
#+begin_src R :cache yes
runif(1)
#+end_src

#+results[a2a72cd647ad44515fab62e144796432793d68e1]: random
0.4659510825295

#+srcname: caller
#+begin_src emacs-lisp :var x=random :cache yes
x
#+end_src

#+results[bec9c8724e397d5df3b696502df3ed7892fc4f5f]: caller
0.254227238707244
```

14.8.2.13 :sep

The `:sep` header argument can be used to control the delimiter used when writing tabular results out to files external to Org-mode. This is used either when opening tabular results of a code block by calling the `org-open-at-point` function bound to `C-c C-o` on the code block, or when writing code block results to an external file (see Section 14.8.2.3 [file], page 164) header argument.

By default, when `:sep` is not specified output tables are tab delimited.

14.8.2.14 :hlines

Tables are frequently represented with one or more horizontal lines, or hlines. The `:hlines` argument to a code block accepts the values `yes` or `no`, with a default value of `no`.

- `no` Strips horizontal lines from the input table. In most languages this is the desired effect because an `hline` symbol is interpreted as an unbound variable and raises an error. Setting `:hlines no` or relying on the default value yields the following results.

```
#+tblname: many-cols
| a | b | c |
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |

#+source: echo-table
#+begin_src python :var tab=many-cols
  return tab
#+end_src

#+results: echo-table
| a | b | c |
| d | e | f |
| g | h | i |
```

- `yes` Leaves hlines in the table. Setting `:hlines yes` has this effect.

```
#+tblname: many-cols
| a | b | c |
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |

#+source: echo-table
#+begin_src python :var tab=many-cols :hlines yes
  return tab
#+end_src

#+results: echo-table
| a | b | c |
```

```
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |
```

14.8.2.15 :colnames

The `:colnames` header argument accepts the values `yes`, `no`, or `nil` for unassigned. The default value is `nil`.

- `nil` If an input table looks like it has column names (because its second row is an hline), then the column names will be removed from the table before processing, then reapplied to the results.

```
#+tblname: less-cols
| a |
|---|
| b |
| c |

#+srcname: echo-table-again
#+begin_src python :var tab=less-cols
  return [[val + '*' for val in row] for row in tab]
#+end_src

#+results: echo-table-again
| a  |
|----|
| b* |
| c* |
```

  Please note that column names are not removed before the table is indexed using variable indexing See Section 14.8.2.1 [var], page 159.

- `no` No column name pre-processing takes place
- `yes` Column names are removed and reapplied as with `nil` even if the table does not "look like" it has column names (i.e. the second row is not an hline)

14.8.2.16 :rownames

The `:rownames` header argument can take on the values `yes` or `no`, with a default value of `no`.

- `no` No row name pre-processing will take place.
- `yes` The first column of the table is removed from the table before processing, and is then reapplied to the results.

```
#+tblname: with-rownames
| one | 1 | 2 | 3 | 4 |  5 |
| two | 6 | 7 | 8 | 9 | 10 |

#+srcname: echo-table-once-again
#+begin_src python :var tab=with-rownames :rownames yes
```

```
  return [[val + 10 for val in row] for row in tab]
#+end_src

#+results: echo-table-once-again
| one | 11 | 12 | 13 | 14 | 15 |
| two | 16 | 17 | 18 | 19 | 20 |
```

Please note that row names are not removed before the table is indexed using variable indexing See Section 14.8.2.1 [var], page 159.

14.8.2.17 `:shebang`

Setting the `:shebang` header argument to a string value (e.g. `:shebang "#!/bin/bash"`) causes the string to be inserted as the first line of any tangled file holding the code block, and the file permissions of the tangled file are set to make it executable.

14.8.2.18 `:eval`

The `:eval` header argument can be used to limit the evaluation of specific code blocks. `:eval` accepts two arguments "never" and "query". `:eval never` will ensure that a code block is never evaluated, this can be useful for protecting against the evaluation of dangerous code blocks. `:eval query` will require a query for every execution of a code block regardless of the value of the `org-confirm-babel-evaluate` variable.

14.9 Results of evaluation

The way in which results are handled depends on whether a session is invoked, as well as on whether `:results value` or `:results output` is used. The following table shows the table possibilities. For a full listing of the possible results header arguments see Section 14.8.2.2 [results], page 162.

	Non-session	**Session**
`:results value`	value of last expression	value of last expression
`:results output`	contents of STDOUT	concatenation of interpreter output

Note: With `:results value`, the result in both `:session` and non-session is returned to Org-mode as a table (a one- or two-dimensional vector of strings or numbers) when appropriate.

14.9.1 Non-session

14.9.1.1 `:results value`

This is the default. Internally, the value is obtained by wrapping the code in a function definition in the external language, and evaluating that function. Therefore, code should be written as if it were the body of such a function. In particular, note that Python does not automatically return a value from a function unless a `return` statement is present, and so a '`return`' statement will usually be required in Python.

This is the only one of the four evaluation contexts in which the code is automatically wrapped in a function definition.

14.9.1.2 `:results output`

The code is passed to the interpreter as an external process, and the contents of the standard output stream are returned as text. (In certain languages this also contains the error output stream; this is an area for future work.)

14.9.2 Session

14.9.2.1 `:results value`

The code is passed to the interpreter running as an interactive Emacs inferior process. The result returned is the result of the last evaluation performed by the interpreter. (This is obtained in a language-specific manner: the value of the variable `_` in Python and Ruby, and the value of `.Last.value` in R).

14.9.2.2 `:results output`

The code is passed to the interpreter running as an interactive Emacs inferior process. The result returned is the concatenation of the sequence of (text) output from the interactive interpreter. Notice that this is not necessarily the same as what would be sent to `STDOUT` if the same code were passed to a non-interactive interpreter running as an external process. For example, compare the following two blocks:

```
#+begin_src python :results output
 print "hello"
 2
 print "bye"
#+end_src

#+resname:
: hello
: bye
```

In non-session mode, the '2' is not printed and does not appear.

```
#+begin_src python :results output :session
 print "hello"
 2
 print "bye"
#+end_src

#+resname:
: hello
: 2
: bye
```

But in `:session` mode, the interactive interpreter receives input '2' and prints out its value, '2'. (Indeed, the other print statements are unnecessary here).

14.10 Noweb reference syntax

The "noweb" (see `http://www.cs.tufts.edu/~nr/noweb/`) Literate Programming system allows named blocks of code to be referenced by using the familiar Noweb syntax:

```
<<code-block-name>>
```

When a code block is tangled or evaluated, whether or not "noweb" references are expanded depends upon the value of the `:noweb` header argument. If `:noweb yes`, then a Noweb reference is expanded before evaluation. If `:noweb no`, the default, then the reference is not expanded before evaluation.

Note: the default value, `:noweb no`, was chosen to ensure that correct code is not broken in a language, such as Ruby, where `<<arg>>` is a syntactically valid construct. If `<<arg>>` is not syntactically valid in languages that you use, then please consider setting the default value.

14.11 Key bindings and useful functions

Many common Org-mode key sequences are re-bound depending on the context.

Within a code block, the following key bindings are active:

```
C-c C-c               org-babel-execute-src-block
C-c C-o               org-babel-open-src-block-result
C-up                  org-babel-load-in-session
M-down                org-babel-pop-to-session
```

In an Org-mode buffer, the following key bindings are active:

```
C-c C-v a  or  C-c C-v C-a          org-babel-sha1-hash
C-c C-v b  or  C-c C-v C-b          org-babel-execute-buffer
C-c C-v f  or  C-c C-v C-f          org-babel-tangle-file
C-c C-v g                           org-babel-goto-named-source-block
C-c C-v h                           org-babel-describe-bindings
C-c C-v l  or  C-c C-v C-l          org-babel-lob-ingest
C-c C-v p  or  C-c C-v C-p          org-babel-expand-src-block
C-c C-v s  or  C-c C-v C-s          org-babel-execute-subtree
C-c C-v t  or  C-c C-v C-t          org-babel-tangle
C-c C-v z  or  C-c C-v C-z          org-babel-switch-to-session
```

14.12 バッチ処理

It is possible to call functions from the command line. This shell script calls `org-babel-tangle` on every one of its arguments.

Be sure to adjust the paths to fit your system.

```
#!/bin/sh
# -*- mode: shell-script -*-
#
# tangle files with org-mode
#
DIR=`pwd`
FILES=""
ORGINSTALL="~/src/org/lisp/org-install.el"
```

```
# wrap each argument in the code required to call tangle on it
for i in $@; do
    FILES="$FILES \"$i\""
done

emacs -Q --batch -l $ORGINSTALL \
--eval "(progn
(add-to-list 'load-path (expand-file-name \"~/src/org/lisp/\"))
(add-to-list 'load-path (expand-file-name \"~/src/org/contrib/lisp/\"))
(require 'org)(require 'org-exp)(require 'ob)(require 'ob-tangle)
(mapc (lambda (file)
       (find-file (expand-file-name file \"$DIR\"))
       (org-babel-tangle)
       (kill-buffer)) '($FILES)))" 2>&1 |grep tangled
```

15 Miscellaneous

15.1 Completion

Emacs は補完無しでは Emacs とはいえません，そして org-mode はそれが意味をなすたびに使用します．もしあなたが *iswitchb-* か *ido-*のようなインタフェースを補完のプロンプトとして好むのであれば，あなたは `org-completion-use-iswitchb`や `org-completion-use-ido`変数のいずれかを設定することで指定することができます．

Org-mode はバッファ中の補完をサポートします．この種類の補完はミニバッファを活用します．あなたは簡単に数文字をバッファに入力し，補完キーを補完するテキストの右側で押します．

M-TAB ポイント位置での補完

- 見出しの先頭では，TODO キーワードを補完します．
- '\\'の後では，エクスポート機能によりサポートされる TEX のシンボルを補完します．
- '`*`'の後では，'`[[*find this headline]]`'のようにリンクを検索できるように，カレントバッファで見出しを補完します．
- 見出し中の '`:`'の後では，タグを補完します．タグのリストは `org-tag-alist`変数 (see Section 6.2 [Setting tags], page 53 より，もしかすると，バッファ中の '`#+TAGS`'オプションでも設定されているかもしれません．) から与えられるか，カレントバッファで使われている全てのタグから動的に生成されます．
- 見出しの外にある '`:`'の後では，プロパティキーを補完します．キーのリストは現在のバッファで使われている全てのキーから動的に構築されます．
- '`[`'の後では，リンクの省略記法を補完します (see Section 4.6 [Link abbreviations], page 38)．
- '`+`'の後では，Org-mode 向けのファイル固有の設定としてセットする '`TYP_TODO`' や '`OPTIONS`'のようなスペシャルキーワードを補完します．オプションキーワードが既に補完されているなら，*M-TAB*を再び押すことでこのキーワードの設定の例を挿入します．
- '`#+STARTUP:` 'の後の行の中では，STARTUP キーワードを補完します，すなわち，はこの行では正しいキーです．
- 他の場所では，Ispell を用いた辞書補完が行われます．

15.2 Easy Templates

Org-mode は僅かなキーストロークのみによる空の構造の (`#+BEGIN_SRC`や`#+END_SRC`のような) 要素の挿入をサポートします．これはネイティブなテンプレート拡張機構を通じて得られるものです．ここで留意すべきこととして，Emacs は例えば '`yasnippet`'のような同じように使うことができるいくつかの他のテンプレート機構を持ちます．

構造要素の挿入には，'`<`'をタイプし，続いてテンプレートセレクタと `TAB`をタイプします．補完は上記のキーストロークが単独で行に入力されている場合のみ働きます．

以下のテンプレートセレクタが現在サポートされています．

```
s       #+begin_src ... #+end_src
e       #+begin_example ... #+end_example
```

```
q         #+begin_quote ... #+end_quote
v         #+begin_verse ... #+end_verse
c         #+begin_center ... #+end_center
l         #+begin_latex ... #+end_latex
L         #+latex:
h         #+begin_html ... #+end_html
H         #+html:
a         #+begin_ascii ... #+end_ascii
A         #+ascii:
i         #+include: line
```

例えば，空の行で\"<e\"と入力し，その後 TAB を入力すると，EXAMPLE テンプレートが補完されます．

あなたは `org-structure-template-list`変数をカスタマイズすることで追加のテンプレートをインストールすることができます．詳細は変数の docstring を参照してください．

15.3 Speed keys

Single keys can be made to execute commands when the cursor is at the beginning of a headline, i.e. before the first star. Configure the variable `org-use-speed-commands` to activate this feature. There is a pre-defined list of commands, and you can add more such commands using the variable `org-speed-commands-user`. Speed keys do not only speed up navigation and other commands, but they also provide an alternative way to execute commands bound to keys that are not or not easily available on a TTY, or on a small mobile device with a limited keyboard.

To see which commands are available, activate the feature and press `?` with the cursor at the beginning of a headline.

15.4 Code evaluation and security issues

Org provides tools to work with the code snippets, including evaluating them.

Running code on your machine always comes with a security risk. Badly written or malicious code can be executed on purpose or by accident. Org has default settings which will only evaluate such code if you give explicit permission to do so, and as a casual user of these features you should leave these precautions intact.

For people who regularly work with such code, the confirmation prompts can become annoying, and you might want to turn them off. This can be done, but you must be aware of the risks that are involved.

Code evaluation can happen under the following circumstances:

Source code blocks

Source code blocks can be evaluated during export, or when pressing `C-c C-c` in the block. The most important thing to realize here is that Org mode files which contain code snippets are, in a certain sense, like executable files. So you should accept them and load them into Emacs only from trusted sources—just like you would do with a program you install on your computer.

Make sure you know what you are doing before customizing the variables which take off the default security brakes.

`org-confirm-babel-evaluate` [User Option]
: When t (the default), the user is asked before every code block evaluation. When nil, the user is not asked. When set to a function, it is called with two arguments (language and body of the code block) and should return t to ask and nil not to ask.

For example, here is how to execute "ditaa" code (which is considered safe) without asking:

```
(defun my-org-confirm-babel-evaluate (lang body)
  (not (string= lang "ditaa")))  ; don't ask for ditaa
(setq org-confirm-babel-evaluate 'my-org-confirm-babel-evaluate)
```

Following `shell` *and* `elisp` *links*
: Org has two link types that can directly evaluate code (see Section 4.3 [External links], page 34). These links can be problematic because the code to be evaluated is not visible.

`org-confirm-shell-link-function` [User Option]
: Function to queries user about shell link execution.

`org-confirm-elisp-link-function` [User Option]
: Functions to query user for Emacs Lisp link execution.

Formulas in tables
: Formulas in tables (see Section 3.5 [The spreadsheet], page 23) are code that is evaluated either by the *calc* interpreter, or by the *Emacs Lisp* interpreter.

15.5 Customization

There are more than 180 variables that can be used to customize Org. For the sake of compactness of the manual, I am not describing the variables here. A structured overview of customization variables is available with *M-x org-customize*. Or select `Browse Org Group` from the `Org->Customization` menu. Many settings can also be activated on a per-file basis, by putting special lines into the buffer (see Section 15.6 [In-buffer settings], page 176).

15.6 Summary of in-buffer settings

Org-mode uses special lines in the buffer to define settings on a per-file basis. These lines start with a '`#+`' followed by a keyword, a colon, and then individual words defining a setting. Several setting words can be in the same line, but you can also have multiple lines for the keyword. While these settings are described throughout the manual, here is a summary. After changing any of those lines in the buffer, press *C-c C-c* with the cursor still in the line to activate the changes immediately. Otherwise they become effective only when the file is visited again in a new Emacs session.

#+ARCHIVE: %s_done::
: This line sets the archive location for the agenda file. It applies for all subsequent lines until the next '`#+ARCHIVE`' line, or the end of the file. The first such line also applies to any entries before it. The corresponding variable is `org-archive-location`.

#+CATEGORY:
: This line sets the category for the agenda file. The category applies for all subsequent lines until the next '`#+CATEGORY`' line, or the end of the file. The first such line also applies to any entries before it.

#+COLUMNS: %25ITEM .....
: Set the default format for columns view. This format applies when columns view is invoked in locations where no `COLUMNS` property applies.

#+CONSTANTS: name1=value1 ...
: Set file-local values for constants to be used in table formulas. This line sets the local variable `org-table-formula-constants-local`. The global version of this variable is `org-table-formula-constants`.

#+FILETAGS: :tag1:tag2:tag3:
: Set tags that can be inherited by any entry in the file, including the top-level entries.

#+DRAWERS: NAME1 .....
: Set the file-local set of drawers. The corresponding global variable is `org-drawers`.

#+LINK: linkword replace
: These lines (several are allowed) specify link abbreviations. See Section 4.6 [Link abbreviations], page 38. The corresponding variable is `org-link-abbrev-alist`.

#+PRIORITIES: highest lowest default
: This line sets the limits and the default for the priorities. All three must be either letters A-Z or numbers 0-9. The highest priority must have a lower ASCII number than the lowest priority.

#+PROPERTY: Property_Name Value
: This line sets a default inheritance value for entries in the current buffer, most useful for specifying the allowed values of a property.

#+SETUPFILE: file
: This line defines a file that holds more in-buffer setup. Normally this is entirely ignored. Only when the buffer is parsed for option-setting lines (i.e. when starting Org-mode for a file, when pressing *C-c C-c* in a settings line, or when exporting), then the contents of this file are parsed as if they had been included in the buffer. In particular, the file can be any other Org-mode file with internal setup. You can visit the file the cursor is in the line with *C-c '*.

#+STARTUP:
: This line sets options to be used at startup of Org-mode, when an Org file is being visited.

  The first set of options deals with the initial visibility of the outline tree. The corresponding variable for global default settings is `org-startup-folded`, with a default value `t`, which means `overview`.

  `overview` top-level headlines only

`content`	all headlines
`showall`	no folding of any entries
`showeverything`	show even drawer contents

Dynamic virtual indentation is controlled by the variable `org-startup-indented`[1]

`indent`	start with `org-indent-mode` turned on
`noindent`	start with `org-indent-mode` turned off

Then there are options for aligning tables upon visiting a file. This is useful in files containing narrowed table columns. The corresponding variable is `org-startup-align-all-tables`, with a default value `nil`.

`align`	align all tables
`noalign`	donflt align tables on startup

When visiting a file, inline images can be automatically displayed. The corresponding variable is `org-startup-with-inline-images`, with a default value `nil` to avoid delays when visiting a file.

`inlineimages`	show inline images
`noinlineimages`	donflt show inline images on startup

Logging the closing and reopening of TODO items and clock intervals can be configured using these options (see variables `org-log-done`, `org-log-note-clock-out` and `org-log-repeat`)

`logdone`	record a timestamp when an item is marked DONE
`lognotedone`	record timestamp and a note when DONE
`nologdone`	donflt record when items are marked DONE
`logrepeat`	record a time when reinstating a repeating item
`lognoterepeat`	record a note when reinstating a repeating item
`nologrepeat`	do not record when reinstating repeating item
`lognoteclock-out`	record a note when clocking out
`nologlognoteclock-out`	donflt record a note when clocking out
`logreschedule`	record a timestamp when scheduling time changes
`lognotereschedule`	record a note when scheduling time changes
`nologreschedule`	do not record when a scheduling date changes
`logredeadline`	record a timestamp when deadline changes
`lognoteredeadline`	record a note when deadline changes
`nologredeadline`	do not record when a deadline date changes
`logrefile`	record a timestamp when refiling
`lognoterefile`	record a note when refiling
`nologrefile`	do not record when refiling

Here are the options for hiding leading stars in outline headings, and for indenting outlines. The corresponding variables are `org-hide-leading-stars` and `org-odd-levels-only`, both with a default setting `nil` (meaning `showstars` and `oddeven`).

`hidestars`	make all but one of the stars starting a headline invisible.
`showstars`	show all stars starting a headline

[1] Emacs 23 and Org-mode 6.29 are required

`indent` virtual indentation according to outline level
`noindent` no virtual indentation according to outline level
`odd` allow only odd outline levels (1,3,...)
`oddeven` allow all outline levels

To turn on custom format overlays over timestamps (variables `org-put-time-stamp-overlays` and `org-time-stamp-overlay-formats`), use

`customtime` overlay custom time format

The following options influence the table spreadsheet (variable `constants-unit-system`).

`constcgs` '`constants.el`' should use the c-g-s unit system
`constSI` '`constants.el`' should use the SI unit system

To influence footnote settings, use the following keywords. The corresponding variables are `org-footnote-define-inline`, `org-footnote-auto-label`, and `org-footnote-auto-adjust`.

`fninline` define footnotes inline
`fnnoinline` define footnotes in separate section
`fnlocal` define footnotes near first reference, but not inline
`fnprompt` prompt for footnote labels
`fnauto` create `[fn:1]`-like labels automatically (default)
`fnconfirm` offer automatic label for editing or confirmation
`fnplain` create `[1]`-like labels automatically
`fnadjust` automatically renumber and sort footnotes
`nofnadjust` do not renumber and sort automatically

To hide blocks on startup, use these keywords. The corresponding variable is `org-hide-block-startup`.

`hideblocks` Hide all begin/end blocks on startup
`nohideblocks` Do not hide blocks on startup

The display of entities as UTF-8 characters is governed by the variable `org-pretty-entities` and the keywords

`entitiespretty` Show entities as UTF-8 characters where possible
`entitiesplain` Leave entities plain

#+TAGS: TAG1(c1) TAG2(c2)
These lines (several such lines are allowed) specify the valid tags in this file, and (potentially) the corresponding *fast tag selection* keys. The corresponding variable is `org-tag-alist`.

#+TBLFM: This line contains the formulas for the table directly above the line.

#+TITLE:, #+AUTHOR:, #+EMAIL:, #+LANGUAGE:, #+TEXT:, #+DATE:,
#+OPTIONS:, #+BIND:, #+XSLT:,
#+DESCRIPTION:, #+KEYWORDS:,
#+LATEX_HEADER:, #+STYLE:, #+LINK_UP:, #+LINK_HOME:,
#+EXPORT_SELECT_TAGS:, #+EXPORT_EXCLUDE_TAGS:
These lines provide settings for exporting files. For more details see Section 12.2 [Export options], page 124.

#+TODO: #+SEQ_TODO: #+TYP_TODO:
: These lines set the TODO keywords and their interpretation in the current file. The corresponding variable is `org-todo-keywords`.

15.7 The very busy C-c C-c key

The key *C-c C-c* has many purposes in Org, which are all mentioned scattered throughout this manual. One specific function of this key is to add *tags* to a headline (see Chapter 6 [Tags], page 53). In many other circumstances it means something like *"Hey Org, look here and update according to what you see here"*. Here is a summary of what this means in different contexts.

- If there are highlights in the buffer from the creation of a sparse tree, or from clock display, remove these highlights.
- If the cursor is in one of the special `#+KEYWORD` lines, this triggers scanning the buffer for these lines and updating the information.
- If the cursor is inside a table, realign the table. This command works even if the automatic table editor has been turned off.
- If the cursor is on a `#+TBLFM` line, re-apply the formulas to the entire table.
- If the current buffer is a capture buffer, close the note and file it. With a prefix argument, file it, without further interaction, to the default location.
- If the cursor is on a `<<<target>>>`, update radio targets and corresponding links in this buffer.
- If the cursor is in a property line or at the start or end of a property drawer, offer property commands.
- If the cursor is at a footnote reference, go to the corresponding definition, and vice versa.
- If the cursor is on a statistics cookie, update it.
- If the cursor is in a plain list item with a checkbox, toggle the status of the checkbox.
- If the cursor is on a numbered item in a plain list, renumber the ordered list.
- If the cursor is on the `#+BEGIN` line of a dynamic block, the block is updated.

15.8 A cleaner outline view

Some people find it noisy and distracting that the Org headlines start with a potentially large number of stars, and that text below the headlines is not indented. While this is no problem when writing a *book-like* document where the outline headings are really section headings, in a more *list-oriented* outline, indented structure is a lot cleaner:

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

If you are using at least Emacs 23.2[2] and version 6.29 of Org, this kind of view can be achieved dynamically at display time using `org-indent-mode`. In this minor mode, all lines are prefixed for display with the necessary amount of space[3]. Also headlines are prefixed with additional stars, so that the amount of indentation shifts by two[4] spaces per level. All headline stars but the last one are made invisible using the `org-hide` face[5] - see below under '`2.`' for more information on how this works. You can turn on `org-indent-mode` for all files by customizing the variable `org-startup-indented`, or you can turn it on for individual files using

```
#+STARTUP: indent
```

If you want a similar effect in an earlier version of Emacs and/or Org, or if you want the indentation to be hard space characters so that the plain text file looks as similar as possible to the Emacs display, Org supports you in the following way:

1. *Indentation of text below headlines*
   You may indent text below each headline to make the left boundary line up with the headline, like

   ```
   *** 3rd level
       more text, now indented
   ```

   Org supports this with paragraph filling, line wrapping, and structure editing[6], preserving or adapting the indentation as appropriate.
2. *Hiding leading stars*
   You can modify the display in such a way that all leading stars become invisible. To do this in a global way, configure the variable `org-hide-leading-stars` or change this on a per-file basis with

   ```
   #+STARTUP: hidestars
   #+STARTUP: showstars
   ```

   With hidden stars, the tree becomes:

   ```
   * Top level headline
    * Second level
     * 3rd level
     ...
   ```

   The leading stars are not truly replaced by whitespace, they are only fontified with the face `org-hide` that uses the background color as font color. If you are not using either white or black background, you may have to customize this face to get the wanted effect. Another possibility is to set this font such that the extra stars are *almost* invisible, for example using the color `grey90` on a white background.
3. Things become cleaner still if you skip all the even levels and use only odd levels 1, 3, 5..., effectively adding two stars to go from one outline level to the next[7]. In this way

[2] Emacs 23.1 can actually crash with `org-indent-mode`

[3] `org-indent-mode` also sets the `wrap-prefix` property, such that `visual-line-mode` (or purely setting `word-wrap`) wraps long lines (including headlines) correctly indented.

[4] See the variable `org-indent-indentation-per-level`.

[5] Turning on `org-indent-mode` sets `org-hide-leading-stars` to `t` and `org-adapt-indentation` to `nil`.

[6] See also the variable `org-adapt-indentation`.

[7] When you need to specify a level for a property search or refile targets, '`LEVEL=2`' will correspond to 3 stars, etc.

we get the outline view shown at the beginning of this section. In order to make the structure editing and export commands handle this convention correctly, configure the variable `org-odd-levels-only`, or set this on a per-file basis with one of the following lines:

```
#+STARTUP: odd
#+STARTUP: oddeven
```

You can convert an Org file from single-star-per-level to the double-star-per-level convention with *M-x org-convert-to-odd-levels RET* in that file. The reverse operation is *M-x org-convert-to-oddeven-levels*.

15.9 Org-mode を tty 端末で使う

Because Org contains a large number of commands, by default many of Org's core commands are bound to keys that are generally not accessible on a tty, such as the cursor keys (`left`, `right`, `up`, `down`), `TAB` and `RET`, in particular when used together with modifiers like `Meta` and/or `Shift`. To access these commands on a tty when special keys are unavailable, the following alternative bindings can be used. The tty bindings below will likely be more cumbersome; you may find for some of the bindings below that a customized workaround suits you better. For example, changing a timestamp is really only fun with *S-cursor* keys, whereas on a tty you would rather use *C-c .* to re-insert the timestamp.

Default	Alternative 1	Speed key	Alternative 2
S-TAB	*C-u TAB*	*C*	
M-left	*C-c C-x l*	*l*	*Esc left*
M-S-left	*C-c C-x L*	*L*	
M-right	*C-c C-x r*	*r*	*Esc right*
M-S-right	*C-c C-x R*	*R*	
M-up	*C-c C-x u*		*Esc up*
M-S-up	*C-c C-x U*	*U*	
M-down	*C-c C-x d*		*Esc down*
M-S-down	*C-c C-x D*	*D*	
S-RET	*C-c C-x c*		
M-RET	*C-c C-x m*		*Esc RET*
M-S-RET	*C-c C-x M*		
S-left	*C-c left*		
S-right	*C-c right*		
S-up	*C-c up*		
S-down	*C-c down*		
C-S-left	*C-c C-x left*		
C-S-right	*C-c C-x right*		

15.10 他のパッケージとの関係

Org lives in the world of GNU Emacs and interacts in various ways with other code out there.

15.10.1 Packages that Org cooperates with

'`calc.el`' by Dave Gillespie
: Org uses the Calc package for implementing spreadsheet functionality in its tables (see Section 3.5 [The spreadsheet], page 23). Org checks for the availability of Calc by looking for the function `calc-eval` which will have been autoloaded during setup if Calc has been installed properly. As of Emacs 22, Calc is part of the Emacs distribution. Another possibility for interaction between the two packages is using Calc for embedded calculations. See Section "Embedded Mode" in *GNU Emacs Calc Manual*.

'`constants.el`' by Carsten Dominik
: In a table formula (see Section 3.5 [The spreadsheet], page 23), it is possible to use names for natural constants or units. Instead of defining your own constants in the variable `org-table-formula-constants`, install the '`constants`' package which defines a large number of constants and units, and lets you use unit prefixes like '`M`' for '`Mega`', etc. You will need version 2.0 of this package, available at `http://www.astro.uva.nl/~dominik/Tools`. Org checks for the function `constants-get`, which has to be autoloaded in your setup. See the installation instructions in the file '`constants.el`'.

'`cdlatex.el`' by Carsten Dominik
: Org-mode can make use of the CDLaTeX package to efficiently enter LaTeX fragments into Org files. See Section 11.7.5 [CDLaTeX mode], page 122.

'`imenu.el`' by Ake Stenhoff and Lars Lindberg
: Imenu allows menu access to an index of items in a file. Org-mode supports Imenu—all you need to do to get the index is the following:

```
(add-hook 'org-mode-hook
          (lambda () (imenu-add-to-menubar "Imenu")))
```

: By default the index is two levels deep—you can modify the depth using the option `org-imenu-depth`.

'`remember.el`' by John Wiegley
: Org used to use this package for capture, but no longer does.

'`speedbar.el`' by Eric M. Ludlam
: Speedbar is a package that creates a special frame displaying files and index items in files. Org-mode supports Speedbar and allows you to drill into Org files directly from the Speedbar. It also allows you to restrict the scope of agenda commands to a file or a subtree by using the command `<` in the Speedbar frame.

'`table.el`' by Takaaki Ota
: Complex ASCII tables with automatic line wrapping, column- and row-spanning, and alignment can be created using the Emacs table package by Takaaki Ota (`http://sourceforge.net/projects/table`, and also part of Emacs 22). Org-mode will recognize these tables and export them properly. Because of interference with other Org-mode functionality, you unfortunately cannot edit these tables directly in the buffer. Instead, you need to use the command `C-c '` to edit them, similar to source code snippets.

C-c ' `org-edit-special`
: Edit a '`table.el`' table. Works when the cursor is in a table.el table.

C-c ~ `org-table-create-with-table.el`
: Insert a '`table.el`' table. If there is already a table at point, this command converts it between the '`table.el`' format and the Org-mode format. See the documentation string of the command `org-convert-table` for the restrictions under which this is possible.

'`table.el`' is part of Emacs since Emacs 22.

'`footnote.el`' by Steven L. Baur
: Org-mode recognizes numerical footnotes as provided by this package. However, Org-mode also has its own footnote support (see Section 2.10 [Footnotes], page 15), which makes using '`footnote.el`' unnecessary.

15.10.2 Packages that lead to conflicts with Org-mode

In Emacs 23, `shift-selection-mode` is on by default, meaning that cursor motions combined with the shift key should start or enlarge regions. This conflicts with the use of *S-cursor* commands in Org to change timestamps, TODO keywords, priorities, and item bullet types if the cursor is at such a location. By default, *S-cursor* commands outside special contexts don't do anything, but you can customize the variable `org-support-shift-select`. Org-mode then tries to accommodate shift selection by (i) using it outside of the special contexts where special commands apply, and by (ii) extending an existing active region even if the cursor moves across a special context.

'`CUA.el`' by Kim. F. Storm
: Key bindings in Org conflict with the *S-<cursor>* keys used by CUA mode (as well as `pc-select-mode` and `s-region-mode`) to select and extend the region. In fact, Emacs 23 has this built-in in the form of `shift-selection-mode`, see previous paragraph. If you are using Emacs 23, you probably don't want to use another package for this purpose. However, if you prefer to leave these keys to a different package while working in Org-mode, configure the variable `org-replace-disputed-keys`. When set, Org will move the following key bindings in Org files, and in the agenda buffer (but not during date selection).

```
S-UP      ⇒  M-p             S-DOWN     ⇒  M-n
S-LEFT    ⇒  M--             S-RIGHT    ⇒  M-+
C-S-LEFT  ⇒  M-S--           C-S-RIGHT  ⇒  M-S-+
```

Yes, these are unfortunately more difficult to remember. If you want to have other replacement keys, look at the variable `org-disputed-keys`.

'`yasnippet.el`'
: The way Org mode binds the TAB key (binding to `[tab]` instead of `"\t"`) overrules YASnippet's access to this key. The following code fixed this problem:

```
(add-hook 'org-mode-hook
          (lambda ()
            (org-set-local 'yas/trigger-key [tab])
```

```
                    (define-key yas/keymap [tab] 'yas/next-field-group)))
```

The latest version of yasnippet doesn't play well with Org mode. If the above code does not fix the conflict, start by defining the following function:

```
(defun yas/org-very-safe-expand ()
       (let ((yas/fallback-behavior 'return-nil)) (yas/expand)))
```

Then, tell Org mode what to do with the new function:

```
(add-hook 'org-mode-hook
          (lambda ()
              (make-variable-buffer-local 'yas/trigger-key)
              (setq yas/trigger-key [tab])
              (add-to-list 'org-tab-first-hook 'yas/org-very-safe-expand)
              (define-key yas/keymap [tab] 'yas/next-field)))
```

'`windmove.el`' by Hovav Shacham

This package also uses the *S-<cursor>* keys, so everything written in the paragraph above about CUA mode also applies here. If you want make the windmove function active in locations where Org-mode does not have special functionality on *S-cursor*, add this to your configuration:

```
;; Make windmove work in org-mode:
(add-hook 'org-shiftup-final-hook 'windmove-up)
(add-hook 'org-shiftleft-final-hook 'windmove-left)
(add-hook 'org-shiftdown-final-hook 'windmove-down)
(add-hook 'org-shiftright-final-hook 'windmove-right)
```

'`viper.el`' by Michael Kifer

Viper uses *C-c /* and therefore makes this key not access the corresponding Org-mode command `org-sparse-tree`. You need to find another key for this command, or override the key in `viper-vi-global-user-map` with

```
(define-key viper-vi-global-user-map "C-c /" 'org-sparse-tree)
```

Appendix A Hacking

This appendix covers some aspects where users can extend the functionality of Org.

A.1 Hooks

Org has a large number of hook variables that can be used to add functionality. This appendix about hacking is going to illustrate the use of some of them. A complete list of all hooks with documentation is maintained by the Worg project and can be found at `http://orgmode.org/worg/org-configs/org-hooks.php`.

A.2 Add-on packages

A large number of add-on packages have been written by various authors. These packages are not part of Emacs, but they are distributed as contributed packages with the separate release available at the Org-mode home page at `http://orgmode.org`. The list of contributed packages, along with documentation about each package, is maintained by the Worg project at `http://orgmode.org/worg/org-contrib/`.

A.3 Adding hyperlink types

Org has a large number of hyperlink types built-in (see Chapter 4 [Hyperlinks], page 33). If you would like to add new link types, Org provides an interface for doing so. Let's look at an example file, '`org-man.el`', that will add support for creating links like '`[[man:printf][The printf manpage]]`' to show Unix manual pages inside Emacs:

```
;;; org-man.el - Support for links to manpages in Org

(require 'org)

(org-add-link-type "man" 'org-man-open)
(add-hook 'org-store-link-functions 'org-man-store-link)

(defcustom org-man-command 'man
  "The Emacs command to be used to display a man page."
  :group 'org-link
  :type '(choice (const man) (const woman)))

(defun org-man-open (path)
  "Visit the manpage on PATH.
PATH should be a topic that can be thrown at the man command."
  (funcall org-man-command path))

(defun org-man-store-link ()
  "Store a link to a manpage."
  (when (memq major-mode '(Man-mode woman-mode))
    ;; This is a man page, we do make this link
    (let* ((page (org-man-get-page-name))
           (link (concat "man:" page))
```

```
            (description (format "Manpage for %s" page)))
        (org-store-link-props
         :type "man"
         :link link
         :description description))))

(defun org-man-get-page-name ()
  "Extract the page name from the buffer name."
  ;; This works for both `Man-mode' and `woman-mode'.
  (if (string-match " \\(\\S-+\\)\\*" (buffer-name))
      (match-string 1 (buffer-name))
    (error "Cannot create link to this man page")))

(provide 'org-man)

;;; org-man.el ends here
```

You would activate this new link type in '`.emacs`' with

```
(require 'org-man)
```

Let's go through the file and see what it does.

1. It does `(require 'org)` to make sure that '`org.el`' has been loaded.
2. The next line calls `org-add-link-type` to define a new link type with prefix '`man`'. The call also contains the name of a function that will be called to follow such a link.
3. The next line adds a function to `org-store-link-functions`, in order to allow the command *`C-c l`* to record a useful link in a buffer displaying a man page.

The rest of the file defines the necessary variables and functions. First there is a customization variable that determines which Emacs command should be used to display man pages. There are two options, `man` and `woman`. Then the function to follow a link is defined. It gets the link path as an argument—in this case the link path is just a topic for the manual command. The function calls the value of `org-man-command` to display the man page.

Finally the function `org-man-store-link` is defined. When you try to store a link with *`C-c l`*, this function will be called to try to make a link. The function must first decide if it is supposed to create the link for this buffer type; we do this by checking the value of the variable `major-mode`. If not, the function must exit and return the value `nil`. If yes, the link is created by getting the manual topic from the buffer name and prefixing it with the string '`man:`'. Then it must call the command `org-store-link-props` and set the `:type` and `:link` properties. Optionally you can also set the `:description` property to provide a default for the link description when the link is later inserted into an Org buffer with *`C-c C-l`*.

When it makes sense for your new link type, you may also define a function `org-PREFIX-complete-link` that implements special (e.g. completion) support for inserting such a link with *`C-c C-l`*. Such a function should not accept any arguments, and return the full link with prefix.

A.4 Context-sensitive commands

Org has several commands that act differently depending on context. The most important example it the *C-c C-c* (see Section 15.7 [The very busy C-c C-c key], page 180). Also the *M-cursor* and *M-S-cursor* keys have this property.

Add-ons can tap into this functionality by providing a function that detects special context for that add-on and executes functionality appropriate for the context. Here is an example from Dan Davison's '`org-R.el`' which allows you to evaluate commands based on the '`R`' programming language[1]. For this package, special contexts are lines that start with `#+R:` or `#+RR:`.

```
(defun org-R-apply-maybe ()
  "Detect if this is context for org-R and execute R commands."
  (if (save-excursion
        (beginning-of-line 1)
        (looking-at "#\\+RR?:"))
      (progn (call-interactively 'org-R-apply)
             t) ;; to signal that we took action
    nil)) ;; to signal that we did not

(add-hook 'org-ctrl-c-ctrl-c-hook 'org-R-apply-maybe)
```

The function first checks if the cursor is in such a line. If that is the case, `org-R-apply` is called and the function returns `t` to signal that action was taken, and *C-c C-c* will stop looking for other contexts. If the function finds it should do nothing locally, it returns `nil` so that other, similar functions can have a try.

A.5 任意のシンタックスによる表やリスト

Since Orgtbl mode can be used as a minor mode in arbitrary buffers, a frequent feature request has been to make it work with native tables in specific languages, for example LaTeX. However, this is extremely hard to do in a general way, would lead to a customization nightmare, and would take away much of the simplicity of the Orgtbl-mode table editor.

This appendix describes a different approach. We keep the Orgtbl mode table in its native format (the *source table*), and use a custom function to *translate* the table to the correct syntax, and to *install* it in the right location (the *target table*). This puts the burden of writing conversion functions on the user, but it allows for a very flexible system.

Bastien added the ability to do the same with lists, in Orgstruct mode. You can use Org's facilities to edit and structure lists by turning `orgstruct-mode` on, then locally exporting such lists in another format (HTML, LaTeX or Texinfo.)

A.5.1 Radio tables

To define the location of the target table, you first need to create two lines that are comments in the current mode, but contain magic words for Orgtbl mode to find. Orgtbl mode will insert the translated table between these lines, replacing whatever was there before. For example:

[1] '`org-R.el`' has been replaced by the org-mode functionality described in Chapter 14 [Working With Source Code], page 152 and is now obsolete.

```
/* BEGIN RECEIVE ORGTBL table_name */
/* END RECEIVE ORGTBL table_name */
```

Just above the source table, we put a special line that tells Orgtbl mode how to translate this table and where to install it. For example:

```
#+ORGTBL: SEND table_name translation_function arguments....
```

`table_name` is the reference name for the table that is also used in the receiver lines. `translation_function` is the Lisp function that does the translation. Furthermore, the line can contain a list of arguments (alternating key and value) at the end. The arguments will be passed as a property list to the translation function for interpretation. A few standard parameters are already recognized and acted upon before the translation function is called:

`:skip N` Skip the first N lines of the table. Hlines do count as separate lines for this parameter!

`:skipcols (n1 n2 ...)`
List of columns that should be skipped. If the table has a column with calculation marks, that column is automatically discarded as well. Please note that the translator function sees the table *after* the removal of these columns, the function never knows that there have been additional columns.

The one problem remaining is how to keep the source table in the buffer without disturbing the normal workings of the file, for example during compilation of a C file or processing of a LaTeX file. There are a number of different solutions:

- The table could be placed in a block comment if that is supported by the language. For example, in C mode you could wrap the table between '`/*`' and '`*/`' lines.
- Sometimes it is possible to put the table after some kind of *END* statement, for example '`\bye`' in TeX and '`\end{document}`' in LaTeX.
- You can just comment the table line-by-line whenever you want to process the file, and uncomment it whenever you need to edit the table. This only sounds tedious—the command *`M-x orgtbl-toggle-comment`* makes this comment-toggling very easy, in particular if you bind it to a key.

A.5.2 A LaTeX example of radio tables

The best way to wrap the source table in LaTeX is to use the `comment` environment provided by '`comment.sty`'. It has to be activated by placing `\usepackage{comment}` into the document header. Orgtbl mode can insert a radio table skeleton[2] with the command *`M-x orgtbl-insert-radio-table`*. You will be prompted for a table name, let's say we use '`salesfigures`'. You will then get the following template:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| | |
```

[2] By default this works only for LaTeX, HTML, and Texinfo. Configure the variable `orgtbl-radio-tables` to install templates for other modes.

```
\end{comment}
```

The `#+ORGTBL: SEND` line tells Orgtbl mode to use the function `orgtbl-to-latex` to convert the table into LaTeX and to put it into the receiver location with name `salesfigures`. You may now fill in the table—feel free to use the spreadsheet features[3]:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| Month | Days | Nr sold | per day |
|-------+------+---------+---------|
| Jan   |   23 |      55 |     2.4 |
| Feb   |   21 |      16 |     0.8 |
| March |   22 |     278 |    12.6 |
#+TBLFM: $4=$3/$2;%.1f
% $ (optional extra dollar to keep font-lock happy, see footnote)
\end{comment}
```

When you are done, press *C-c C-c* in the table to get the converted table inserted between the two marker lines.

Now let's assume you want to make the table header by hand, because you want to control how columns are aligned, etc. In this case we make sure that the table translator skips the first 2 lines of the source table, and tell the command to work as a *splice*, i.e. to not produce header and footer commands of the target table:

```
\begin{tabular}{lrrr}
Month & \multicolumn{1}{c}{Days} & Nr.\ sold & per day\\
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\end{tabular}
%
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex :splice t :skip 2
| Month | Days | Nr sold | per day |
|-------+------+---------+---------|
| Jan   |   23 |      55 |     2.4 |
| Feb   |   21 |      16 |     0.8 |
| March |   22 |     278 |    12.6 |
#+TBLFM: $4=$3/$2;%.1f
\end{comment}
```

The LaTeX translator function `orgtbl-to-latex` is already part of Orgtbl mode. It uses a `tabular` environment to typeset the table and marks horizontal lines with `\hline`. Furthermore, it interprets the following parameters (see also see Section A.5.3 [Translator functions], page 191):

[3] If the '`#+TBLFM`' line contains an odd number of dollar characters, this may cause problems with font-lock in LaTeX mode. As shown in the example you can fix this by adding an extra line inside the `comment` environment that is used to balance the dollar expressions. If you are using AUCTeX with the font-latex library, a much better solution is to add the `comment` environment to the variable `LaTeX-verbatim-environments`.

`:splice nil/t`
: When set to t, return only table body lines, don't wrap them into a tabular environment. Default is nil.

`:fmt fmt`
: A format to be used to wrap each field, it should contain `%s` for the original field value. For example, to wrap each field value in dollars, you could use `:fmt "$%s$"`. This may also be a property list with column numbers and formats. for example `:fmt (2 "$%s$" 4 "%s\\%%")`. A function of one argument can be used in place of the strings; the function must return a formatted string.

`:efmt efmt`
: Use this format to print numbers with exponentials. The format should have `%s` twice for inserting mantissa and exponent, for example `"%s\\times10^{%s}"`. The default is `"%s\\,(%s)"`. This may also be a property list with column numbers and formats, for example `:efmt (2 "$%s\\times10^{%s}$" 4 "$%s\\cdot10^{%s}$")`. After `efmt` has been applied to a value, `fmt` will also be applied. Similar to `fmt`, functions of two arguments can be supplied instead of strings.

A.5.3 Translator functions

Orgtbl mode has several translator functions built-in: `orgtbl-to-csv` (comma-separated values), `orgtbl-to-tsv` (TAB-separated values) `orgtbl-to-latex`, `orgtbl-to-html`, and `orgtbl-to-texinfo`. Except for `orgtbl-to-html`[4], these all use a generic translator, `orgtbl-to-generic`. For example, `orgtbl-to-latex` itself is a very short function that computes the column definitions for the `tabular` environment, defines a few field and line separators and then hands processing over to the generic translator. Here is the entire code:

```
(defun orgtbl-to-latex (table params)
  "Convert the Orgtbl mode TABLE to LaTeX."
  (let* ((alignment (mapconcat (lambda (x) (if x "r" "l"))
                               org-table-last-alignment ""))
         (params2
          (list
           :tstart (concat "\\begin{tabular}{" alignment "}")
           :tend "\\end{tabular}"
           :lstart "" :lend " \\\\" :sep " & "
           :efmt "%s\\,(%s)" :hline "\\hline")))
    (orgtbl-to-generic table (org-combine-plists params2 params))))
```

As you can see, the properties passed into the function (variable *PARAMS*) are combined with the ones newly defined in the function (variable *PARAMS2*). The ones passed into the function (i.e. the ones set by the '`ORGTBL SEND`' line) take precedence. So if you would like to use the LaTeX translator, but wanted the line endings to be '`\\[2mm]`' instead of the default '`\\`', you could just overrule the default with

```
#+ORGTBL: SEND test orgtbl-to-latex :lend " \\\\[2mm]"
```

For a new language, you can either write your own converter function in analogy with the LaTeX translator, or you can use the generic function directly. For example, if you have

[4] The HTML translator uses the same code that produces tables during HTML export.

a language where a table is started with '`!BTBL!`', ended with '`!ETBL!`', and where table lines are started with '`!BL!`', ended with '`!EL!`', and where the field separator is a TAB, you could call the generic translator like this (on a single line!):

```
#+ORGTBL: SEND test orgtbl-to-generic :tstart "!BTBL!" :tend "!ETBL!"
                                      :lstart "!BL! " :lend " !EL!" :sep "\t"
```

Please check the documentation string of the function `orgtbl-to-generic` for a full list of parameters understood by that function, and remember that you can pass each of them into `orgtbl-to-latex`, `orgtbl-to-texinfo`, and any other function using the generic function.

Of course you can also write a completely new function doing complicated things the generic translator cannot do. A translator function takes two arguments. The first argument is the table, a list of lines, each line either the symbol `hline` or a list of fields. The second argument is the property list containing all parameters specified in the '`#+ORGTBL: SEND`' line. The function must return a single string containing the formatted table. If you write a generally useful translator, please post it on @`emacs-orgmodegnu.org` so that others can benefit from your work.

A.5.4 ラジオリスト

Sending and receiving radio lists works exactly the same way as sending and receiving radio tables (see Section A.5.1 [Radio tables], page 188). As for radio tables, you can insert radio list templates in HTML, LaTeX and Texinfo modes by calling `org-list-insert-radio-list`.

Here are the differences with radio tables:

- Orgstruct mode must be active.
- Use the `ORGLST` keyword instead of `ORGTBL`.
- The available translation functions for radio lists don't take parameters.
- *C-c C-c* will work when pressed on the first item of the list.

Here is a LaTeX example. Let's say that you have this in your LaTeX file:

```
% BEGIN RECEIVE ORGLST to-buy
% END RECEIVE ORGLST to-buy
\begin{comment}
#+ORGLST: SEND to-buy org-list-to-latex
- a new house
- a new computer
  + a new keyboard
  + a new mouse
- a new life
\end{comment}
```

Pressing 'C-c C-c' on `a new house` and will insert the converted LaTeX list between the two marker lines.

A.6 Dynamic blocks

Org documents can contain *dynamic blocks*. These are specially marked regions that are updated by some user-written function. A good example for such a block is the clock table inserted by the command *C-c C-x C-r* (see Section 8.4 [Clocking work time], page 72).

Dynamic blocks are enclosed by a BEGIN-END structure that assigns a name to the block and can also specify parameters for the function producing the content of the block.

```
#+BEGIN: myblock :parameter1 value1 :parameter2 value2 ...

#+END:
```

Dynamic blocks are updated with the following commands

C-c C-x C-u `org-dblock-update`
: Update dynamic block at point.

C-u C-c C-x C-u
: Update all dynamic blocks in the current file.

Updating a dynamic block means to remove all the text between BEGIN and END, parse the BEGIN line for parameters and then call the specific writer function for this block to insert the new content. If you want to use the original content in the writer function, you can use the extra parameter `:content`.

For a block with name `myblock`, the writer function is `org-dblock-write:myblock` with as only parameter a property list with the parameters given in the begin line. Here is a trivial example of a block that keeps track of when the block update function was last run:

```
#+BEGIN: block-update-time :format "on %m/%d/%Y at %H:%M"

#+END:
```

The corresponding block writer function could look like this:

```
(defun org-dblock-write:block-update-time (params)
   (let ((fmt (or (plist-get params :format) "%d. %m. %Y")))
     (insert "Last block update at: "
             (format-time-string fmt (current-time)))))
```

If you want to make sure that all dynamic blocks are always up-to-date, you could add the function `org-update-all-dblocks` to a hook, for example `before-save-hook`. `org-update-all-dblocks` is written in a way such that it does nothing in buffers that are not in `org-mode`.

You can narrow the current buffer to the current dynamic block (like any other block) with `org-narrow-to-block`.

A.7 Special agenda views

Org provides a special hook that can be used to narrow down the selection made by these agenda views: `todo`, `alltodo`, `tags`, `tags-todo`, `tags-tree`. You may specify a function that is used at each match to verify if the match should indeed be part of the agenda view, and if not, how much should be skipped. You can specify a global condition that will be applied to all agenda views, this condition would be stored in the variable `org-agenda-skip-function-global`. More commonly, such a definition is applied only to specific custom searches, using `org-agenda-skip-function`.

Let's say you want to produce a list of projects that contain a WAITING tag anywhere in the project tree. Let's further assume that you have marked all tree headings that define a project with the TODO keyword PROJECT. In this case you would run a TODO search

for the keyword PROJECT, but skip the match unless there is a WAITING tag anywhere in the subtree belonging to the project line.

To achieve this, you must write a function that searches the subtree for the tag. If the tag is found, the function must return `nil` to indicate that this match should not be skipped. If there is no such tag, return the location of the end of the subtree, to indicate that search should continue from there.

```
(defun my-skip-unless-waiting ()
  "Skip trees that are not waiting"
  (let ((subtree-end (save-excursion (org-end-of-subtree t))))
    (if (re-search-forward ":waiting:" subtree-end t)
        nil          ; tag found, do not skip
      subtree-end))) ; tag not found, continue after end of subtree
```

Now you may use this function in an agenda custom command, for example like this:

```
(org-add-agenda-custom-command
 '("b" todo "PROJECT"
   ((org-agenda-skip-function 'my-skip-unless-waiting)
    (org-agenda-overriding-header "Projects waiting for something: "))))
```

Note that this also binds `org-agenda-overriding-header` to get a meaningful header in the agenda view.

A general way to create custom searches is to base them on a search for entries with a certain level limit. If you want to study all entries with your custom search function, simply do a search for '`LEVEL>0`'[5], and then use `org-agenda-skip-function` to select the entries you really want to have.

You may also put a Lisp form into `org-agenda-skip-function`. In particular, you may use the functions `org-agenda-skip-entry-if` and `org-agenda-skip-subtree-if` in this form, for example:

`'(org-agenda-skip-entry-if 'scheduled)`
: Skip current entry if it has been scheduled.

`'(org-agenda-skip-entry-if 'notscheduled)`
: Skip current entry if it has not been scheduled.

`'(org-agenda-skip-entry-if 'deadline)`
: Skip current entry if it has a deadline.

`'(org-agenda-skip-entry-if 'scheduled 'deadline)`
: Skip current entry if it has a deadline, or if it is scheduled.

`'(org-agenda-skip-entry-if 'todo '("TODO" "WAITING"))`
: Skip current entry if the TODO keyword is TODO or WAITING.

`'(org-agenda-skip-entry-if 'todo 'done)`
: Skip current entry if the TODO keyword marks a DONE state.

`'(org-agenda-skip-entry-if 'timestamp)`
: Skip current entry if it has any timestamp, may also be deadline or scheduled.

[5] Note that, when using `org-odd-levels-only`, a level number corresponds to order in the hierarchy, not to the number of stars.

`'(org-agenda-skip-entry 'regexp "regular expression")`
: Skip current entry if the regular expression matches in the entry.

`'(org-agenda-skip-entry 'notregexp "regular expression")`
: Skip current entry unless the regular expression matches.

`'(org-agenda-skip-subtree-if 'regexp "regular expression")`
: Same as above, but check and skip the entire subtree.

Therefore we could also have written the search for WAITING projects like this, even without defining a special function:

```
(org-add-agenda-custom-command
 '("b" todo "PROJECT"
   ((org-agenda-skip-function '(org-agenda-skip-subtree-if
                                'regexp ":waiting:"))
    (org-agenda-overriding-header "Projects waiting for something: "))))
```

A.8 Extracting agenda information

Org provides commands to access agenda information for the command line in Emacs batch mode. This extracted information can be sent directly to a printer, or it can be read by a program that does further processing of the data. The first of these commands is the function `org-batch-agenda`, that produces an agenda view and sends it as ASCII text to STDOUT. The command takes a single string as parameter. If the string has length 1, it is used as a key to one of the commands you have configured in `org-agenda-custom-commands`, basically any key you can use after *C-c* `a`. For example, to directly print the current TODO list, you could use

```
emacs -batch -l ~/.emacs -eval '(org-batch-agenda "t")' | lpr
```

If the parameter is a string with 2 or more characters, it is used as a tags/TODO match string. For example, to print your local shopping list (all items with the tag '`shop`', but excluding the tag '`NewYork`'), you could use

```
emacs -batch -l ~/.emacs                                      \
      -eval '(org-batch-agenda "+shop-NewYork")' | lpr
```

You may also modify parameters on the fly like this:

```
emacs -batch -l ~/.emacs                                      \
   -eval '(org-batch-agenda "a"                               \
            org-agenda-span month                             \
            org-agenda-include-diary nil                      \
            org-agenda-files (quote ("~/org/project.org")))'  \
   | lpr
```

which will produce a 30-day agenda, fully restricted to the Org file '`~/org/projects.org`', not even including the diary.

If you want to process the agenda data in more sophisticated ways, you can use the command `org-batch-agenda-csv` to get a comma-separated list of values for each agenda item. Each line in the output will contain a number of fields separated by commas. The fields in a line are:

`category`	The category of the item
`head`	The headline, without TODO keyword, TAGS and PRIORITY
`type`	The type of the agenda entry, can be
	`todo` selected in TODO match
	`tagsmatch` selected in tags match
	`diary` imported from diary
	`deadline` a deadline
	`scheduled` scheduled
	`timestamp` appointment, selected by timestamp
	`closed` entry was closed on date
	`upcoming-deadline` warning about nearing deadline
	`past-scheduled` forwarded scheduled item
	`block` entry has date block including date
`todo`	The TODO keyword, if any
`tags`	All tags including inherited ones, separated by colons
`date`	The relevant date, like 2007-2-14
`time`	The time, like 15:00-16:50
`extra`	String with extra planning info
`priority-l`	The priority letter if any was given
`priority-n`	The computed numerical priority

Time and date will only be given if a timestamp (or deadline/scheduled) led to the selection of the item.

A CSV list like this is very easy to use in a post-processing script. For example, here is a Perl program that gets the TODO list from Emacs/Org and prints all the items, preceded by a checkbox:

```
#!/usr/bin/perl

# define the Emacs command to run
$cmd = "emacs -batch -l ~/.emacs -eval '(org-batch-agenda-csv \"t\")'";

# run it and capture the output
$agenda = qx{$cmd 2>/dev/null};

# loop over all lines
foreach $line (split(/\n/,$agenda)) {
  # get the individual values
  ($category,$head,$type,$todo,$tags,$date,$time,$extra,
   $priority_l,$priority_n) = split(/,/,$line);
  # process and print
  print "[ ] $head\n";
}
```

A.9 Using the property API

Here is a description of the functions that can be used to work with properties.

`org-entry-properties` **&optional** *pom which* [Function]
Get all properties of the entry at point-or-marker POM.
This includes the TODO keyword, the tags, time strings for deadline, scheduled, and clocking, and any additional properties defined in the entry. The return value is an alist. Keys may occur multiple times if the property key was used several times.
POM may also be nil, in which case the current entry is used. If WHICH is nil or 'all', get all properties. If WHICH is 'special' or 'standard', only get that subclass.

`org-entry-get` *pom property* **&optional** *inherit* [Function]
Get value of PROPERTY for entry at point-or-marker POM. By default, this only looks at properties defined locally in the entry. If INHERIT is non-nil and the entry does not have the property, then also check higher levels of the hierarchy. If INHERIT is the symbol `selective`, use inheritance if and only if the setting of `org-use-property-inheritance` selects PROPERTY for inheritance.

`org-entry-delete` *pom property* [Function]
Delete the property PROPERTY from entry at point-or-marker POM.

`org-entry-put` *pom property value* [Function]
Set PROPERTY to VALUE for entry at point-or-marker POM.

`org-buffer-property-keys` **&optional** *include-specials* [Function]
Get all property keys in the current buffer.

`org-insert-property-drawer` [Function]
Insert a property drawer at point.

`org-entry-put-multivalued-property` *pom property* **&rest** *values* [Function]
Set PROPERTY at point-or-marker POM to VALUES. VALUES should be a list of strings. They will be concatenated, with spaces as separators.

`org-entry-get-multivalued-property` *pom property* [Function]
Treat the value of the property PROPERTY as a whitespace-separated list of values and return the values as a list of strings.

`org-entry-add-to-multivalued-property` *pom property value* [Function]
Treat the value of the property PROPERTY as a whitespace-separated list of values and make sure that VALUE is in this list.

`org-entry-remove-from-multivalued-property` *pom property value* [Function]
Treat the value of the property PROPERTY as a whitespace-separated list of values and make sure that VALUE is *not* in this list.

`org-entry-member-in-multivalued-property` *pom property value* [Function]
Treat the value of the property PROPERTY as a whitespace-separated list of values and check if VALUE is in this list.

`org-property-allowed-value-functions` [User Option]
Hook for functions supplying allowed values for a specific property. The functions must take a single argument, the name of the property, and return a flat list of allowed values. If '`:ETC`' is one of the values, use the values as completion help, but allow also other values to be entered. The functions must return `nil` if they are not responsible for this property.

A.10 マッピング API を使う

Org has sophisticated mapping capabilities to find all entries satisfying certain criteria. Internally, this functionality is used to produce agenda views, but there is also an API that can be used to execute arbitrary functions for each or selected entries. The main entry point for this API is:

`org-map-entries` *func* **&optional** *match scope* **&rest** *skip* [Function]

Call FUNC at each headline selected by MATCH in SCOPE.

FUNC is a function or a Lisp form. The function will be called without arguments, with the cursor positioned at the beginning of the headline. The return values of all calls to the function will be collected and returned as a list.

The call to FUNC will be wrapped into a save-excursion form, so FUNC does not need to preserve point. After evaluation, the cursor will be moved to the end of the line (presumably of the headline of the processed entry) and search continues from there. Under some circumstances, this may not produce the wanted results. For example, if you have removed (e.g. archived) the current (sub)tree it could mean that the next entry will be skipped entirely. In such cases, you can specify the position from where search should continue by making FUNC set the variable 'org-map-continue-from' to the desired buffer position.

MATCH is a tags/property/todo match as it is used in the agenda match view. Only headlines that are matched by this query will be considered during the iteration. When MATCH is nil or t, all headlines will be visited by the iteration.

SCOPE determines the scope of this command. It can be any of:

`nil` the current buffer, respecting the restriction if any
`tree` the subtree started with the entry at point
`file` the current buffer, without restriction
`file-with-archives`
the current buffer, and any archives associated with it
`agenda` all agenda files
`agenda-with-archives`
all agenda files with any archive files associated with them
`(file1 file2 ...)`
if this is a list, all files in the list will be scanned

The remaining args are treated as settings for the skipping facilities of the scanner. The following items can be given here:

`archive` skip trees with the archive tag
`comment` skip trees with the COMMENT keyword
`function or Lisp form`
will be used as value for `org-agenda-skip-function`, so whenever the function returns t, FUNC will not be called for that entry and search will continue from the point where the function leaves it

The function given to that mapping routine can really do anything you like. It can use the property API (see Section A.9 [Using the property API], page 196) to gather more

information about the entry, or in order to change metadata in the entry. Here are a couple of functions that might be handy:

`org-todo` **&optional** *arg* [Function]
Change the TODO state of the entry. See the docstring of the functions for the many possible values for the argument ARG.

`org-priority` **&optional** *action* [Function]
Change the priority of the entry. See the docstring of this function for the possible values for ACTION.

`org-toggle-tag` *tag* **&optional** *onoff* [Function]
Toggle the tag TAG in the current entry. Setting ONOFF to either `on` or `off` will not toggle tag, but ensure that it is either on or off.

`org-promote` [Function]
Promote the current entry.

`org-demote` [Function]
Demote the current entry.

Here is a simple example that will turn all entries in the current file with a tag `TOMORROW` into TODO entries with the keyword `UPCOMING`. Entries in comment trees and in archive trees will be ignored.

```
(org-map-entries
   '(org-todo "UPCOMING")
   "+TOMORROW" 'file 'archive 'comment)
```

The following example counts the number of entries with TODO keyword `WAITING`, in all agenda files.

```
(length (org-map-entries t "/+WAITING" 'agenda))
```

Appendix B MobileOrg

MobileOrg（`http://mobileorg.ncogni.to/`）は Richard Moreland よって開発された *iPhone/iPod Touch* シリーズの携帯端末のためのアプリケーションです。*MobileOrg* は「リアル」のコンピュータ上にある Org-mode システムのために、オフラインのビューとキャプチャーによるサポートを提供します。その機能によって、実際のエントリーがどのように変化したかについて記録することができます。Android のユーザーは Matt Jones よって作成された MobileOrg Android (`http://wiki.github.com/matburt/mobileorg-android/`) のアプリをチェックしてください。

この付録では、*MobileOrg* で表示されるフォーマットの中でアジェンダビューを作成し、キャプチャーされたノートと *MobileOrg* で変更を、メインのシステムに統合していくために、Org-mode のサポートについて説明します。

MobileOrg の中でタグや TODO の状態を変更するためには、あなたは、例え、ひとつひとつのファイルが、一部しか使っていないとしても、全ての重要なタグや TODO キーワードを網羅するように `org-todo-keywords`と `org-tags-alist`変数のカスタマイズを設定しなければなりません。MobileOrg は、同様にインバッファの設定で状態やタグを提供しますが、これらの変数の中で設定されているものについてのみ、TODO の状態についての設定(see Section 5.2.5 [Per-file keywords], page 44) や相互に排他的な タグ (see Section 6.2 [Setting tags], page 53) についての装備状況を理解してください。

B.1 Setting up the staging area

MobileOrg はサーバー上のディレクトリを通して、Emacs と相互に連携させる必要があります。もしも公開のサーバーを使用しているなら、そのサーバーにアップロードされるファイルを暗号化したいと考えるかもしれません。この機能は Org-mode7.02 の *MobileOrg 1.5*（iPhone バージョン）で実現していますが、あなたのシステムに '`openssl`' をインストールしておく必要があるでしょう。暗号化するために、*MobileOrg* にパスワードを設定し、Emacs 上では、`org-mobile-use-encryption`[1] 変数を設定しておく必要があります。

無料の Dropbox.com（`http://dropbox.com`）のアカウント[2] を使い、ディレクトリを作成するのが最も簡単な方法です。MobileOrg で最初に Dropbox に接続したときに Dropbox の中に MobileOrg のディレクトリが作成されます。そのディレクトリが作成されたあと、次のように Emacs に書き込みます。

```
(setq org-mobile-directory "~/Dropbox/MobileOrg")
```

Org-mode はそのディレクトリの中に、*MobileOrg* 用のファイルを置いたり、そこからキャプチャーされたノートを読み込んだりするコマンドを持っています。

B.2 Pushing to MobileOrg

この操作では、`org-mobile-files`の中にリストアップされている全てのファイルを、`org-mobile-directory`で指定したディレクトリにコピーします。デフォルトではこのリストにはすべてのアジェン

[1] もしもあなたの Emacs の設定ファイルの中にパスワードを安全に保存したいならば、`org-mobile-encryption-password`変数を設定すると良いでしょう。その変数の説明文を読んでください。暗号化は、'`.org`' ファイルの内容のみに適用されることに注意してください。ファイルの名称そのものは、そのまま表示されます。

[2] もしも Dropbox を利用できない場合、または MobileOrg のバージョンがそれをサポートしていない場合には、webdav サーバが利用できます。詳しい情報を得るには、MobileOrg の説明部と FAQ entry (`http://orgmode.org/worg/org-faq.html#mobileorg_webdav`) をチェックしてください。

ダファイル（org-agenda-filesに登録されている）を含んでいます。しかしながら、org-mobiles-filesをカスタマイズすることでファイルを追加できます。ファイル名は、org-directoryとの相対パスで登録されるので、すべてのファイルがこのディレクトリの中に入ることになります。プッシュする操作で、ユーザー[3] によって定義されたすべてのカスタマイズされたアジェンダビューを持った‘agendas.org’という特別な Org-mode ファイルを作成します。最後に、Org-mode は全ての他のファイルへのリンクを含んだ‘index.org’というファイルを書き込みます。*MobileOrg* は、最初サーバーからこのファイルを読み込み、それから、そこに置かれているすべてのアジェンダファイルと Org-mode ファイルをダウンロードします。ダウンロードのスピードを上げるために、MobileOrg は、どのファイルのチェック記号[4] が変更されたかどうかを読み取るだけなのです。

B.3 MobileOrg から pull する

MobileOrg がサーバーと同期する際に、Org-mode のファイルを閲覧するために呼び出すだけではありません。それによってサーバー上の‘mobileorg.org’というファイルに対して、フラグがつけられたり、変更されたりしたエントリーに対して、キャプチャーされたエントリーやポインターを追加します。Org-mode では、この情報を InBox ファイルに統合し、フラッグがつけられたエントリーにポインタを使って操作するという *pull* の操作機能をもっています。どのように動作するのでしょうか。

1. Org-mode は、‘mobileorg.org’[5] の中で発見した全てのエントリーを移動し、org-mobile-inbox-for-pull変数によって、ポインターが付けられたファイルに追加します。記録されたエントリと編集されたイベントは、それぞれ InBox ファイル中でトップレベルのエントリーとして位置づけられるでしょう。
2. エントリーを移動したあと、Org-mode は、*MobileOrg* の中で作られた変更を実行することを試みます。いくつかの変更は直接、ユーザーの確認無しに適用されます。例では、タグ、TODO の状態、見出しそして本文に対するすべての変更がはっきりと適用されるというものです。将来の行動のために、フラグを付けられたエントリーは、:FLAGGED:というタグが付けられるでしょう。そのため、再び簡単に見つけることができるでしょう。あるエントリを探したり、変更を適用するさいに問題があれば、ポインターのついたエントリーは inbox に残され、エラーメッセージの印がつけられるでしょう。あなたはあとでこれらの案件を手動で解決する必要があります。
3. Org-mode では、その際にフラグがつけられたすべてのエントリーとともに、アジェンダビューを作成できます。そしてユーザーはそれらの項目をやり終えたり、必要な行動を実行するでしょう。*MobileOrg* のエントリーにフラグが付けられている間に、ノートが保存されていたら、そのノートは、カーソルがアジェンダの行の上に置かれた時に、エコーエリア上に表示されるでしょう。

   `?` そういう特別なアジェンダの中で、?が入力されたときには、別のウインドウでフラグの付けられたノートの全てが表示され、キルリング上に内容がコピーされます。そして、*? z C-y C-c C-c*を使用することで、フラグのつけられたノートを、そのエントリーの通常のノートとして保存することができます。?を２度続けて入力すると、（プロパティの中に保存されていた）記録されているフラグの付いたノートと一緒に、:FLAGGED:というタグを削除するよう指示したことになります。この方

[3] アジェンダを作成する際に、Org-mode ではすべての参照されるエントリーに ID 属性を強制的に付加します。そのため、これらのエントリーは、将来の行動のために、それらのエントリーに *MobileOrg* によってフラグを付けたとしても、ユニークなものとして識別されます。もしも、こんなにも沢山のエントリーにそういう属性値をつけたくない場合は、org-mobile-force-id-on-agenda-items変数を nilと設定してください。Org-mode は、各エントリーが十分ユニークであることを期待したうえで、アウトラインの階層構造に依存することになるでしょう。

[4] ‘checksums.dat’というファイルの中に自動的に保存されます。

[5] ‘mobileorg.org’はこの操作のあとで空になります。

法で、あなたはこのフラグの付けられたエントリーを意図したプロセスで完了させるという指示をすることになります。

もしも、すべてのフラグのついたエントリーを直接処理することができないならば、あなたは *C-c a ?*を入力して、アジェンダビュー[6] にいつでも戻ることができます。

[6] しかしながら、微妙な差があることに注意してください。*M-x org-mobile-pull RET*によって、自動的に作成されたビューは、最後に pull されて配置されたすべてのファイルを検索することを保証されています。これは、あなたのアジェンダファイルのリストに、現在含まれていないファイルも含みます。もしもあなたが、ビューを再作成するために、*C-c a ?*を最後に使用したならば、カレントのアジェンダファイルのみが検索されます。

Appendix C History and acknowledgments

Org was born in 2003, out of frustration over the user interface of the Emacs Outline mode. I was trying to organize my notes and projects, and using Emacs seemed to be the natural way to go. However, having to remember eleven different commands with two or three keys per command, only to hide and show parts of the outline tree, that seemed entirely unacceptable to me. Also, when using outlines to take notes, I constantly wanted to restructure the tree, organizing it parallel to my thoughts and plans. *Visibility cycling* and *structure editing* were originally implemented in the package '`outline-magic.el`', but quickly moved to the more general '`org.el`'. As this environment became comfortable for project planning, the next step was adding *TODO entries*, basic *timestamps*, and *table support*. These areas highlighted the two main goals that Org still has today: to be a new, outline-based, plain text mode with innovative and intuitive editing features, and to incorporate project planning functionality directly into a notes file.

Since the first release, literally thousands of emails to me or to `emacs-orgmode@gnu.org` have provided a constant stream of bug reports, feedback, new ideas, and sometimes patches and add-on code. Many thanks to everyone who has helped to improve this package. I am trying to keep here a list of the people who had significant influence in shaping one or more aspects of Org. The list may not be complete, if I have forgotten someone, please accept my apologies and let me know.

Before I get to this list, a few special mentions are in order:

Bastien Guerry
: Bastien has written a large number of extensions to Org (most of them integrated into the core by now), including the LaTeX exporter and the plain list parser. His support during the early days, when he basically acted as co-maintainer, was central to the success of this project. Bastien also invented Worg, helped establishing the Web presence of Org, and sponsors hosting costs for the orgmode.org website.

Eric Schulte and Dan Davison
: Eric and Dan are jointly responsible for the Org-babel system, which turns Org into a multi-language environment for evaluating code and doing literate programming and reproducible research.

John Wiegley
: John has contributed a number of great ideas and patches directly to Org, including the attachment system ('`org-attach.el`'), integration with Apple Mail ('`org-mac-message.el`'), hierarchical dependencies of TODO items, habit tracking ('`org-habits.el`'), and encryption ('`org-crypt.el`'). Also, the capture system is really an extended copy of his great '`remember.el`'.

Sebastian Rose
: Without Sebastian, the HTML/XHTML publishing of Org would be the pitiful work of an ignorant amateur. Sebastian has pushed this part of Org onto a much higher level. He also wrote '`org-info.js`', a Java script for displaying webpages derived from Org using an Info-like or a folding interface with single-key navigation.

OK, now to the full list of contributions! Again, please let me know what I am missing here!

- *Russel Adams* came up with the idea for drawers.
- *Thomas Baumann* wrote '`org-bbdb.el`' and '`org-mhe.el`'.
- *Christophe Bataillon* created the great unicorn logo that we use on the Org-mode website.
- *Alex Bochannek* provided a patch for rounding timestamps.
- *Jan B^^c3^^b6cker* wrote '`org-docview.el`'.
- *Brad Bozarth* showed how to pull RSS feed data into Org-mode files.
- *Tom Breton* wrote '`org-choose.el`'.
- *Charles Cave*'s suggestion sparked the implementation of templates for Remember, which are now templates for capture.
- *Pavel Chalmoviansky* influenced the agenda treatment of items with specified time.
- *Gregory Chernov* patched support for Lisp forms into table calculations and improved XEmacs compatibility, in particular by porting '`nouline.el`' to XEmacs.
- *Sacha Chua* suggested copying some linking code from Planner.
- *Baoqiu Cui* contributed the DocBook exporter.
- *Eddward DeVilla* proposed and tested checkbox statistics. He also came up with the idea of properties, and that there should be an API for them.
- *Nick Dokos* tracked down several nasty bugs.
- *Kees Dullemond* used to edit projects lists directly in HTML and so inspired some of the early development, including HTML export. He also asked for a way to narrow wide table columns.
- *Thomas S. Dye* contributed documentation on Worg and helped integrating the Org-Babel documentation into the manual.
- *Christian Egli* converted the documentation into Texinfo format, inspired the agenda, patched CSS formatting into the HTML exporter, and wrote '`org-taskjuggler.el`'.
- *David Emery* provided a patch for custom CSS support in exported HTML agendas.
- *Nic Ferrier* contributed mailcap and XOXO support.
- *Miguel A. Figueroa-Villanueva* implemented hierarchical checkboxes.
- *John Foerch* figured out how to make incremental search show context around a match in a hidden outline tree.
- *Raimar Finken* wrote '`org-git-line.el`'.
- *Mikael Fornius* works as a mailing list moderator.
- *Austin Frank* works as a mailing list moderator.
- *Eric Fraga* drove the development of BEAMER export with ideas and testing.
- *Barry Gidden* did proofreading the manual in preparation for the book publication through Network Theory Ltd.
- *Niels Giesen* had the idea to automatically archive DONE trees.
- *Nicolas Goaziou* rewrote much of the plain list code.
- *Kai Grossjohann* pointed out key-binding conflicts with other packages.

- *Brian Gough* of Network Theory Ltd publishes the Org mode manual as a book.
- *Bernt Hansen* has driven much of the support for auto-repeating tasks, task state change logging, and the clocktable. His clear explanations have been critical when we started to adopt the Git version control system.
- *Manuel Hermenegildo* has contributed various ideas, small fixes and patches.
- *Phil Jackson* wrote '`org-irc.el`'.
- *Scott Jaderholm* proposed footnotes, control over whitespace between folded entries, and column view for properties.
- *Matt Jones* wrote *MobileOrg Android*.
- *Tokuya Kameshima* wrote '`org-wl.el`' and '`org-mew.el`'.
- *Shidai Liu* ("Leo") asked for embedded LaTeX and tested it. He also provided frequent feedback and some patches.
- *Matt Lundin* has proposed last-row references for table formulas and named invisible anchors. He has also worked a lot on the FAQ.
- *David Maus* wrote '`org-atom.el`', maintains the issues file for Org, and is a prolific contributor on the mailing list with competent replies, small fixes and patches.
- *Jason F. McBrayer* suggested agenda export to CSV format.
- *Max Mikhanosha* came up with the idea of refiling.
- *Dmitri Minaev* sent a patch to set priority limits on a per-file basis.
- *Stefan Monnier* provided a patch to keep the Emacs-Lisp compiler happy.
- *Richard Moreland* wrote *MobileOrg* for the iPhone.
- *Rick Moynihan* proposed allowing multiple TODO sequences in a file and being able to quickly restrict the agenda to a subtree.
- *Todd Neal* provided patches for links to Info files and Elisp forms.
- *Greg Newman* refreshed the unicorn logo into its current form.
- *Tim O'Callaghan* suggested in-file links, search options for general file links, and TAGS.
- *Osamu Okano* wrote '`orgcard2ref.pl`', a Perl program to create a text version of the reference card.
- *Takeshi Okano* translated the manual and David O'Toole's tutorial into Japanese.
- *Oliver Oppitz* suggested multi-state TODO items.
- *Scott Otterson* sparked the introduction of descriptive text for links, among other things.
- *Pete Phillips* helped during the development of the TAGS feature, and provided frequent feedback.
- *Martin Pohlack* provided the code snippet to bundle character insertion into bundles of 20 for undo.
- *T.V. Raman* reported bugs and suggested improvements.
- *Matthias Rempe* (Oelde) provided ideas, Windows support, and quality control.
- *Paul Rivier* provided the basic implementation of named footnotes. He also acted as mailing list moderator for some time.
- *Kevin Rogers* contributed code to access VM files on remote hosts.

- *Frank Ruell* solved the mystery of the `keymapp nil` bug, a conflict with '`allout.el`'.
- *Jason Riedy* generalized the send-receive mechanism for Orgtbl tables with extensive patches.
- *Philip Rooke* created the Org reference card, provided lots of feedback, developed and applied standards to the Org documentation.
- *Christian Schlauer* proposed angular brackets around links, among other things.
- *Paul Sexton* wrote '`org-ctags.el`'.
- Linking to VM/BBDB/Gnus was first inspired by *Tom Shannon*'s '`organizer-mode.el`'.
- *Ilya Shlyakhter* proposed the Archive Sibling, line numbering in literal examples, and remote highlighting for referenced code lines.
- *Stathis Sideris* wrote the '`ditaa.jar`' ASCII to PNG converter that is now packaged into Org's '`contrib`' directory.
- *Daniel Sinder* came up with the idea of internal archiving by locking subtrees.
- *Dale Smith* proposed link abbreviations.
- *James TD Smith* has contributed a large number of patches for useful tweaks and features.
- *Adam Spiers* asked for global linking commands, inspired the link extension system, added support for mairix, and proposed the mapping API.
- *Ulf Stegemann* created the table to translate special symbols to HTML, LaTeX, UTF-8, Latin-1 and ASCII.
- *Andy Stewart* contributed code to '`org-w3m.el`', to copy HTML content with links transformation to Org syntax.
- *David O'Toole* wrote '`org-publish.el`' and drafted the manual chapter about publishing.
- *Sebastien Vauban* reported many issues with LaTeX and BEAMER export and enabled source code highlighling in Gnus.
- *Stefan Vollmar* organized a video-recorded talk at the Max-Planck-Institute for Neurology. He also inspired the creation of a concept index for HTML export.
- *Jürgen Vollmer* contributed code generating the table of contents in HTML output.
- *Samuel Wales* has provided important feedback and bug reports.
- *Chris Wallace* provided a patch implementing the '`QUOTE`' keyword.
- *David Wainberg* suggested archiving, and improvements to the linking system.
- *Carsten Wimmer* suggested some changes and helped fix a bug in linking to Gnus.
- *Roland Winkler* requested additional key bindings to make Org work on a tty.
- *Piotr Zielinski* wrote '`org-mouse.el`', proposed agenda blocks and contributed various ideas and code snippets.

Concept index

#

1

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Key index

S

T

U

V

W

X

Z

Command and function index

C

L

N

O

P

S

W

Variable index

This is not a complete index of variables and faces, only the ones that are mentioned in the manual. For a more complete list, use *M-x org-customize RET* and then click yourself through the tree.

C

H

L

O

P

U