Org Mode Manual

Release 3.15

by Carsten Dominik

This manual is for Org-mode (version 3.15).

Table of Contents

1 Introduction

1.1 Summary

Org-mode is a mode for keeping notes, maintaining ToDo lists, and doing project planning with a fast and effective plain-text system.

Org-mode develops organizational tasks around NOTES files that contain information about projects as plain text. Org-mode is implemented on top of outline-mode, which makes it possible to keep the content of large files well structured. Visibility cycling and structure editing help to work with the tree. Tables are easily created with a built-in table editor. Org-mode supports ToDo items, deadlines, time stamps, and scheduling. It dynamically compiles entries into an agenda that utilizes and smoothly integrates much of the Emacs calendar and diary. Plain text URL-like links connect to websites, emails, Usenet messages, BBDB entries, and any files related to the projects. For printing and sharing of notes, an Org-mode file can be exported as a structured ASCII file, or as HTML.

Org-mode keeps simple things simple. Not every outline branch needs to be an action item, not every action item needs to have priority or scheduling information associated with it. Org-mode can be used on different levels and in different ways, for example

- as an outline extension with visibility cycling and structure editing
- as an ASCII system and table editor to take structured notes
- as an ASCII table editor with some spreadsheet-like capabilities
- as a simple hypertext system, with HTML export
- as a TODO list editor
- as a full agenda and planner with deadlines and work scheduling

The Org-mode table editor can be integrated into any major mode by activating the minor Orgtbl-mode.

There is a website for Org-mode which provides links to the newest version of Org-mode, as well as additional information, screen shots and example files. This page is located at http://www.astro.uva.nl/~dominik/Tools/org/.

1.2 Installation and Activation

If Org-mode is part of the Emacs distribution or an XEmacs package, you only need to copy the following lines to your '`.emacs`' file. The last two lines define *global* keys for the commands `org-store-link` and `org-agenda` - please choose suitable keys yourself.

```
;; The following lines are always needed.  Choose your own keys.
(add-to-list 'auto-mode-alist '("\\.org$" . org-mode))
(define-key global-map "\C-cl" 'org-store-link)
(define-key global-map "\C-ca" 'org-agenda)
```

If you have downloaded Org-mode from the Web, you must byte-compile '`org.el`' and put it on your load path. In addition to the Emacs Lisp lines above, you also need to add the following lines to '`.emacs`':

```
;; These lines only if org-mode is not part of the X/Emacs distribution.
(autoload 'org-mode "org" "Org mode" t)
(autoload 'org-diary "org" "Diary entries from Org mode")
(autoload 'org-agenda "org" "Multi-file agenda from Org mode" t)
(autoload 'org-store-link "org" "Store a link to the current location" t)
(autoload 'orgtbl-mode "org" "Org tables as a minor mode" t)
(autoload 'turn-on-orgtbl "org" "Org tables as a minor mode")
```

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'`.

1.3 Feedback

If you find problems with Org-mode, or if you have questions, remarks, or ideas about it, please contact the maintainer Carsten Dominik at dominik@science.uva.nl.

For bug reports, please provide as much information as possible, including the version information of Emacs (*C-h v emacs-version* ⟨RET⟩) and Org-mode (*M-x org-version*), as well as the Org-mode related setup in '`.emacs`'. If an error occurs, a traceback can be very useful. 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?

Thanks for helping to improve this mode.

2 Document Structure

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

2.1 Outlines

Org-mode is implemented on top of outline-mode. Outlines allow to organize a document in a hierarchical structure, which (at least for me) is the best representation of notes and thoughts. Overview over 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-mode 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-mode start with one or more stars, for example

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

2.3 Visibility cycling

Outlines make it possible to hide parts of the text in the buffer. Org-mode uses a single command bound to the ⟨TAB⟩ key to change the visibility in the buffer.

⟨TAB⟩ Rotate current subtree between the states

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

At the beginning of the buffer (or when called with *C-u*), this does the same as the command *S-*⟨TAB⟩ below.

S-⟨TAB⟩ Rotate the entire buffer between the states

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

Note that inside tables, *S-*⟨TAB⟩ jumps to the previous field.

C-c C-a Show all.

When Emacs first visits an Org-mode 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: fold
#+STARTUP: nofold
#+STARTUP: content
```

2.4 Motion

The following commands jump to other headlines in the buffer.

`C-c C-n` Next heading.

`C-c C-p` Previous heading.

`C-c C-f` Next heading same level.

`C-c C-b` Previous heading same level.

`C-c C-u` Backward to higher level heading.

`C-c C-j` Jump to a different place without changing the current outline visibility. Shows the document structure in a temporary buffer, where you can use visibility cycling (⟨TAB⟩) to find your destination. After pressing ⟨RET⟩, the cursor moves to the selected location in the original buffer, and the headings hierarchy above it is made visible.

2.5 Structure editing

`M-`⟨RET⟩ Insert new heading with same level as current

`M-S-`⟨RET⟩ Insert new TODO entry with same level as current heading.

`M-`⟨left⟩ Promote current heading by one level

`M-`⟨right⟩ Demote current heading by one level

`M-S-`⟨left⟩ Promote the current subtree by one level

`M-S-`⟨right⟩ Demote the current subtree by one level

`M-S-`⟨up⟩ Move subtree up (swap with previous subtree of same level)

`M-S-`⟨down⟩ Move subtree down (swap with next subtree of same level)

`C-c C-h C-w`
Kill subtree, i.e. remove it from buffer but save in kill ring.

`C-c C-h M-w`
Copy subtree to kill ring.

`C-c C-h C-y`
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 prefix arg, or by yanking after a headline marker like '`****`'.

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 6), the Meta-Cursor keys have different functionality.

2.6 Archiving

When a project represented by a (sub)tree is finished, you may want to move the tree to an archive place, either in the same file under a special top-level heading, or even to a different file.

`C-c $` Archive the subtree starting at the cursor position to the location given by `org-archive-location`.

The default archive 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`. If you are also using the Org-mode agenda, archiving to a different file is a good way to keep archived trees from contributing agenda items.

2.7 Sparse trees

An important feature of Org-mode is the ability to construct *sparse trees* for selected information in an outline tree. A sparse tree means that the entire document is folded as much as possible, but the selected information is made visible along with the headline structure above it[1]. Just try it out and you will see immediately how it works.

Org-mode contains several commands creating such trees. The most basic one is `org-occur`:

`C-c /` 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.

Other commands are using this feature as well. For example `C-c C-v` creates a sparse TODO tree (see Section 5.1 [TODO basics], page 17).

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[2]. Or you can use the command `C-c C-x v` to copy the visible part of the document to another file (extension '`.txt`') which then can be printed in any desired way.

---

[1] See also the variable `org-show-following-heading`.

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

3 Tables

Org-mode has a very fast and intuitive table editor built-in. Spreadsheet-like calculations are supported in connection with the Emacs '`calc`' package.

3.1 The built-in table editor

Org-mode makes it easy to format tables in plain ASCII. Any line with '`|`' as the first non-white 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.

Creation and conversion

M-x org-table-create
: Creates an empty Org-mode table. However, it is much easier to just start typing, like *|Name|Phone|Age* ⟨RET⟩ *|-* ⟨TAB⟩

C-c C-c
: Convert region to table. Works when the cursor is not in an existing table, and when there is a region defined. If every line contains at least one TAB character, the function assumes that the material is tab separated. If not, lines are split at whitespace into fields. You can use a prefix argument to indicate how many consecutive spaces are at least required to indicate a field separator (default: just one).

Re-aligning and field motion

C-c C-c
: Re-align the table without moving the cursor.

⟨TAB⟩
: Re-align the table, move to the next field. Creates a new row if necessary.

S-⟨TAB⟩
: Re-align, move to previous field.

⟨RET⟩
: 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.

Column and row editing

M-⟨left⟩
M-⟨right⟩
: Move the current column left/right

`M-S-⟨left⟩` Kill the current column.

`M-S-⟨right⟩` Insert a new column to the left of the cursor position.

`M-⟨up⟩`
`M-⟨down⟩` Move the current row up/down

`M-S-⟨up⟩` Kill the current row or horizontal line.

`M-S-⟨down⟩` Insert a new row above (with arg: below) the current row.

`C-c -` Insert a horizontal line below current row. With prefix arg, the line is created above the current line.

Regions

`C-c C-h M-w`
Copy a rectangular region from a table to a special clipboard. Point and mark determine edge fields of the rectangle. The process ignores horizontal separator lines.

`C-c C-h C-w`
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-h C-y`
Paste a rectangular region into a table. The upper right 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.

`C-c C-q` Wrap several fields in a column like a paragraph. 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 prefix ARG may be used to change the number of desired lines. If there is no region, the current field is split at the cursor position and the text fragment to the right of the cursor is prepended to the field one line down. If there is no region, but you specify a prefix ARG, the current field gets blank, and the content is appended to the field above.

Calculations

`C-c =` Install a new formula for the current column and replace current field with the result of the formula.

`C-u C-c =` Install a new formula for the current field, which must be a named field. Evaluate the formula and replace the field content with the result.

`C-c '` Edit all formulas associated with the current table in a separate buffer.

`C-c *` Recalculate the current row by applying the stored formulas from left to right. When called with a `C-u` prefix, recalculate the entire table, starting with the first non-header line (i.e. below the first horizontal separator line). For details, see Section 3.2 [Table calculations], page 8.

`C-#` Rotate the calculation mark in first column through the states '', '`#`', '`*`', '`!`', '`$`'. For the meaning of these marks see Section 3.2.3 [Advanced features], page 10. When there is an active region, change all marks in the region.

`C-c ?` Which table column is the cursor in? Displays number `>0` in echo area.

`C-c +` 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⟩ 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. This key is also used by CUA-mode (see Section 9.4 [Interaction], page 31).

Miscellaneous

`C-c |` Toggle the visibility of vertical lines in tables. The lines are still there, only made invisible with a text property. Any '`|`' added by hand will become invisible on the next align.

`M-x org-table-import`
Import a file as a table. The table should be TAB- or whitespace separated. Useful for example to import an Excel 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.

`M-x org-table-export`
Export the table as a TAB-separated file. Useful for data exchange with for example Excel or database programs.

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

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

The only table command which then still works is `C-c C-c` to do a manual re-align.

3.2 Calculations in tables

The table editor has some spreadsheet-like capabilities. The Emacs '`calc`' package is required for this feature to work. There are basically two levels of complexity for table calculations in Org-mode. On the basic level, tables do only horizontal computations, so a field can be computed from other fields *in the same row*, and Org-mode assumes that there is only one formula for each column. This is very efficient to work with and enough for many tasks. On the complex level, columns and individual fields can be named for easier referencing in formulas, individual named fields can have their own formula associated with them, and recalculation can be automated.

3.2.1 Formula syntax

A formula can be any algebraic expression understood by the Emacs '`calc`' package. Before evaluation by `calc-eval` (see section "Calling calc from Your Lisp Programs" in *GNU Emacs Calc Manual*), variable substitution takes place:

`$`	refers to the current field
`$3`	refers to the field in column 3 of the current row
`$3..$7`	a vector of the fields in columns 3-7 of current row
`$P1..$P3`	vector of column range, using column names
`&2`	second data field above the current, in same column
`&5-2`	vector from fifth to second field above current
`&III-II`	vector of fields between 2nd and 3rd hline above
`&III`	vector of fields between third hline above and current field
`$name`	a named field, parameter or constant

The range vectors can be directly fed into the calc vector functions like functions '`vmean`' and '`vsum`'.

'`$name`' is interpreted as the name of a column, parameter or constant. Constants are defined globally through the variable `org-table-formula-constants`. If you have the '`constants.el`' package, it will also be used to resolve constants, including natural constants like '`$k`' for Planck's constant, units like '`$km`' for kilometers. Column names and parameters can be specified in special table lines. These are described below, see Section 3.2.3 [Advanced features], page 10.

A formula can contain an optional mode string after a semicolon. This string consists of flags to influence calc's modes[1] during execution, e.g. '`p20`' to switch the internal precision to 20 digits, '`n3`', '`s3`', '`e2`' or '`f4`' to switch to normal, scientific, engineering, or fix display format, respectively, and '`D`', '`R`', '`F`', and '`S`' to turn on degrees, radians, fraction and symbolic modes, respectively. In addition, you may provide a `printf` format specifier to reformat the final result. A few examples:

`$1+$2`	Sum of first and second field
`$1+$2;%.2f`	Same, format result to two decimals
`exp($2)+exp($1)`	Math functions can be used
`$;%.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
`vsum(&III)`	Sum numbers from 3rd hline above to here
`taylor($3,x=7,2)`	taylor series of $3, at x=7, second degree

[1] By default, Org-mode uses the standard calc modes (precision 12, angular units degrees, fraction and symbolic modes off). However, the display format which has been changed to `(float 5)` to keep tables compact. The default settings can be configured using the variable `org-calc-default-modes`.

3.2.2 Column formulas

To apply a formula to a field, type it directly into the field, 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-mode will remember the most recently used formula. The information is stored in a special line starting with '`#+TBLFM`' directly below the table. When adding/deleting/moving columns with the appropriate commands, the stored equations will be modified accordingly. When a column used in a calculation is removed, references to this column become invalid and will cause an error upon applying the equation.

Instead of typing an equation into the field, you may also use the command `C-c =`. It prompts for a formula (with default taken from the '`#+TBLFM:`' line) and applies it to the current field. A numerical prefix (e.g. `C-5 C-c =`) will apply it to that many subsequent fields in the current column.

To recompute all the fields in a line, use the command `C-c *`. It re-applies all stored equations to the current row, from left to right. With a `C-u` prefix, this will be done to every line in the table, so use this command it you want to make sure the entire table is up-to-date. `C-u C-c C-c` is another way to update the entire table. Global updating does not touch the line(s) above the first horizontal separator line, assuming that this is the table header.

3.2.3 Advanced features

If you want want the recalculation of fields to happen automatically, or if you want to be able to assign a formula to an individual field (instead of an entire column) you need to reserve the first column of the table for special marking characters. 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 |
| # | Sara    |      6 |     14 |     19 |    39 |  7.8 |
| # | 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);%.1f
```

Important: Please note that for these special tables, recalculating the table with `C-u C-c *` does only affect rows which are marked '`#`' or '`*`', and named fields. 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 define 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, named fields can have their own formula associated with them.

'`_`' 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. Changing a parameter and then recalculating the table can be useful.

'`#`' 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 exempted from recalculation with `C-u C-c *`. All lines that should be recalculated should be marked with '`#`' or '`*`'.

3.2.4 Named-field formulas

A named field can have its own formula associated with it. In the example above, this is used for the '`at`' field that contains the average result of the students. To enter a formula for a named field, just type it onto the buffer, preceded by '`:=`'. Or use `C-u C-c =`. This equation will be stored below the table like '`$name=...`'. Any recalculation in the table (even if only requested for the current line) will also update all named field formulas.

3.2.5 Editing and debugging formulas

To edit a column or field formula, you can use the commands `C-c =` and `C-u C-c =`, respectively. The currently active expression is then presented as default in the minibuffer, were it may be edited.

Note that making a table field blank does not remove the formula associated with the field - 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.

In particular for large tables with many formulas, it is convenient to use the command *C-c '* to edit the formulas of the current table in a separate buffer. That buffer will show the formulas one per line, and you are free to edit, add and remove formulas. Press *C-c ?* on a '`$...`' expression to get information about its interpretation. Exiting the buffer with *C-c C-c* only stores the modified formulas below the table. Exiting with *C-u C-c C-c* also applies them to the entire table. *C-c C-q* exits without installing the changes.

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 menu and repeat the calculation by pressing, for example by pressing *C-c =* ⟨RET⟩ in a field. Detailed information will be displayed.

3.2.6 Appetizer

Finally, just to wet your appetite on what can be done with the fantastic '`calc`' package, here is a table that computes the Taylor series for a couple of functions (homework: try that with Excel :-)

```
|---+-------------+---+-----+--------------------------------------|
|   | 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
```

3.3 The Orgtbl minor mode

If you like the intuitive way the Org-mode table editor works, you might want to use it also 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 mail mode, use

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

3.4 The '`table.el`' package

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). When ⟨TAB⟩ or *C-c C-c* is pressed in such a table, Org-mode will call `table-recognize-table` and move the cursor into the table. Inside a table, the keymap of Org-mode is inactive. In order to execute Org-mode-related commands, leave the table.

`C-c C-c` Recognize '`table.el`' table. Works when the cursor is in a table.el table.

`C-c ~` 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.

4 Hyperlinks

Just like HMTL, Org-mode provides links to other files, Usenet articles, emails and much more.

4.1 Links

Org-mode supports links to files, websites, Usenet and email messages; and BBDB database entries. Links are just plain-text URL-like locators, optionally enclosed by angular brackets. The following list shows examples for each link type.

`<http://www.astro.uva.nl/~dominik>`	on the web
`<file:/home/dominik/images/jupiter.jpg>`	file, absolute path
`<file:papers/last.pdf>`	file, relative path
`<file:~/code/main.c:255>`	file, with line number
`<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
`<rmail:folder>`	RMAIL folder link
`<rmail:folder#id>`	RMAIL message link
`<gnus:group>`	GNUS group link
`<gnus:group#id>`	GNUS article link
`<bbdb:Richard Stallman>`	BBDB link
`<shell:ls *.org>`[1]	A shell command

A link may contain space characters and is terminated by '`>`' or by the end of a line. In tables, the end of a table field also terminates a link. Angle brackets around a link are not required, but are recommended to avoid problems with punctuation and other text following the link. See also the variable `org-allow-space-in-links`.

`C-c l` Store a link to the current location. This is a *global* command which can be used in any buffer to create a link. The link will be stored for later insertion into an Org-mode buffer (see below). For VM, RMAIL, WANDERLUST, GNUS and BBDB buffers, the link will point to the current article/entry. For W3 and W3M buffer, the link goes to the current URL. For any other files, the link will just point to the file. The key binding `C-c l` is only a suggestion - see Section 1.2 [Installation and Activation], page 2.

`C-c C-l` Insert a link. This prompts for a link to be inserted into the buffer. You can just type a link, using one of the link type prefixes mentioned in the examples above. Through completion, all links stored during the current session can be accessed. When called with prefix arg, you can use file name completion to enter

[1] Note that '`<`' and '`>`' cannot be part of a link, and therefore of a shell command. If you need redirection, use `@{` and `@}` instead.

a file link. The link will be formatted as given in the variable `org-link-format` and inserted into the buffer. Note that you don't have to use this command to insert a link. Links in Org-mode are plain text, and you can type or paste them straight into the buffer.

C-c C-o Open link at point. This will launch a web browser for URLs (using `browse-url-at-point`), run vm/gnus/bbdb for the corresponding links, execute the command in a shell link, visit text files with Emacs and select a suitable application for non-text files. Classification of files is based on file extension only. See option `org-file-apps`. If there is no link at point, the current subtree will be searched for one. If you want to override the default application and visit the file with Emacs, use a *C-u* prefix. If the cursor is on a time stamp, compiles the agenda for that date.

IMPORTANT: Be careful not to use any dangerous commands in a shell link.

mouse-2 On links, *mouse-2* will open the link just like *C-c C-o* would.

mouse-3 Like *mouse-2*, but force file links to be opened with Emacs.

4.2 Remember

Another way to create org entries with links to other files is through the *Remember* package by John Wiegley. *Remember* lets you store quick notes with little interruption of your work flow. See http://www.emacswiki.org/cgi-bin/wiki/RememberMode for more information. The notes produced by *Remember* can be stored in different ways, and Org-mode files are a good target. Org-mode allows to file away notes either to a default file, or directly to the correct location in your Org-mode outline tree. The following customization[2] will tell *Remember* to use org files as target, and to create annotations compatible with Org-mode links.

```
(autoload 'org-remember-annotation "org")
(autoload 'org-remember-handler "org")
(setq org-directory "~/path/to/my/orgfiles/")
(setq org-default-notes-file "~/.notes")
(setq remember-annotation-functions '(org-remember-annotation))
(setq remember-handler-functions '(org-remember-handler))
```

When you compose a note with remember, you have to press *C-c C-c* to exit remember-mode and to file away the note. The handler first prompts for a target file - if you press ⟨RET⟩, the value of `org-default-notes-file` is used. Then the command offers the headings tree of the selected file. You can either immediately press ⟨RET⟩ to get the note appended to the file. Or you can use vertical cursor motion (⟨up⟩ and ⟨down⟩) and visibility cycling (⟨TAB⟩) to find a better place. Pressing ⟨RET⟩ or ⟨left⟩ or ⟨right⟩ leads to the following result.

Cursor position	**Key**	**Note gets inserted**
buffer-start	⟨RET⟩	as level 2 heading at end of file
on headline	⟨RET⟩	as sublevel of the heading at cursor

[2] The two autoload forms are only necessary if '`org.el`' is not part of the Emacs distribution or and XEmacs package.

	⟨left⟩	as same level, before current heading
	⟨right⟩	as same level, after current heading
not on headline	⟨RET⟩	at cursor position, level taken from context. Or use prefix arg to specify level manually.

So a fast way to store the note is to press `C-c C-c` ⟨RET⟩ ⟨RET⟩ to append it to the default file. Even shorter would be `C-u C-c C-c`, which does the same without even showing the tree. But with little extra effort, you can push it directly to the correct location.

Before inserting the text into a tree, the function ensures that the text has a headline, i.e. a first line that starts with a '`*`'. If not, a headline is constructed from the current date and some additional data. If the variable `org-adapt-indentation` is non-nil, the entire text is also indented so that it starts in the same column as the headline (after the asterisks).

5 TODO items

Org-mode does not maintain TODO lists as a separate document. TODO items are an integral part of the notes file, because TODO items usually come up while taking notes! With Org-mode, you simply mark any entry in a tree as being a TODO item. In this way, the information is not duplicated, and the entire context from which the item emerged is always present when you check.

Of course, this technique causes TODO items to be scattered throughout your file. Org-mode provides methods to give you an overview over all things you have to do.

5.1 Basic TODO functionality

Any headline can become a TODO item by starting it with the word TODO, for example

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

The most important commands to work with TODO entries are:

`C-c C-t` Rotate the TODO state of the current item between

```
,-> (unmarked) -> TODO -> DONE --.
'--------------------------------'
```

The same rotation can also be done "remotely" from the timeline and agenda buffers with the `t` command key (see Section 7.3 [Agenda commands], page 24).

`C-c C-v` View TODO items in a *sparse tree* (see Section 2.7 [Sparse trees], page 5). Folds the entire buffer, but shows all TODO items and the headings hierarchy above them. With prefix arg, show also the DONE entries.

`C-u C-c a` A `C-u` argument to the `org-agenda command` (see Section 7.2 [Agenda], page 22) collects all unfinished TODO items into a single place.

5.2 Extended use of TODO keywords

The default implementation of TODO entries is just two states: TODO and DONE. You can, however, use the TODO feature for more complicated things by configuring the variables `org-todo-keywords` and `org-todo-interpretation`. Using special setup, you can even use TODO keywords in different ways in different org files.

5.2.1 TODO keywords as workflow states

You can use TODO keywords to indicate different states in the process of working on an item, for example

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

With this setup, the command `C-c C-t` will cycle an entry from TODO to FEEDBACK, then to VERIFY, and finally too DONE. You may also use a prefix argument to quickly

select a specific state. For example *C-3 C-c C-t* will change the state immediately to VERIFY. If you define many keywords, you can use in-buffer completion (see Section 9.1 [Completion], page 30) to insert these words into the buffer.

5.2.2 TODO keywords as types

The second possibility is to use TODO keywords to indicate different types of action items. For example, you might want to indicate that items are for "work" or "home". Or, when you work with several people on a single project, you might want to assign action items directly to persons, by using their names as TODO keywords. This would be set up like this:

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

In this case, different keywords do not indicate a sequence, but rather different types. So it is normally not useful to change from one type to another. Therefore, in this case the behavior of the command *C-c C-t* is changed slightly[1]. When used several times in succession, it will still cycle through all names. But when when you return to the item after some time and execute *C-c C-t* again, it will switch from each name directly to DONE. Use prefix arguments or completion to quickly select a specific name.

5.2.3 Setting up TODO keywords for individual files

It can be very useful to use different aspects of the TODO mechanism in different files, which is not possible with the global settings described above. For file-local settings, you need to add special lines to the file which set the keywords and interpretation for that file only. For example, to set one of the two examples discussed above, you need one of the following lines, starting in column zero anywhere in the file:

```
#+SEQ_TODO: TODO FEEDBACK VERIFY DONE
#+TYP_TODO: Fred Sara Lucy Mike DONE
```

To make sure you are using the correct keyword, type '`#+`' into the buffer and then use *M-*⟨TAB⟩ completion.

Remember that the last keyword must always mean that the item is DONE (you may use a different word, though). Also note that in each file, only one of the two aspects of TODO keywords can be used. After changing one of these lines, use *C-c C-c* with the cursor still in the line to make the changes known to Org-mode[2].

If you want to use very many keywords, for example when working with a large group of people, you may split the names over several lines:

```
#+TYP_TODO: Fred Sara Lucy Mike
#+TYP_TODO: Luis George Jules Jessica
#+TYP_TODO: Kim Arnold Peter
#+TYP_TODO: DONE
```

[1] This is also true for the `t` command in the timeline and agenda buffers.

[2] Org-mode parses these lines only when Org-mode is activated after visiting a file. *C-c C-c* with the cursor in a line starting with '`#-`' is simply restarting Org-mode, making sure that these changes will be respected.

5.3 Priorities

If you use Org-mode extensively to organize your work, you may end up with a number of TODO entries so large that you'd like to prioritize them. This can be done by placing a *priority cookie* into the headline, like this

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

With its standard setup, Org-mode supports priorities '`A`', '`B`', and '`C`'. '`A`' is the highest priority. An entry without a cookie is treated as priority '`B`'. Priorities make a difference only in the agenda (see Section 7.2 [Agenda], page 22).

`C-c ,` Set the priority of the current item. 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 7.3 [Agenda commands], page 24).

`S-`⟨up⟩
`S-`⟨down⟩ Increase/decrease priority of current item. Note that these keys are also used to modify time stamps (see Section 6.2 [Creating timestamps], page 20). Furthermore, these keys are also used by CUA-mode (see Section 9.4 [Interaction], page 31).

6 Timestamps

Items can be labeled with timestamps to make them useful for project planning.

6.1 Time stamps, deadlines and scheduling

A time stamp is a specification of a date (possibly with time) in a special format, either '`<2003-09-16 Tue>`' or '`<2003-09-16 Tue 09:39>`'. A time stamp can appear anywhere in the headline or body of an org-tree entry. Its presence allows to show entries on specific dates in the agenda (see Section 7.2 [Agenda], page 22). We distinguish:

TIMESTAMP
: A simple time stamp just assigns a date/time to an item. In the timeline and agenda displays, the headline of the entry will be shown exactly on that date.

TIMERANGE
: Two time stamps connected by '`--`' denote a time 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>
```

DEADLINE
: If a time stamp is preceded by the word '`DEADLINE:`', the task (most likely a TODO item) is supposed to be finished on that date, and it will be listed then. In addition, the compilation 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>
```

SCHEDULED
: If a time stamp is preceded by the word '`SCHEDULED:`', it means you are planning to start working on that task on the given date. The headline will be listed under the given date. 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.

6.2 Creating timestamps

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

`C-c .`
: Prompt for a date and insert a corresponding time stamp. When the cursor is at a previously used time stamp, it is updated to NOW. When this command is used twice in succession, a time range is inserted.

`C-u C-c .` Like `C-c .`, but use the alternative format which contains date and time.

`C-c <` Insert a time stamp corresponding to the cursor date in the Calendar.

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

`C-c C-o` Access the agenda for the date given by the time stamp at point (see Section 7.2 [Agenda], page 22).

`C-c C-d` Insert '`DEADLINE`' keyword along with a stamp.

`C-c C-w` 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 C-w` shows all deadlines due tomorrow.

`C-c C-s` Insert '`SCHEDULED`' keyword along with a stamp.

`S-`⟨left⟩
`S-`⟨right⟩ Change date at cursor by one day. These key bindings conflict with CUA-mode (see Section 9.4 [Interaction], page 31).

`S-`⟨up⟩
`S-`⟨down⟩ Change the item under the cursor in a timestamp. The cursor can be on a year, month, day, hour or minute. Note that if the cursor is not at a time stamp, these same keys modify the priority of an item. (see Section 5.3 [Priorities], page 19). The key bindings also conflict with CUA-mode (see Section 9.4 [Interaction], page 31).

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

When Org-mode prompts for a date/time, the function reading your input will replace anything you choose not to specify with the current date and time. For details, see the documentation string of `org-read-date`. Also, a calender will pop up to allow selecting a date. The calendar can be fully controlled from the minibuffer, and a date can be selected with the following commands:

`<` Scroll calendar backwards by one month.

`>` Scroll calendar forwards by one month.

`mouse-1` Select date by clicking on it.

`S-`⟨right⟩ One day forward.

`S-`⟨left⟩ One day back.

`S-`⟨down⟩ One week forward.

`S-`⟨up⟩ One week back.

`M-S-`⟨right⟩ One month forward.

`M-S-`⟨left⟩ One month back.

⟨RET⟩ Choose date in calendar (only if nothing typed into minibuffer).

7 Timeline and Agenda

We have already described three commands to filter important information in an org file into a sparse tree (see Section 2.7 [Sparse trees], page 5):

- The TODO tree, (`C-c C-v`), see Chapter 5 [TODO items], page 17.
- The occur tree `C-c /`, see Chapter 5 [TODO items], page 17.
- Checking upcoming deadlines with `C-c C-w`, see Section 6.2 [Creating timestamps], page 20.

Instead of using the sparse trees, Org-mode can also collect and time-sort the important items into a separate buffer, which we call the *timeline* of the org file. It can also collect information from a *list of files* and in this way provide an *agenda* which covers all of your current projects, action items and appointments.

7.1 Timeline for a single file

The timeline shows all time-stamped items in a single Org-mode file, in *time-sorted view*. The main purpose of this command is to give an overview over events in a project.

`C-c C-r` Show a time-sorted view of the org file, with all time-stamped items of today or later. When called with a `C-u` prefix, past dates will be included as well. When called with two `C-u C-u` prefixes, all unfinished TODO entries (scheduled or not) are also listed under the current date.

The timeline is shown in a temporary buffer '`*Org Agenda*`'. The commands available in the Agenda buffer are listed in Section 7.3 [Agenda commands], page 24.

7.2 Agenda

An agenda can be compiled from one or more org files. The main purpose of this command is to act like a paper agenda, showing you all the tasks for the current day or week.

The Org-mode files to be processed in order to generate the agenda are listed in the variable `org-agenda-files`. You can customize this variable, but the easiest way to maintain it is through the following commands

`C-c [` Add current file to the list of agenda files

`C-c ]` Remove current file from the list of agenda files.

The Org menu contains the list of all files and can be used to quickly visit any of them.

The global command `org-agenda` compiles the agenda from all listed files.

`C-c a` Compile an agenda for the current week from a list of org files. The agenda shows the entries for each day. With a `C-u` prefix (or when the variable `org-agenda-include-all-todo` is `t`), all unfinished TODO items (also those without a date) are also listed at the beginning of the buffer, before the first date. The key binding `C-c a` is only a suggestion - see Section 1.2 [Installation and Activation], page 2.

The commands available in the Agenda buffer are listed in Section 7.3 [Agenda commands], page 24.

7.2.1 Categories

In the agenda buffer, each entry is preceded by a *category*, which is derived from the file name. The category can also be set with a special line anywhere in the buffer, looking like this:

```
#+CATEGORY: Thesis
```

After changing this line, press `C-c C-c` with the cursor still in the line, to make the changes known to org-mode. Otherwise, the change will only be active the next time you visit this file with Emacs.

The display in the agenda buffer looks best if the category is not longer than 10 characters.

7.2.2 Time-of-Day Specifications

Org-mode checks each agenda item for a time-of-day specification. The time can be part of the time stamp that triggered inclusion into the agenda, for example as in '`<2005-05-10 Tue 19:00>`'. Time ranges can be specified with two time stamps, like '`<2005-05-10 Tue 20:30>--<2005-05-10 Tue 22:15>`'.

In the headline of the entry itself, a time(range) may also appear as plain text (like '`12:45`' or a '`8:30-1pm`'. If the agenda integrates the Emacs diary (see Section 7.4 [Calendar/Diary integration], page 26), time specifications in diary entries are recognized as well.

For agenda display, Org-mode extracts the time and displays it in a standard 24 hour format as part of the prefix. The example times in the previous paragraphs would end up in the agenda like this:

```
 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 Marwin escorts the Hitchhikers to the bridge
```

If the agenda is in single-day mode, or for the display of today, the timed entries are embedded in a time grid, like

```
 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 Marwin escorts the Hitchhikers to the bridge
```

The time grid can be turned on and off with the variable `org-agenda-use-time-grid`, and can be configured with `org-agenda-time-grid`.

7.2.3 Sorting of agenda items

The entries for each day are sorted. The default order is to first collect all items containing an explicit time-of-day specification. These entries will be shown at the beginning of the list, as a *schedule* for the day. After that, items remain grouped in categories, in the sequence given by `org-agenda-files`. Within each category, items are sorted by priority (see Section 5.3 [Priorities], page 19).

The priority is a numerical quantity composed of the base priority (2000 for priority '`A`', 1000 for '`B`', and 0 for '`C`'), plus additional increments for overdue scheduled or deadline items.

Sorting can be customized using the variable `org-agenda-sorting-strategy`.

7.3 Commands in the agenda buffer

Entries in the agenda buffer are linked back to the org file or diary file where they originate. You are not allowed to edit the agenda buffer itself, but commands are provided to show and jump to the original entry location, and to edit the org-files "remotely" from the agenda buffer. In this way, all information is stored only once, and you don't risk that your agenda and note files diverge.

Some commands can be executed with mouse clicks on agenda lines. For the other commands, the cursor needs to be in the desired line. Most commands are available for both timelines and the agenda. The exceptions are marked.

Motion

n Next line (same as ⟨up⟩).

p Previous line (same as ⟨down⟩).

View/GoTo org file

mouse-3

⟨SPC⟩ Display the original location of the item in another window.

l Display original location and recenter that window.

mouse-2

⟨TAB⟩ Go to the original location of the item in another window.

⟨RET⟩ Go to the original location of the item and delete other windows.

f Toggle follow mode. In follow mode, as you move the cursor through the agenda buffer, the other window always shows the corresponding location in the org file.

Change display

o Delete other windows.

w Switch to weekly view (7 days displayed together)

`d` Switch to daily view (just one day displayed)

`D` Toggle the inclusion of diary entries. See Section 7.4 [Calendar/Diary integration], page 26.

`g` Toggle the time grid on and off. See also the variables `org-agenda-use-time-grid` and `org-agenda-time-grid`.

`r` Recreate the agenda buffer, for example to reflect the changes after modification of the time stamps of items with S-⟨left⟩ and S-⟨right⟩.

⟨right⟩ Display the following `org-agenda-ndays` days. For example, if the display covers a week, switch to the following week. With prefix arg, go forward that many times `org-agenda-ndays` days. Not available in timelines.

⟨left⟩ Display the previous dates. Not available in timelines.

`.` Goto today.

Remote editing

`0-9` Digit argument.

`t` Change the TODO state of the item, both in the agenda and in the original org file.

`,` Set the priority for the current item. Org-mode prompts for the priority character. If you reply with ⟨SPC⟩, the priority cookie is removed from the entry.

`p` Display weighted priority of current item.

`+`

`S-`⟨up⟩ Increase the priority of the current item. The priority is changed in the original buffer, but the agenda is not resorted. Use the `r` key for this.

`-`

`S-`⟨down⟩ Decrease the priority of the current item.

`S-`⟨right⟩ Change the time stamp associated with the current line by one day into the future. With prefix argument, change it by that many days. For example, `3 6 5 S-`⟨right⟩ will change it by a year. The stamp is changed in the original org file, but the change is not directly reflected in the agenda buffer. Use the `r` key to update the buffer.

`S-`⟨left⟩ Change the time stamp associated with the current line by one day into the past.

`>` Change the time stamp associated with the current line to today. The key `>` has been chosen, because it is the same as `S-.` on my keyboard.

`i` Insert a new entry into the diary. Prompts for the type of entry (day, weekly, monthly, yearly, anniversary, cyclic) and creates a new entry in the diary, just like `i d` etc. would do in the calendar. The date is taken from the cursor position.

Calendar commands

`c` Open the Emacs calendar and move to the date at the agenda cursor.

`c` When in the calendar, compute and show the Org-mode agenda for the date at the cursor.

`M` Show the phases of the moon for three month around current date.

`S` Show sunrise and sunset times. The geographical location must be set with calendar variables, see documentation of the Emacs calendar.

`C` Convert the date at cursor into many other cultural and historic calendars.

`H` Show holidays for three month around the cursor date.

Quit and Exit

`q` Quit Agenda, remove the agenda buffer.

`x` Exit agenda, remove the agenda buffer and all buffers loaded by Emacs for the compilation of the agenda. Buffers created by the user to visit org files will not be removed.

7.4 Calendar/Diary integration

Emacs contains the calendar and diary by Edward M. Reingold. The calendar displays a three-month calendar with holidays from different countries and cultures. The diary allows to keep track of anniversaries, lunar phases, sunrise/set, recurrent appointments (weekly, monthly) and more. In this way, it is quite complementary to Org-mode. It can be very useful to combine output from Org-mode with the diary.

The interaction between Org-mode and diary works both ways: You can list entries from the diary in the Org-mode agenda, from which many calendar and diary commands are directly accessible. Or you can display entries from the org agenda in the Emacs diary.

7.4.1 Including the diary into the agenda

In order to include entries from the Emacs diary into Org-mode's agenda, you only need to customize the variable

```
(setq org-agenda-include-diary t)
```

After that, everything will happen automatically. All diary entries including holidays, anniversaries etc will be included in the agenda buffer created by Org-mode. ⟨SPC⟩, ⟨TAB⟩, and ⟨RET⟩ can be used from the agenda buffer to jump to the diary file, in order to edit existing diary entries. The `i` command to insert new entries for the current date works in the agenda buffer, as well as the commands `S`, `M`, and `C` to display Sunrise/Sunset times, show lunar phases and to convert to other calendars, respectively. `c` can be used to switch back and forth between calendar and agenda.

7.4.2 Including the agenda into the diary

If you prefer to use the Emacs diary as your main instrument and if you wish to include the Org-mode agenda into it, the following steps are necessary: Autoload the function `org-diary` as shown above under Section 1.2 [Installation and Activation], page 2. You also need to use *fancy diary display* by setting in '`.emacs`':

```
(add-hook 'diary-display-hook 'fancy-diary-display)
```

Then include the following line into your '`~/diary`' file, in order to get the entries from all files listed in the variable `org-agenda-files`:

```
&%%(org-diary)
```

You may also select specific files with

```
&%%(org-diary) ~/path/to/some/org-file.org
&%%(org-diary) ~/path/to/another/org-file.org
```

If you now launch the calendar and press `d` to display a diary, the headlines of entries containing a timestamp, date range, schedule, or deadline referring to the selected date will be listed. Just like in Org-mode's agenda view, the diary for *today* contains additional entries for overdue deadlines and scheduled items. See also the documentation of the `org-diary` function.

8 Exporting

For printing and sharing of notes, an Org-mode document can be exported as an ASCII file, or as HTML. In the exported version, the first 3 outline levels will become headlines, defining a general document structure. Additional levels will be exported as itemize lists. If you want that transition to occur at a different level, specify it with a prefix argument. For example,

```
M-1 M-x org-export-as-html
```

creates only top level headlines and does the rest as items.

8.1 Export commands

`C-c C-x a` Export as ASCII file. If there is an active region, only the region will be exported. For an org file '`myfile.org`', the ASCII file will be '`myfile.txt`'. The file will be overwritten without warning.

`C-c C-x h` Export as HTML file '`myfile.html`'.

`C-c C-x C-h`
Export as HTML file and open it with a browser.

`C-c C-x t` Insert template with export options, see below.

`C-c :` Toggle fixed-width for line or region, see below.

8.2 HTML formatting

Not all text is transferred literally to the exported HTML file. The exporter implements the following interpretation:

- You can make words ***bold***, */italic/*, and _underlined_
- Simple TEX-like math constructs are interpreted:
  - '`10^22`' and '`J_n`' are super- and subscripts. You can quote '`^`' and '`_`' with a backslash: '`\_`' and '`\^`'
  - '`\alpha`' indicates a Greek letter, '`\to`' an arrow. You can use completion for these macros, just type '`\`' and maybe a few letters, and press *M-*⟨TAB⟩ to see possible completions.
- Tables are transformed into HTML tables. Data fields before the first horizontal separator line will be formatted as table header fields.
- If a headline starts with the word '`QUOTE`', the text below the headline will be typeset as fixed-width, to allow quoting of computer codes etc. Lines starting with '`:`' are also typeset in fixed-width font.
- If you want to include HTML tags which should be interpreted as such, mark them with a '`@`' like in '`@<b>bold text@</b>`'. Plain '`<`' and '`>`' are always transformed to '`<`' and '`>`' in HTML export.

If these conversions conflict with your habits of typing ASCII text, they can all be turned off with corresponding variables.

8.3 Export options

The exporter recognizes special lines in the buffer which provide additional information. These lines may be put anywhere in the file. The whole set of lines can be inserted into the buffer with `C-c C-x t`. For individual lines, a good way to make sure the keyword is correct it to type '`#+`' and then use `M-`⟨TAB⟩ completion (see Section 9.1 [Completion], page 30).

```
#+TITLE:     the title to be shown (default is the buffer name)
#+AUTHOR:    the author (default taken from user-full-name)
#+EMAIL:     his/her email address (default from user-mail-address)
#+LANGUAGE:  language for HTML, e.g. 'en' (org-export-default-language)
#+TEXT:      Some descriptive text to be inserted at the beginning.
#+TEXT:      Several lines may be given.
#+OPTIONS:   H:2  num:t  toc:t  \n:nil  t  ::t  |:t  ^:t  *:nil  TeX:t
```

The OPTIONS line is a compact form to specify export settings. Here you can

`H:` set the number of headline levels for export
`num:` turn on/off section-numbers
`toc:` turn on/off table of contents
`\n:` turn on/off linebreak-preservation
`@:` turn on/off quoted html tags
`::` turn on/off fixed-width sections
`|:` turn on/off tables
`^:` turn on/off TEX-like syntax for sub- and superscripts.
`*:` turn on/off emphasized text (bold, italic, underlined)
`TeX:` turn on/off TEX macros

8.4 Comment lines

Lines starting with '`#`' in column zero are treated as comments and will never be exported. Also entire subtrees starting with the word '`COMMENT`' will never be exported. Finally, any text before the first headline will not be exported either.

`C-c ;` Toggle the COMMENT keyword at the beginning of an entry.

9 Miscellaneous

9.1 Completion

Org-mode supports in-buffer completion. This type of completion does not make use of the minibuffer. You simply type a few letters into the buffer and use the key to complete text right there.

M-⟨TAB⟩ Complete word at point

- At the beginning of a headline, complete TODO keywords.
- After '\', complete TEX symbols supported by the exporter.
- After '`#+`', complete the special keywords like '`TYP_TODO`' or '`OPTIONS`' which set file-specific options for Org-mode. When the option keyword is already complete, pressing *M*-⟨TAB⟩ again will insert example settings for this keyword.
- Elsewhere, complete dictionary words using ispell.

9.2 Customization

There is a large number of variables which can be used to customize Org-mode. For the sake of compactness of the manual, we are not describing the variables here. For an overview of customization variables, use *M-x org-customize*. Or select `Browse Org Group` from the `Org->Customization` menu.

9.3 Frequently asked questions

1. **Org-mode seems to be a useful default mode for the various '`README`' files I have scattered through my directories. How do I turn it on for all '`README`' files?**

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

2. **I would like to have two windows on the same Org-mode file, but with different outline visibility. Is that possible?**
   In GNU Emacs, you may use *indirect buffers* which do exactly this. See the documentation on the command `make-indirect-buffer`. In XEmacs, this is currently not possible because of the different outline implementation.

3. **Is there an easy way to insert links to web locations?**
   Sure, just type or paste them into the buffer. A plain-text URL-like string is directly interpreted as a link.

4. **When I export my TODO list, every TODO item becomes a separate section. How do I enforce these items to be exported as an itemized list?**
   If you plan to use ASCII or HTML export, make sure things you want to be exported as item lists are level 4 at least, even if that does mean there is a level jump. For example

```
* Todays top priorities
**** TODO write a letter to xyz
**** TODO Finish the paper
**** Pick up kids at the school
```

Alternatively, if you need a specific value for the heading/item transition in a particular file, use the '`+OPTIONS`' line to configure the '`H`' switch.

```
+OPTIONS:   H:2; ...
```

5. **I would like to export only a subtree of my file to HTML. How?**
   If you want to export a subtree, mark the subtree as region and then export. Marking can be done with `C-c @ C-x C-x`, for example.
6. **Org-mode takes over the S-cursor keys. I also want to use CUA-mode, is there a way to fix this conflict?**
   Yes, see Section 9.4 [Interaction], page 31
7. **Is there an easy way to insert an empty table template with a default number of rows and columns?**
   To insert an empty table template, just type '`|-`' and use ⟨TAB⟩. The default size can be changed with the variable `org-table-default-size`. However, just starting to type the first line is usually much easier.
8. **One of my table columns has started to fill up with '`#ERROR`'. What is going on?**
   Org-mode tried to compute the column from other fields using a formula stored in the '`#+TBLFMT:`' line just below the table, and the evaluation of the formula fails. Fix the fields used in the formula, or fix the formula, or remove it!
9. **When I am in the last column of a table and just above a horizontal line in the table, pressing TAB creates a new table line *before* the horizontal line. How can I quickly move to the line *below* the horizontal line instead?**
   Press ⟨down⟩ (to get on the separator line) and then ⟨TAB⟩.
10. **How can I change the indentation of an entire table without fixing every line by hand?**
    The indentation of a table is set by the first line. So just fix the indentation of the first line and realign with ⟨TAB⟩.

9.4 Interaction with other packages

Org-mode can cooperate with the following packages:

'`table.el`' by Takaaki Ota
: Org mode cooperates with table.el, see Section 3.4 [table.el], page 12.

'`calc.el`' by Dave Gillespie
: Org-mode uses the calc package for implementing spreadsheet functionality in its tables (see Section 3.2 [Table calculations], page 8). Org-modes checks for the availability of calc by looking for the function `calc-eval` which should be autoloaded in your 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 3.2 [Table calculations], page 8), it is possible to use names for natural constants or units. Instead of defining you 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-mode checks for the function `constants-get`, which has to be autoloaded in your setup. See the installation instructions in the file '`constants.el`'.

'`CUA.el`' by Kim. F. Storm
: Keybindings in Org-mode 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. If you want to use one of these packages along with Org-mode, configure the variable `org-CUA-compatible`. When set, Org-mode will move the following keybindings in org-mode 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-+
S-RET   -> C-S-RET
```

  Yes, these are unfortunately more difficult to remember. If you want to have other replacement keys, look at the variable `org-disputed-keys`.

'`remember.el`' by John Wiegley
: Org mode cooperates with remember, see Section 4.2 [Remember], page 15.

'`planner.el`' by John Wiegley
: Planner is another tool to plan work and keep track of tasks. Planner uses a multi-file approach with project pages and day pages. Is based on Emacs-Wiki. If Planner is your primary tool, it can be useful to display the agenda entries resulting from org files in day-pages of the planner. This can be done through the diary of the calendar: Integrate org files into the diary as described above, and then turn on the diary support of planner.

9.5 Bugs

Here is a list of things which should work differently, but which I have found too hard to fix.

- If you call `fill-paragraph` (bound to *`M-q`*) in a table, the filling is correctly disabled. However, if some text directly (without an empty line in between) precedes or follows a table, calling `fill-paragraph` in that text will also fill the table like normal text. Also, `fill-region` does bypass the `fill-paragraph` code and will fill tables like normal text.
- When the application called by *`C-c C-o`* to open a file link fails (for example because the application does not exits or refuses to open the file), it does so silently. No error message is displayed.
- Recalculating a table line applies the formulas from left to right. If a formula calculated fields further down the row, multiple recalculation may be needed to get all fields consistent.

- Under XEmacs, if Org-mode entries are included into the diary, it is not possible to jump back from the diary to the org file. Apparently, the text properties are lost when the fancy-diary-display is used. However, from Org-mode's timeline and agenda buffers (created with `C-c C-r` and `C-c a`), things do work correctly.
- Linux should also have a default viewer application, using mailcap. Maybe we can use GNUS or VM mime code? Or dired's guessing commands? Any hints (or even patches) are appreciated.
- When you write '`x = a /b/ c`', b will be exported in italics.
- The exporters work well, but could be made more efficient.

9.6 Acknowledgments

Org-mode was written by Carsten Dominik, who still maintains it at the Org-mode homepage http://www.astro.uva.nl/~dominik/Tools/org/. The following people have helped the development along with ideas, suggestions and patches.

- Matthias Rempe (Oelde) provided ideas, a patch introducing Windows NT/2000 support, and quality control.
- Kevin Rogers contributed code to access VM files on remote hosts.
- Juergen Vollmer contributed code generating the table of contents in HTML output, and other export improvements.
- Christian Egli converted the documentation into TeXInfo format. He also showed me his plans for a multifile summary for Org-mode. Some of his ideas have found their way into the agenda.
- Philip Rooke created the Org-mode reference card. He also helped with beta testing and contributed a number of very useful ideas.
- Christian Schlauer proposed angular brackets around links, among other things.
- David Wainberg suggested to implement an archiving mechanism and helped testing.
- Linking to VM/BBDB/GNUS was inspired by Tom Shannon's '`organizer-mode.el`'.
- Scheduling TODO items was inspired by John Wiegley's '`planner.el`'.
- Sacha Chua, the current maintainer of Planner, offered linking code from Planner. I made use of the offer for links to RMAIL and Wanderlust.
- Oliver Oppitz sent several useful suggestions.
- Carsten Wimmer suggested some changes and helped fix a bug in linking to GNUS.
- Pavel Chalmoviansky reported bugs and suggested improvements related to the agenda treatment of items with specified time.
- Stefan Monnier provided a patch with lots of little fixes to keep the Emacs-Lisp compiler happy.
- Kai Grossjohann pointed out that a number of key bindings in Org-mode conflict with other packages.

10 Index

L

M

N

O

P

Q

R

S

T

U

V

W

11 Key Index

M

N

O

P

Q

R

S

T

W

X