Org Mode Manual

Release 4.29

by Carsten Dominik

This manual is for Org-mode (version 4.29).

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, as HTML, or (todo and agenda items only) as an iCalendar file.

Org-mode keeps simple things simple. When first fired up, it should feel like a simple, easy to use outliner. Complexity is not imposed, but a large amount of functionality is available when you need 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 for taking structured notes
- as an ASCII table editor with 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 (*C-h v org-version* ⟨RET⟩), 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?

Thank you 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, on the left margin. 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 10.5 [Clean view], page 44 describes a setup to realize this.

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

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. If the cursor is in a plain list item, a new item is created (see Section 2.8 [Plain lists], page 6). To force creation of a new headline, use a prefix arg, or first press ⟨RET⟩ to get to the beginning of the next line. 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. If the command is used at the beginning of a headline, the new headline is created before the current line. It at the beginning of any other line, the content of that line is made the new heading.

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

C-c C-x M-w
: Copy subtree to kill ring.

C-c C-x 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 8), 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

[1] See also the variables `org-show-hierarchy-above` and `org-show-following-heading`.

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 with an editing command.

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 8.2 [Agenda dispatcher], page 31). 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`.

Other commands are using sparse trees as well. For example *C-c C-v* creates a sparse TODO tree (see Section 5.1 [TODO basics], page 23).

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 export only the visible part of the document and print the resulting file.

2.8 Plain lists

Headlines define both the structure of the Org-mode file, and also lists (for example, TODO items (see Chapter 5 [TODO items], page 23) should be created using headline levels). However, when taking notes, the plain text is sometimes easier to read with hand-formatted lists. Org-mode supports editing such lists, and the HTML exporter (see Chapter 9 [Exporting], page 38) does parse and format them.

Org-mode knows ordered and unordered lists. Unordered list items start with '`-`', '`+`', or '`*`'[3] as bullets. Ordered list items start with '`1.`' or '`1)`'. 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. Indentation also determines the end of a list item. It ends before the next line that is indented like the bullet/number, or less. For example:

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

[3] 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 are visually indistinguishable from true headlines. In short: even though '`*`' is supported, it may be better to not use it for plain list items

```
** Lord of the Rings
My favorite scenes are (in this order)
1. Eowyns fight with the witch king
   + this was already my favorite scene in the book
   + I really like Miranda Otto.
2. The attack of the Rohirrim
3. Peter Jackson being shot by Legolas
    - on DVD only
   He makes a really funny face when it happens.
But in the end, not individual scenes matter but the film as a whole.
```

Org-mode supports these lists by tuning filling and wrapping commands to correctly deal with them. Furthermore, the following commands act on items when the cursor is in the first line of an item (the line with the bullet or number).

⟨TAB⟩ Items can be folded just like headline levels if you set the variable `org-cycle-include-plain-lists`. The level of an item is then given by the indentation of the bullet/number. However, items are always subordinate to real headlines, the hierarchies remain completely separated.

M-⟨RET⟩ Insert new item at current level. With prefix arg, force a new heading (see Section 2.5 [Structure editing], page 4). If this command is used in the middle of a line, the line is *split* and the rest of the line becomes the new item. If this command is executed in the *whitespace before a bullet or number*, the new item is created *before* the current item. If the command is executed in the white space before the text that is part of an item but does not contain the bullet, a bullet is added to the current line.

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-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.

C-c C-c Renumber the ordered list at the cursor.

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.

When typing text into a field, Org-mode 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 | 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 not, lines are split at whitespace into fields. You can use a prefix argument to indicate the minimum number of consecutive spaces required to identify a field separator (default: just one).
If there is no active region, this command creates an empty Org-mode table. However, it's easier to just start typing, like *|Name|Phone|Age* ⟨RET⟩ *|-* ⟨TAB⟩.

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.

`C-c ^` Sort the table lines in the region. Point and mark must be in the first and last line to be included, and must be in the column that should be used for sorting. The command prompts for numerical versus alphanumerical sorting.

Regions

`C-c C-x 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-x 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-x 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 is made 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 3.3 [Table calculations], page 11.

`C-#` Rotate the calculation mark in first column through the states ‘’, ‘`#`’, ‘`*`’, ‘`!`’, ‘`$`’. For the meaning of these marks see Section 3.3.4 [Advanced features], page 13. 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 10.8 [Interaction], page 48).

Miscellaneous

`C-c ‘` Edit the current field in a separate window. This is useful for fields that are not fully visible (see Section 3.2 [Narrow columns], page 10). When called with a `C-u` prefix, just make the full field visible, so that it can be edited in place.

`C-c` ⟨TAB⟩ This is an alias for `C-u C-c ‘` to make the current field fully visible.

`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 in your way on 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 Narrow columns

The width of columns is automatically determined by the table editor. Sometimes a single field or a few fields need to carry more text, leading to inconveniently wide columns.

To limit[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 no more than 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, it is only invisible. To see the full text, hold the mouse over the field - a tooltip window will show the full content. To edit such a field, use the command `C-c '` (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
```

3.3 Calculations in tables

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. Org-mode has two levels of complexity for table calculations. 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.

3.3.1 Formula syntax

A formula can be any algebraic expression understood by the Emacs 'calc' package. Note that 'calc' has the slightly 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:

`$`	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

[1] This feature does not work on XEmacs.

`$P1..$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 '`vmean`' and '`vsum`'.

'`$name`' is interpreted as the name of a column, parameter or constant. Constants are defined globally through the variable `org-table-formula-constants`. If you have the '`constants.el`' package, it will also be used to resolve constants, including natural constants like '`$h`' for Planck's constant, and units like '`$km`' for kilometers. Column names and parameters can be specified in special table lines. These are described below, see Section 3.3.4 [Advanced features], page 13.

A formula can contain an optional mode string after a semicolon. This string consists of flags to influence calc's modes[2] 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 fixed 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, up to here
`taylor($3,x=7,2)`	taylor series of $3, at x=7, second degree

3.3.2 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 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 like with '`calc`' formulas, you can provide a format specifier after a semicolon. A few examples:

swap the first two characters of the content of column 1

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

Add columns 1 and 2, equivalent to the calc's `$1+$2`

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

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

3.3.3 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.

3.3.4 Advanced features

If you 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 *` will 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 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, 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 exempt from recalculation with `C-u C-c *`. All lines that should be recalculated should be marked with '`#`' or '`*`'.

3.3.5 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 into 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.

3.3.6 Editing and debugging formulas

To edit a column or field formula, use the commands `C-c =` and `C-u C-c =`, respectively. The currently active expression is then presented as default in the minibuffer, where 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, for example by pressing *C-c =* ⟨RET⟩ in a field. Detailed information will be displayed.

3.3.7 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
```

3.4 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)
```

3.5 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, and also part of Emacs 22). 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 inside a file, and external links to other files, Usenet articles, emails and much more.

4.1 Link format

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

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

Once a link in the buffer is complete (all brackets present), Org-mode 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 not 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 does hide 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. 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 19). The preferred match for such a link is a dedicated target: The same string in double angular brackets. Targets may be located anywhere, often it is convenient to put them into a comment line, for example

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

In HTML export (see Section 9.2 [HTML export], page 38), such targets will become named anchors for direct access through '`http`' links.

If no dedicated target exists, Org-mode will search for the words in the link. In the above example the search would be for '`my target`'. Links starting with a star like '`*My Target`' restrict the search to headlines. When searching, Org-mode will first try an exact match, but then move on to more and more lenient searches. For example, the link '`[[*My Targets]]`' will find any of the following:

```
** My targets
** TODO my targets are bright
** my 20 targets are
```

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. See Section 4.4 [Handling links], page 19, for more commands creating links.

Following a link pushes a mark onto Org-mode'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

You can configure Org-mode to link any occurrences of certain target names in normal text. 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-mode 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.

4.2.2 CamelCase words as links

Org-mode also supports CamelCase words as links. This feature is not turned on by default because of the inconsistencies this system suffers from. To activate CamelCase words as links, you need to customize the option `org-activate-links`. A CamelCase word then leads to a text search such that '`CamelCaseLink`' is equivalent to '`[[camel case link]]`'.

4.3 External links

Org-mode supports links to files, websites, Usenet and email messages; and BBDB database entries. External links are URL-like locators. 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
`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:Richard Stallman`	BBDB link
`shell:ls *.org`	A shell command

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 17), for example:

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

Org-mode 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 to remove ambiguities about the end of the link, enclose them in angular brackets.

4.4 Handling links

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

`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 Org-mode files, if there is a '`<<target>>`' at the cursor, the link points to the target. Otherwise it points to the current headline. For VM, RMAIL, WANDERLUST, MH-E, GNUS and BBDB buffers, the link will indicate to the current article/entry. For W3 and W3M buffers, the link goes to the current URL. For any other files, the link will point to the file, with a search string (see Section 4.5 [Search options], page 20) 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 4.6 [Custom searches], page 21. 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 text for an internal link, or one of the link type prefixes mentioned in the examples above. Through completion, all links stored during the current session can be accessed. The link will be inserted into the buffer, along with a descriptive text. 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. By using this command, the links are automatically enclosed in double brackets, and you will be asked for the optional descriptive text.

`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 the linked file is in the current directory or in a subdirectory 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 to edit the link and description parts of the link.

C-c C-o
: 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 commands runs the corresponding search. When the cursor is on a TAGS list in a headline, it creates the corresponding TAGS view. If the cursor is on a time stamp, it compiles the agenda for that date. Furthermore, it will visit text files in '`file:`' links 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 you want to override the default application and visit the file with Emacs, use a *C-u* prefix.

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

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

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

C-c &
: 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.

4.5 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[1] colon. For example, when the command *C-c l* creates a link (see Section 4.4 [Handling links], page 19) 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::/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 17. In HTML export (see Section 9.2 [HTML export], page 38), such a file link will become an html reference to the corresponding named anchor in the linked file.

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

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

`/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 like '`[[find me]]`' would.

4.6 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. 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-mode actually uses this mechanism for BibTEX database files, and you can use the corresponding code as an implementation example. Search for '`BibTeX links`' in the source file.

4.7 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.

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

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

In combination with Org-mode, you can use templates to generate different types of remember notes. For example, if you would like to use one template to create general TODO entries, and another one for journal entries, you could use:

```
(setq org-remember-templates
      '((?t "* TODO %?\n  %i\n  %a" "~/org/TODO.org")
        (?j "* %U %?\n\n  %i\n  %a" "~/org/JOURNAL.org")))
```

In these entries, the character specifies how to select the template, the first string specifies the template, and the (optional) second string specifies a default file (overruling `org-default-notes-file`) as a target for this note.

When you call *M-x remember* to remember something, org will prompt for a key to select the template and then prepare the buffer like

```
* TODO
  <file:link to where you called remember>
```

or

```
* [2006-03-21 Tue 15:37]

  <file:link to where you called remember>
```

See the variable `org-remember-templates` for more details.

When you are finished composing a note with remember, you have to press *C-c C-c* to file the note away. 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
	⟨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 8.7 [Agenda commands], page 35).

`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. With numerical prefix N, show the tree for the Nth keyword in the variable `org-todo-keywords`.

`C-c a t` Show the global TODO list. This collects the TODO items from all agenda files (see Chapter 8 [Agenda views], page 31) into a single buffer. The buffer is in `agenda-mode`, so there are commands to examine and manipulate the TODO entries directly from that buffer (see Section 8.7 [Agenda commands], page 35). See Section 8.4 [Global TODO list], page 34, for more information.

5.2 Progress Logging

If you want to keep track of *when* a certain TODO item was finished, turn on logging with

```
(setq org-log-done t)
```

Then each time you turn a TODO entry into DONE using either `C-c C-t` in the Org-mode buffer or `t` in the agenda buffer, a line '`CLOSED: [timestamp]`' will be inserted just after the headline. If you turn the entry back into a TODO item again through further state cycling, that line will be removed again. In the timeline (see Section 8.6 [Timeline], page 35) and in the agenda (see Section 8.3 [Weekly/Daily agenda], page 32), you can then use the `L` key to display the TODO items closed on each day, giving you an overview of what has been done on a day.

5.3 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.

Note that *tags* are another way to classify headlines in general and TODO items in particular (see Chapter 7 [Tags], page 29).

5.3.1 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)
```

Changing these variables becomes only effective in a new Emacs session. With this setup, the command `C-c C-t` will cycle an entry from TODO to FEEDBACK, then to VERIFY, and finally to 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 10.1 [Completion], page 42) to insert these words into the buffer.

5.3.2 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". If you are into David Allen's *Getting Things DONE*, you might want to use todo types '`NEXTACTION`', '`WAITING`', '`MAYBE`'. 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 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. You can also review the items of a specific TODO type in a sparse tree by using a numeric prefix to `C-c C-v`. For example, to see all things Lucy has to do, you would use `C-3 C-c C-v`. To collect Lucy's items from all agenda files into a single buffer, you would use the prefix arg as well when creating the global todo list: `C-3 C-c t`.

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

5.3.3 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
```

5.4 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 8.3 [Weekly/Daily agenda], page 32).

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 8.7 [Agenda commands], page 35).

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 27). Furthermore, these keys are also used by CUA-mode (see Section 10.8 [Interaction], page 48).

[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.

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 entries to be shown on specific dates in the agenda (see Section 8.3 [Weekly/Daily agenda], page 32). We distinguish:

TIMESTAMP
: A simple time stamp just assigns a date/time to an item. This is just like writing down an appointment in a paper agenda, or like writing down an event in a diary, when you want to take not of when something happened. In the timeline and agenda displays, the headline of an entry associated with a plain time stamp 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>
```

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. So this is not about recording an event, but about planning your work. 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 until completed.

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

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>
```

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. The default time can be rounded to multiples of 5 minutes, see the option `org-time-stamp-rounding-minutes`.

`C-c !` Like `C-c .`, but insert an inactive time stamp not triggering the agenda.

`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 8.3 [Weekly/Daily agenda], page 32).

`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 10.8 [Interaction], page 48).

`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 5.4 [Priorities], page 25). The key bindings also conflict with CUA-mode (see Section 10.8 [Interaction], page 48).

`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 Tags

If you wish to implement a system to cross-correlate information, an excellent way is to assign *tags* to headline. Org-mode has extensive support for using tags.

Every headline can contain a list of tags, at the end of the headline. Tags are normal words containing letters, numbers, '_', and '@'. Tags must be preceded and followed by a single colon; like ':WORK:'. Several tags can be specified like ':WORK:URGENT:'.

7.1 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:'. When executing tag searches and Org-mode finds that a certain headline matches the search criterion, it will not check any sublevel headline, assuming that these likely also match, and that the list of matches can become very long. However, this may not be what you want, and you can influence inheritance and searching using the variables `org-use-tag-inheritance` and `org-tags-match-list-sublevels`.

7.2 Setting tags

As Org-mode deals with plain text files, tags can simply be typed into the buffer. After a colon, *M-*⟨TAB⟩ offers completion on all tags being used in the current buffer. There is also a special command for inserting tags:

C-c C-c Enter new tags for the current headline. The minibuffer will prompt for a list of tags and offer completion with respect to all other tags used in the current buffer. Several tags, separated by colons, may be specified at the prompt. 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 23).

7.3 Tag searches

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

*C-c * Create a sparse tree with all headlines matching a tags search.

C-c a m Create a global list of tag matches from all agenda files. See Section 8.5 [Matching headline tags], page 35.

`C-c a M` 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`).

A *tags* search string can use Boolean operators '`&`' for AND and '`|`' for OR. '`&`' binds more strongly than '`|`'. Parenthesis are currently not implemented. A tag may also be preceded by '`-`', to select against it, and '`+`' is syntactic sugar for positive selection. The AND operator '`&`' is optional when '`+`' or '`-`' is present. For example, '`+WORK-BOSS`' would select all headlines that are tagged '`:WORK:`', but discard those also tagged '`:BOSS:`'. The search string '`WORK|LAPTOP`' selects all lines tagged '`:WORK:`' or '`:LAPTOP:`'. The string '`WORK|LAPTOP&NIGHT`' requires that the '`:LAPTOP:`' lines are also tagged '`NIGHT`'.

8 Agenda Views

Due to the way Org-mode works, TODO items, time-stamped items, and tagged headlines can be scattered throughout a file or even a number of files. To get an overview over open action items, or over events that are important for a particular date, this information must be collected, sorted and displayed in an organized way.

Org-mode can select items based on various criteria, and display them in a separate buffer. Three different views are provided:

- an *agenda* that is like a calendar and shows information for specific dates
- a *TODO list* that covers all unfinished action items, and
- a *tags view* that shows information based on the tags associated with headlines in the outline tree.

The extracted information is displayed in a special *agenda buffer*. This buffer is read-only, but provides commands to visit the corresponding locations in the original Org-mode files, and even to edit these files remotely.

8.1 Agenda files

The information to be shown is collected from all *agenda files*, the files listed in the variable `org-agenda-files`[1]. Thus even if you only work with a single Org-mode file, this file should be put into that list[2]. You can customize `org-agenda-files`, but the easiest way to maintain it is through the following commands

`C-c [` Add current file to the list of agenda files. The file is added to the front of the list. If it was already in the list, it is moved to the front. With prefix arg, file is added/moved to the end.

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

`C-,` Cycle through agenda file list, visiting one file after the other.

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

8.2 The agenda dispatcher

The views are created through a dispatcher that should be bound to a global key, for example `C-c a` (see Section 1.2 [Installation and activation], page 2). In the following we will assume that `C-c a` is indeed how the dispatcher is accessed and list keyboard access to commands accordingly. After pressing `C-c a`, an additional letter is required to execute a command. The dispatcher offers the following default commands:

`a` Create the calendar-like agenda (see Section 8.3 [Weekly/Daily agenda], page 32).

[1] If the value of that variable is not a list, but a single file name, then the list of agenda files will be maintained in that external file.

[2] When using the dispatcher pressing `1` before selecting a command will actually limit the command to the current file, and ignore `org-agenda-files` until the next dispatcher command.

`t / T` Create a list of all TODO items (see Section 8.4 [Global TODO list], page 34).

`m / M` Create a list of headlines matching a TAGS expression (see Section 8.5 [Matching headline tags], page 35).

You can also define custom commands that will be accessible through the dispatcher, just like the default commands. Custom commands are global searches for tags and specific TODO keywords, or a variety of sparse tree creating commands (see Section 2.7 [Sparse trees], page 5). As sparse trees are only defined for a single org-mode file, these latter commands act on the current buffer instead of the list of agenda files.

Custom commands are configured in the variable `org-agenda-custom-commands`. You can customize this variable, for example by pressing `C-c a C`. You can also directly set it with Emacs Lisp in '`.emacs`'. For example:

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING")
        ("u" tags "+BOSS-URGENT")
        ("U" tags-tree "+BOSS-URGENT")
        ("f" occur-tree "\\<FIXME\\>")))
```

will define `C-c a w` as a global search for TODO entries with '`WAITING`' as the TODO keyword, `C-c a u` as a global tags search for headlines marked '`:BOSS:`' but not '`:URGENT:`', `C-c a U` to do the same search but only in the current buffer and display the result as a sparse tree, and `C-c a f` to create a sparse tree with all entries containing the word '`FIXME`'. For more information, look at the documentation string of the variable `org-agenda-custom-commands`.

8.3 The weekly/daily agenda

The purpose of the weekly/daily *agenda* is to act like a page of a paper agenda, showing all the tasks for the current week or day.

`C-c a 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 (including those without a date) are also listed at the beginning of the buffer, before the first date.

Remote editing from the agenda buffer means, for example, that you can change the dates of deadlines and appointments from the agenda buffer. The commands available in the Agenda buffer are listed in Section 8.7 [Agenda commands], page 35.

8.3.1 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
```

If there are several such lines in a file, each specifies the category for the text below it (but the first category also applies to any text before the first CATEGORY line). The display in the agenda buffer looks best if the category is not longer than 10 characters.

8.3.2 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 8.3.3 [Calendar/Diary integration], page 33), 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`.

8.3.3 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 you 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.

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.

8.3.4 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 5.4 [Priorities], page 25).

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`.

8.4 The global TODO list

The global TODO list contains all unfinished TODO items, formatted and collected into a single place.

`C-c a t` Show the global TODO list. This collects the TODO items from all agenda files (see Chapter 8 [Agenda views], page 31) into a single buffer. The buffer is in `agenda-mode`, so there are commands to examine and manipulate the TODO entries directly from that buffer (see Section 8.7 [Agenda commands], page 35). See Section 8.4 [Global TODO list], page 34, for more information.

`C-c a T` Like the above, but allows selection of a specific TODO keyword. You can also do this by specifying a prefix argument to `C-c a t`. With a `C-u` prefix you are prompted for a keyword. With a numeric prefix, the Nth keyword in `org-todo-keywords` is selected. The `r` key in the agenda buffer regenerates it, and you can give a prefix argument to this command to change the selected TODO keyword, for example `3 r`. If you often need a search for a specific keyword, define a custom command for it (see Section 8.2 [Agenda dispatcher], page 31).

Remote editing of TODO items means that you can change the state of a TODO entry with a single key press. The commands available in the TODO list are described in Section 8.7 [Agenda commands], page 35.

8.5 Matching headline tags

If headlines in the agenda files are marked with *tags* (see Chapter 7 [Tags], page 29), you can select headlines based on the tags that apply to them and collect them into an agenda buffer.

`C-c a m` Produce a list of all headlines that match a given set of tags. The command prompts for a selection criterion, which is a boolean logic expression with tags, like '`+WORK+URGENT-WITHBOSS`' or '`WORK|HOME`' (see Chapter 7 [Tags], page 29). If you often need a specific search, define a custom command for it (see Section 8.2 [Agenda dispatcher], page 31).

`C-c a M` Like `C-c a m`, but only select headlines that are also TODO items and force checking subitems (see variable `org-tags-match-list-sublevels`.

The commands available in the tags list are described in Section 8.7 [Agenda commands], page 35.

8.6 Timeline for a single file

The timeline is not really an agenda view, because it only summarizes items from a single Org-mode file. But it also uses the agenda buffer and provides similar commands, so we discuss it here. The timeline shows all time-stamped items in a single Org-mode file (or the selected part of it), in a *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. When called with a `C-u` prefix, all unfinished TODO entries (scheduled or not) are also listed under the current date.

The commands available in the timeline buffer are listed in Section 8.7 [Agenda commands], page 35.

8.7 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, removing the risk that your agenda and note files may 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.

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

mouse-1

⟨TAB⟩ Go to the original location of the item in another window. Under Emacs 22, *mouse-1* will also works for this.

⟨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. The initial setting for this mode in new agenda buffers can be set with the variable `org-agenda-start-with-follow-mode`.

l Toggle Logbook mode. In Logbook mode, entries that where marked DONE while logging was on (variable `org-log-done`) are shown in the agenda.

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 8.3.3 [Calendar/Diary integration], page 33.

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⟩. When the buffer is the global todo list, a prefix argument is interpreted to create a selective list for a specific TODO keyword.

⟨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.

⟨left⟩ Display the previous dates.

. 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.

T Show all tags associated with the current item. Because of inheritance, this may be more than the tags listed in the line itself.

: Set tags for the current headline.

`,` 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.

`C-c C-s` Schedule this item

`C-c C-d` Set a deadline for this 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 the three months 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.

`C-c C-x C-c`
Export a single iCalendar file containing entries from all agenda files.

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.

9 Exporting

Org-mode documents can be exported into a variety of other formats. For printing and sharing of notes, ASCII export produces a readable and simple version of an Org-mode file. HTML export allows to publish a notes file on the web, while the XML format provides a solid base for exchange with a broad range of other applications. To incorporate entries with associated times like deadlines or appointments into a desktop calendar program like iCal, Org-mode can also produce extracts in the iCalendar format. Currently Org-mode only supports export, not import of these different formats.

When exporting, Org-mode uses special conventions to enrich the output produced. See Section 9.5 [Text interpretation], page 40, for more details.

9.1 ASCII export

ASCII export produces an simple and very readable version of an Org-mode file.

`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 v a`
Export only the visible part of the document.

In the exported version, the first 3 outline levels will become headlines, defining a general document structure. Additional levels will be exported as itemized lists. If you want that transition to occur at a different level, specify it with a prefix argument. For example,

```
C-1 C-c C-x a org-export-as-ascii
```

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

9.2 HTML export

Org-mode contains an HTML exporter with extensive HTML formatting, in ways similar to John Grubers *markdown* language, but with additional support for tables.

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

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

`C-c C-x v h`
`C-c C-x v b`
Export only the visible part of the document.

In the exported version, the first 3 outline levels will become headlines, defining a general document structure. Additional levels will be exported as itemized lists. If you want that transition to occur at a different level, specify it with a prefix argument. For example,

```
C-2 C-c C-x b
```

creates two levels of headings and does the rest as items.

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.

You can also give style information for the exported file. The default specification can be configured through the option `org-export-html-style`. If you want to use a file-local style, you may use file variables, best wrapped into a COMMENT section at the end of the outline tree. For example:

```
* COMMENT HTML style specifications

# Local Variables:
# org-export-html-style: "   <style type=\"text/css\">
       p {font-weight: normal; color: gray; }
       h1 {color: black; }
   </style>"
# End: ***
```

Remember to execute *M-x normal-mode* after changing this to make the new style visible to Emacs. This command restarts org-mode for the current buffer and forces Emacs to re-evaluate the local variables section in the buffer.

9.3 XML export

Org-mode contains an XML exporter that produces XOXO-style XML. Currently, this exporter only handles the general outline structure and does not interpret any additional Org-mode features.

C-c C-x C-x
: Export as XML file '`myfile.xml`'.

C-c C-x v x
: Export only the visible part of the document.

9.4 iCalendar export

Some people like to 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 have deadlines and other time-stamped items in Org-mode files show up in the calendar application. Org-mode can export calendar information in the standard iCalendar format.

C-c C-x i
: Create iCalendar entries for the current file and store them in the same directory, using a file extension '`.ics`'.

C-c C-x C-i
: Like *C-c C-x 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-x c
: 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`.

How this calendar is best read and updated, depends on the application you are using. For example, when using iCal under Apple MacOS X, you could create a new calendar '`OrgMode`' (the default name for the calendar created by `C-c C-x c`, see the variables `org-icalendar-combined-name` and `org-combined-agenda-icalendar-file`). Then set Org-mode to overwrite the corresponding file '`~/Library/Calendars/OrgMode.ics`'. You may even use AppleScript to make iCal re-read the calendar files each time a new version of '`OrgMode.ics`' is produced. Here is the setup needed for this:

```
(setq org-combined-agenda-icalendar-file
    "~/Library/Calendars/OrgMode.ics")
(add-hook 'org-after-save-iCalendar-file-hook
 (lambda ()
  (shell-command
   "osascript -e 'tell application \"iCal\" to reload calendars'")))
```

9.5 Text interpretation by the exporter

The exporter backends interpret additional structure in the Org-mode file in order to produce better output.

9.5.1 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.

9.5.2 Enhancing text for export

Some of the export backends of Org-mode allow for sophisticated text formatting, this is true in particular for the HTML backend. Org-mode has a number of typing conventions that allow to produce a richly formatted output.

- Plain lists '`-`', '`*`' or '`+`' as bullet, or with '`1.`' or '`2)`' as enumerator will be recognized and transformed if the backend supports lists. See See Section 2.8 [Plain lists], page 6.
- 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 native tables under the exporter, if the export backend supports this. 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.

  C-c : Toggle fixed-width for entry (QUOTE) or region, see below.

If these conversions conflict with your habits of typing ASCII text, they can all be turned off with corresponding variables (see the customization group org-export-general, and the following section which explains how to set export options with special lines in a buffer.

9.5.3 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 is to type '#+' and then use M-⟨TAB⟩ completion (see Section 10.1 [Completion], page 42).

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

```
#+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

10 Miscellaneous

10.1 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 CamelCase versions of all headlines in the buffer.
- After ':', complete tags used elsewhere in the buffer.
- 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.

10.2 Customization

There are more than 100 variables that can be used to customize Org-mode. For the sake of compactness of the manual, we are 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 10.3 [Summary of in-buffer settings], page 42).

10.3 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 settings words con 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.

#+STARTUP:

This line sets options to be used at startup of org-mode, when an Org-mode 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 at all, show everything

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`	don't align tables on startup

Here are the options for hiding leading stars in outline headings. 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
`odd`	allow only odd outline levels (1,3,...)
`oddeven`	allow all outline levels

#+SEQ_TODO: #+TYP_TODO:
: These lines that the TODO keywords and their interpretation in the current file. The corresponding variables are `org-todo-keywords` and `org-todo-interpretation`.

#+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.

#+TBLFM: This line contains the formulas for the table directly above the line.

#+TITLE:, #+AUTHOR:, #+EMAIL:, #+LANGUAGE:, #+TEXT:, #+OPTIONS:
: These lines provide setting for exporting files. For more details see Section 9.5.3 [Export options], page 41.

10.4 The very busy C-c C-c key

The key *C-c C-c* has many purposes in org-mode, which are all mentioned scattered throughout this manual. One specific function of this key is to add *tags* to a headline (see Chapter 7 [Tags], page 29). In many other circumstances it means something like *Hey Org-mode, look here and update according to what you see here.* Here is a summary what this means in different contexts.

- 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 cursor is inside a table created by the '`table.el`' package, activate that table.
- If the current buffer is a remember buffer, close 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 on a numbered item in a plain list, renumber the ordered list.

10.5 A cleaner outline view

Some people find it noisy and distracting that the Org-mode headlines are starting with a potentially large number of stars. For example the tree from Section 2.2 [Headlines], page 3:

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

Unfortunately this is deeply ingrained into the code of Org-mode and cannot be easily changed. You can, however, modify the display in such a way that all leading stars become invisible and the outline more easy to read. To do this, customize the variable `org-hide-leading-stars` like this:

```
(setq org-hide-leading-stars t)
```

or change this on a per-file basis with one of the lines (anywhere in the buffer)

```
#+STARTUP: showstars
#+STARTUP: hidestars
```

Press *C-c C-c* with the cursor in a '`STARTUP`' line to activate the modifications.

With stars hidden, the tree becomes:

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

Note that 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 are 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.

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:

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

```
    * 3rd level
      more text
* Another top level headline
```

In order to make the structure editing and export commands handle this convention correctly, use

```
(setq org-odd-levels-only t)
```

or set this on a per-file basis with one of the following lines (don't forget to press *C-c C-c* with the cursor in the startup line to activate changes immediately).

```
#+STARTUP: odd
#+STARTUP: oddeven
```

You can convert an Org-mode 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*.

10.6 Using org-mode on a tty

Org-mode uses a number of keys that are not accessible on a tty. This applies to most special keys like cursor keys, ⟨TAB⟩ and ⟨RET⟩, when these are combined with modifier keys like ⟨Meta⟩ and/or ⟨Shift⟩. Org-mode uses these bindings because it needs to provide keys for a large number of commands, and because these keys appeared particularly easy to remember. In order to still be able to access the core functionality of Org-mode on a tty, alternative bindings are provided. Here is a complete list of these bindings, which are obviously more cumbersome to use. Note that sometimes a work-around can be better. For example changing a time stamp is really only fun with *S-*⟨cursor⟩ keys. On a tty you would rather use *C-c .* to re-insert the timestamp.

Default	**Alternative 1**	**Alternative 2**
S-⟨TAB⟩	*C-u* ⟨TAB⟩	
M-⟨left⟩	*C-c C-x l*	⟨Esc⟩ ⟨left⟩
M-S-⟨left⟩	*C-c C-x L*	
M-⟨right⟩	*C-c C-x r*	⟨Esc⟩ ⟨right⟩
M-S-⟨right⟩	*C-c C-x R*	
M-⟨up⟩	*C-c C-x u*	⟨Esc⟩ ⟨up⟩
M-S-⟨up⟩	*C-c C-x U*	
M-⟨down⟩	*C-c C-x d*	⟨Esc⟩ ⟨down⟩
M-S-⟨down⟩	*C-c C-x 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 C-x* ⟨left⟩	
S-⟨right⟩	*C-c C-x* ⟨right⟩	
S-⟨up⟩	*C-c C-x* ⟨up⟩	
S-⟨down⟩	*C-c C-x* ⟨down⟩	

10.7 Frequently asked questions

1. **When I try to use Org-mode, I always get `(wrong-type-argument keymapp nil)`.**
   This is a conflict with an outdated version of the '`allout.el`' package which pretends to be also the standard outline-mode but is not. This happens with older versions of '`allout.el`', for example the one distributed with Emacs 21. Upgrade to Emacs 22 and this problem will disappear. If for some reason you cannot do this, make sure that org.el is loaded *before* '`allout.el`', for example by putting `(require 'org)` early enough into your '`.emacs`' file.
2. **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))
   ```
3. **Some of my links stopped working after I upgraded to a version 4.20 or later. Why is this, and how can I fix it?**
   These must be links in plain text, containing white space, such as '`bbdb:Richard Stallman`'. You need to protect these links by putting double brackets around them, like '`[[bbdb:Richard Stallman]]`'.
4. **I see that Org-mode now creates links using the double bracket convention that hides the link part and the brackets, only showing the description part. How can I convert my old links to this new format?**
   Execute once in each Org-mode file: *`M-x org-upgrade-old-links`*. This replaces angular brackets with the new link format.
5. **I don't care if you find the new bracket links great, I am attached to the old style using angular brackets and no hiding of the link text. Please give them back to me, don't tell me it is not possible!**
   Would I let you down like that? If you must, you can do this
   ```
   (setq org-link-style 'plain
         org-link-format "<%s>")
   ```
6. **When I am executing shell links I always get a confirmation prompt and need to type *`yes`* ⟨RET⟩, thats 4 key presses! Can I get rid of this?**
   The confirmation is there to protect you from unwantingly execute potentially dangerous commands. For example, imagine a link '`[[shell:rm -rf ~/*][Google Search]]`'. In an Org-mode buffer, this command would look like '`Google Search`', but really it would remove your home directory. If you wish, you can make it easier to respond to the query by setting `org-confirm-shell-links` to `y-or-n-p`. Then a single *`y`* keypress will be enough to confirm shell links. It is also possible to turn off this check entirely, but I do not recommend to do this. Be warned.
7. **All these stars are driving me mad, I just find the Emacs outlines unreadable. Can't you just put white space and a single star as a starter for headlines?**
   See Section 10.5 [Clean view], page 44.
8. **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.

9. **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; ...
   ```

10. **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.

11. **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 10.8 [Interaction], page 48.

12. **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 '`#+TBLFM:`' 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!

13. **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⟩. Or configure the variable `org-table-tab-jumps-over-hlines`.

14. **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⟩.

15. **Is it possible to include entries from org-mode files into my emacs diary?**
    Since the org-mode agenda is much more powerful and can contain the diary (see Section 8.3.3 [Calendar/Diary integration], page 33), you should think twice before deciding to do this. Integrating Org-mode information into the diary is, however, possible. 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. Under XEmacs, it is not possible to jump back from the diary to the org, this works only in the agenda buffer.

10.8 Interaction with other packages

Org-mode can cooperate with the following packages:

'`org-mouse.el`' by Piotr Zielinski
: This package implements extended mouse functionality for Org-mode. It allows you to cycle visibility and to edit the document structure with the mouse. Best of all, it provides a context-sensitive menu on ⟨mouse-3⟩ that changes depending on the context of a mouse-click. Use a search engine to find this package on the web.

'`table.el`' by Takaaki Ota
: Org mode cooperates with table.el, see Section 3.5 [table.el], page 15. '`table.el`' is part of Emacs 22.

'`calc.el`' by Dave Gillespie
: Org-mode uses the calc package for implementing spreadsheet functionality in its tables (see Section 3.3 [Table calculations], page 11). 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 3.3 [Table calculations], page 11), 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-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`.

'`windmove.el`' by Hovav Shacham
: Also this package uses the *S-<cursor>* keys, so everything written in the paragraph above about CUA mode also applies here.

'`remember.el`' by John Wiegley
: Org mode cooperates with remember, see Section 4.7 [Remember], page 21. '`Remember.el`' is not part of Emacs, find it on the web.

10.9 Bugs

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

- If a table field starts with a link, and if the corresponding table column is narrowed (see Section 3.2 [Narrow columns], page 10) to a width too small to display the link, the field would look entirely empty even though it is not. To prevent this, Org-mode throws an error. The work-around is to make the column wide enough to fit the link, or to add some text (at least 2 characters) before the link in the same field.
- Narrowing table columns does not work on XEmacs, because the `format` function does not transport text properties.
- Text in an entry protected with the '`QUOTE`' keyword should not autowrap.
- 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.
- Plain list items should be able to hold a TODO item. Unfortunately this has so many technical problems that I will only consider this change for the next major release (5.0).
- The remote-editing commands in the agenda buffer cannot be undone with `undo` called from within the agenda buffer. But you can go to the corresponding buffer (using ⟨TAB⟩ or ⟨RET⟩ and execute `undo` there.
- Recalculating a table line applies the formulas from left to right. If a formula uses *calculated* fields further down the row, multiple recalculation may be needed to get all fields consistent.
- You can only make a single word boldface or italic. To emphasize several words in a row, each must have the emphasize markers, like in '`*three* *bold* *words*`'.
- The exporters work well, but could be made more efficient.

10.10 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 (in

alphabetic order) have helped the development along with ideas, suggestions and patches. Many thanks to all of you, Org-mode would not be what it is without your input.

- Thomas Baumann contributed the code for links to the MH-E email system.
- Alex Bochannek provided a patch for rounding time stamps.
- Charles Caves' suggestion sparked the implementation of templates for Remember.
- Pavel Chalmoviansky influenced the agenda treatment of items with specified time.
- Gregory Chenov patched support for lisp forms into table calculations and improved XEmacs compatibility.
- Sacha Chua suggested to copy some linking code from Planner.
- Kees Dullemond inspired the use of narrowed tabled columns.
- Christian Egli converted the documentation into TeXInfo format, patched CSS formatting into the HTML exporter, and inspired the agenda.
- Nic Ferrier contributed mailcap and XOXO support.
- Kai Grossjohann pointed out key-binding conflicts caused by Org-mode.
- Stefan Monnier provided a patch to keep the Emacs-Lisp compiler happy.
- Tim O'Callaghan suggested in-file links, search options for general file links, and TAGS.
- Oliver Oppitz suggested multi-state TODO items.
- Scott Otterson sparked the introduction of descriptive text for links, among other things.
- Pete Phillips helped the development of the TAGS feature.
- T.V. Raman reported bugs and suggested improvements.
- Matthias Rempe (Oelde) provided ideas, Windows support, and quality control.
- 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`'.
- Philip Rooke created the Org-mode reference card and provided lots of feedback.
- Christian Schlauer proposed angular brackets around links, among other things.
- Linking to VM/BBDB/GNUS was inspired by Tom Shannon's '`organizer-mode.el`'.
- David O'Toole wrote '`org-publish.el`' and came up with lots is ideas for small changes.
- Jürgen Vollmer contributed code generating the table of contents in HTML output.
- Chris Wallace provided a patch implementing the '`QUOTE`' keyword.
- David Wainberg suggested archiving, and improvements to the linking system.
- John Wiegley wrote '`emacs-wiki.el`' and '`planner.el`'. The development of Org-mode was fully independent, and both systems are really different beasts in their basic ideas and implementation details. However, I have later looked at John's code and learned from his implementation of (i) links where the link itself is hidden and only a description is shown, and (ii) popping up a calendar to select a date.
- Carsten Wimmer suggested some changes and helped fix a bug in linking to GNUS.
- Roland Winkler requested additional keybindings to make Org-mode work on a tty.
- Piotr Zielinski wrote '`org-mouse.el`' and showed how to follow links with mouse-1.

11 Index

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

12 Key Index