

Org Mode マニュアル

リリース 7.5

by Carsten Dominik

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

このマニュアルは、Org-mode 7.5 に対応しています。

Copyright © 2004, 2005, 2006, 2007, 2008, 2009, 2010 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being “A GNU Manual,” and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License.”

(a) The FSF’s Back-Cover Text is: “You have the freedom to copy and modify this GNU manual. Buying copies from the FSF supports it in developing GNU and promoting software freedom.”

This document is part of a collection distributed under the GNU Free Documentation License. If you want to distribute this document separately from the collection, you can do so by adding a copy of the license to the document, as described in section 6 of the license.

Table of Contents

1	Introduction	1
1.1	Summary	1
1.2	Installation	3
1.3	Activation	4
1.4	Feedback	4
1.5	本マニュアルで使われる植字の置き換え	5
2	ドキュメントの構造	7
2.1	Outlines	7
2.2	Headlines	7
2.3	Visibility cycling	7
2.4	Motion	9
2.5	Structure editing	9
2.6	Sparse trees	12
2.7	Plain lists	13
2.8	Drawers	16
2.9	Blocks	16
2.10	Footnotes	16
2.11	The Orgstruct minor mode	18
3	Tables	19
3.1	The built-in table editor	19
3.2	Column width and alignment	22
3.3	Column groups	23
3.4	The Orgtbl minor mode	23
3.5	The spreadsheet	24
3.5.1	References	24
3.5.2	Formula syntax for Calc	26
3.5.3	Emacs Lisp forms as formulas	27
3.5.4	Field and range formulas	28
3.5.5	Column formulas	28
3.5.6	Editing and debugging formulas	29
3.5.7	Updating the table	30
3.5.8	Advanced features	31
3.6	Org-Plot	32
4	Hyperlinks	34
4.1	Link format	34
4.2	Internal links	34
4.2.1	Radio targets	35
4.3	External links	35
4.4	Handling links	36

4.5	Using links outside Org	39
4.6	Link abbreviations	39
4.7	ファイルリンクにおける検索オプション	40
4.8	カスタム検索	41
5	TODO アイテム	42
5.1	基本的な TODO の機能	42
5.2	TODO キーワードの拡張的な使い方	43
5.2.1	ワークフローの状態としての TODO キーワード	43
5.2.2	種類としての TODO キーワード	43
5.2.3	同一ファイル内での複数のキーワードセット	44
5.2.4	Fast access to TODO states	44
5.2.5	ファイル別にキーワードを設定する	45
5.2.6	Faces for TODO keywords	45
5.2.7	TODO dependencies	46
5.3	Progress logging	47
5.3.1	Closing items	47
5.3.2	Tracking TODO state changes	47
5.3.3	習慣の追跡	48
5.4	Priorities	50
5.5	Breaking tasks down into subtasks	51
5.6	Checkboxes	51
6	Tags	54
6.1	Tag inheritance	54
6.2	Setting tags	54
6.3	Tag searches	56
7	プロパティ (属性) とカラム (列)	58
7.1	Property syntax	58
7.2	Special properties	59
7.3	Property searches	60
7.4	Property Inheritance	60
7.5	Column view	61
7.5.1	Defining columns	61
7.5.1.1	Scope of column definitions	61
7.5.1.2	Column attributes	61
7.5.2	Using column view	63
7.5.3	カラム表示の保存	64
7.6	The Property API	65

8	日付と時刻	66
8.1	Timestamps, deadlines, and scheduling	66
8.2	Creating timestamps	67
8.2.1	The date/time prompt	68
8.2.2	Custom time format	69
8.3	Deadlines and scheduling	70
8.3.1	Inserting deadlines or schedules	71
8.3.2	Repeated tasks	71
8.4	Clocking work time	73
8.4.1	Clocking commands	73
8.4.2	The clock table	74
8.4.3	Resolving idle time	76
8.5	Effort estimates	77
8.6	Taking notes with a relative timer	78
8.7	カウントダウンタイマ	79
9	Capture - Refile - Archive	80
9.1	Capture	80
9.1.1	Setting up capture	80
9.1.2	Using capture	80
9.1.3	Capture templates	81
9.1.3.1	Template elements	82
9.1.3.2	テンプレートの拡張	84
9.2	Attachments	85
9.3	RSS フィード	86
9.4	Protocols for external access	87
9.5	Refilng notes	87
9.6	Archiving	88
9.6.1	Moving a tree to the archive file	88
9.6.2	ファイル内部でのアーカイブ	88
10	アジェンダビュー	90
10.1	Agenda files	90
10.2	アジェンダのコマンド選択画面	91
10.3	agenda に組み込まれているビュー	92
10.3.1	1 週間／1 日のアジェンダ	92
10.3.2	The global TODO list	94
10.3.3	Matching tags and properties	95
10.3.4	Timeline for a single file	97
10.3.5	Search view	98
10.3.6	Stuck projects	98
10.4	Presentation and sorting	99
10.4.1	Categories	99
10.4.2	Time-of-day specifications	100
10.4.3	agenda の項目をソートする	100
10.5	Commands in the agenda buffer	101
10.6	Custom agenda views	110

10.6.1	Storing searches	110
10.6.2	Block agenda	111
10.6.3	Setting options for custom commands	111
10.7	Exporting Agenda Views	113
10.8	Using column view in the agenda	115
11	Markup for rich export	116
11.1	Structural markup elements	116
11.2	画像と表	118
11.3	Literal examples	118
11.4	Include files	120
11.5	Index entries	120
11.6	Macro replacement	120
11.7	Embedded L ^A T _E X	121
11.7.1	Special symbols	121
11.7.2	Subscripts and superscripts	121
11.7.3	L ^A T _E X の断片的なコード	122
11.7.4	Previewing LaTeX fragments	123
11.7.5	CDL ^A T _E X を数学の入力に使う	123
12	Exporting	125
12.1	Selective export	125
12.2	Export options	125
12.3	The export dispatcher	127
12.4	ASCII/Latin-1/UTF-8 export	127
12.5	HTML export	128
12.5.1	HTML エクスポートのコマンド	128
12.5.2	Quoting HTML tags	129
12.5.3	Links in HTML export	129
12.5.4	Tables	130
12.5.5	Images in HTML export	130
12.5.6	Math formatting in HTML export	130
12.5.7	Text areas in HTML export	131
12.5.8	CSS support	131
12.5.9	ウェブページの表示に関する JavaScript のサポート	132
12.6	L ^A T _E X と PDF のエクスポート	133
12.6.1	L ^A T _E X エクスポートのコマンド	133
12.6.2	見出しと構造の分割	134
12.6.3	L ^A T _E X コードの引用	135
12.6.4	L ^A T _E X エクスポートにおける表	135
12.6.5	L ^A T _E X エクスポートにおける画像	135
12.6.6	Beamer クラスのエクスポート	136
12.7	DocBook export	138
12.7.1	DocBook export commands	138
12.7.2	Quoting DocBook code	138
12.7.3	Recursive sections	139
12.7.4	Tables in DocBook export	139
12.7.5	Images in DocBook export	139

12.7.6	DocBook 出力における特殊文字	140
12.8	TaskJuggler export	140
12.8.1	TaskJuggler export commands	140
12.8.2	Tasks	141
12.8.3	Resources	141
12.8.4	Export of properties	141
12.8.5	Dependencies	141
12.8.6	Reports	142
12.9	Freemind export	142
12.10	XOXO export	142
12.11	iCalendar エクスポート	143
13	Publishing	144
13.1	Configuration	144
13.1.1	The variable <code>org-publish-project-alist</code>	144
13.1.2	Sources and destinations for files	144
13.1.3	Selecting files	145
13.1.4	Publishing action	145
13.1.5	Options for the HTML/L ^A T _E X exporters	146
13.1.6	Links between published files	147
13.1.7	Generating a sitemap	147
13.1.8	Generating an index	148
13.2	Uploading files	148
13.3	Sample configuration	149
13.3.1	Example: simple publishing configuration	149
13.3.2	Example: complex publishing configuration	149
13.4	公開の開始	150
14	ソースコードとの連携	152
14.1	Structure of code blocks	152
14.2	Editing source code	153
14.3	Exporting code blocks	153
14.4	Extracting source code	154
14.5	Evaluating code blocks	154
14.6	Library of Babel	155
14.7	Languages	156
14.8	Header arguments	156
14.8.1	Using header arguments	157
14.8.2	Specific header arguments	159
14.8.2.1	<code>:var</code>	159
14.8.2.2	<code>:results</code>	162
14.8.2.3	<code>:file</code>	164
14.8.2.4	<code>:dir</code> and remote execution	164
14.8.2.5	<code>:exports</code>	165
14.8.2.6	<code>:tangle</code>	165
14.8.2.7	<code>:mkdirp</code>	165
14.8.2.8	<code>:comments</code>	166
14.8.2.9	<code>:no-expand</code>	166

14.8.2.10	<code>:session</code>	166
14.8.2.11	<code>:noweb</code>	166
14.8.2.12	<code>:cache</code>	167
14.8.2.13	<code>:sep</code>	168
14.8.2.14	<code>:hlines</code>	168
14.8.2.15	<code>:colnames</code>	169
14.8.2.16	<code>:rownames</code>	169
14.8.2.17	<code>:shebang</code>	170
14.8.2.18	<code>:eval</code>	170
14.9	Results of evaluation	170
14.9.1	Non-session	170
14.9.1.1	<code>:results value</code>	170
14.9.1.2	<code>:results output</code>	171
14.9.2	Session	171
14.9.2.1	<code>:results value</code>	171
14.9.2.2	<code>:results output</code>	171
14.10	Noweb reference syntax	171
14.11	Key bindings and useful functions	172
14.12	バッチ処理	172
15	Miscellaneous	174
15.1	Completion	174
15.2	Easy Templates	174
15.3	Speed keys	175
15.4	コードの評価とセキュリティの問題	175
15.5	Customization	176
15.6	バッファ中での設定の要約	176
15.7	The very busy C-c C-c key	180
15.8	より見やすいアウトラインビュー	180
15.9	Org-mode を tty 端末で使う	182
15.10	他のパッケージとの関係	183
15.10.1	Packages that Org cooperates with	183
15.10.2	Packages that lead to conflicts with Org-mode	184
Appendix A	Hacking	186
A.1	Hooks	186
A.2	Add-on packages	186
A.3	Adding hyperlink types	186
A.4	Context-sensitive commands	188
A.5	任意のシンタックスによる表やリスト	188
A.5.1	Radio tables	188
A.5.2	A L ^A T _E X example of radio tables	189
A.5.3	Translator functions	191
A.5.4	ラジオリスト	192
A.6	Dynamic blocks	192
A.7	Special agenda views	193
A.8	Extracting agenda information	195
A.9	Using the property API	196

A.10 マッピング API を使う	198
Appendix B MobileOrg	200
B.1 Setting up the staging area	200
B.2 Pushing to MobileOrg	200
B.3 MobileOrg から pull する	201
Appendix C History and acknowledgments	
.....	203
Concept index	207
Key index	215
Command and function index	220
Variable index	224

1 Introduction

1.1 Summary

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

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

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

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

表計算ソフトと互換性のある Org-mode の自動的で文脈依存な表編集機能は、マイナーモードの Orgtbl を動かすことで、どのようなメジャーモードにも組み込むことができます。表を変換することで、たとえば \LaTeX の表のように、任意のファイルタイプで表を維持することができます。構造編集とリスト生成の機能は、マイナーモードの Orgstruct によって、Org-mode の外部で利用することができます。

Org-mode は、単純なものは単純なまま保持します。初めて起動した Org-mode は、わかりやすく、簡単に使えるアウトライナーのように感じるはずですが、Org-mode に複雑さはありませんが、数多くの機能が必要とする時に使えます。Org-mode はツールボックスであり、様々な方法で、そして様々な目的で利用できます。例えば、

- 視覚的に表示が循環し、構造を編集するように拡張されたアウトライナー
- 構造化されたノートを取るためのアスキーシステムと表編集機能
- TODO リストの編集機能
- 締切日とスケジュールを含む完全なアジェンダと予定表
- Devid Allen 氏の GTD システムを実行するための環境
- シンプルなハイパーテキストシステム (HTML と \LaTeX エクスポートを含む)
- 内部リンクで構成されたウェブページ群を生成するための公開ツール
- 文芸的プログラミングのための環境

最新の Org-mode へのリンクを提供する Org-mode のためのウェブページがあります。追加情報や FAQ、チュートリアルなどがあります。<http://orgmode.org> で公開されています。

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

1.2 Installation

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

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

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

もし `‘contrib’` サブディレクトリのコードを使うのならば、このディレクトリについても同様のステップを実行します。

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

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

```
make
```

If you are running *Org* from the distribution directory, this is これですべてです。もし *Org-mode* をシステムディレクトリにインストールしたい場合には、管理者権限で次のコマンドを使います。

```
make install
```

INFO ファイルのインストールは、システムに依存します。これは、`‘install-info’` プログラムにおける違いに原因があります。Debian であれば、*INFO* ファイルはカレントディレクトリにインストールされ、*INFO directory* ファイルを変更します。その他の多くのシステムでは、それぞれのファイルは、別々に正しいディレクトリにコピーされる必要があります。そして、`‘install-info’` がディレクトリファイルだけを修正します。次のコマンドのうち必要なものをシステムの文書を参考に調べてください。

```
make install-info
```

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

次に、以下の一行を `‘.emacs’` に追加します。これは Emacs が、*Org-mode* が開始する時には即座に読み込まれないファイルに含まれる関数を自動的に読み込めるようにするために必要です。

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

次節の解説ように *Org-mode* をアクティベーションすることを忘れないでください。

1.3 Activation

拡張子が`‘.org’`のファイルがOrg-modeを利用することを確実にするために、次の行を`‘.emacs’`に追加します。

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

Org mode buffers need font-lock to be turned on - this is the これは Emacs で標準の設定です¹。

Org-mode の 4 つのコマンド (`org-store-link`, `org-capture`, `org-agenda`, `org-iswitchb`) は、グローバルキーを割り当てて使いやすくするべきでしょう。言い換えれば、Org-mode のバッファだけではなく、Emcas でいつでも使えるようにします。これらのキーバインドとして以下を割り当てることをお勧めします。自分の環境に応じて適当にキーを変更してください。

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

この設定をすると、拡張子が`‘.org’`のファイルは、Org-mode に設定されます。別の方法として、ファイルの一行目に次のような一文を追加することでも、Org-mode に設定できます。

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

which will select Org-mode for this buffer no matter what (nonintent 問題箇所) 変数 `org-insert-mode-line-in-empty-file` も確認してください。

Org-mode の多くのコマンドは、リージョンが *active* な場合に動作します。アクティブなリージョンをハイライトするためには、`transient-mark-mode` (XEmacs では `zmacs-regions`) を有効にする必要があります。Emacs23 では標準で有効になっていますが、Emacs22 では次のコマンドを使う必要があります。

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

If you do not like `transient-mark-mode`, you can create an (nointent 問題箇所) リージョンを選択するためにマウスを利用できます。もしくは、カーソルを移動する前に `C-SPC` を二回押します。

1.4 Feedback

Org-mode で問題を発見した場合、あるいは質問や意見、アイディアがある場合には、Org-mode のメーリングリスト `emacs-orgmode@gnu.org` へメールしてください。あなたがメーリングリストのメンバーでないと、メールは管理者が承認した後にメーリングリストへ転送されます²。

バグをレポートする時は、まず始めに最新バージョンの Org-mode を利用して該当のバグが再現されるか試してください。古いバージョンを利用している場合、すでにそのバグが修正されている可能性が高いです。バグの再現性が確認できたならば、レポートを準備して可能な限り多くの情報を提供してください。具体的には、Emacs のバージョン情報 (`M-x emacs-version RET`) と Org-mode のバージョン情報 (`M-x org-version RET`)、また、Org-mode に関連する `‘.emacs’` の設定をバグレポートに記載してください。このようなバグレポートの形式を守るための最も簡単な方法は、次のコマンドを利用します。

¹ もしグローバルに `font-lock` を使わない場合は、(`add-hook 'org-mode-hook 'turn-on-font-lock`) を使い Org-mode のバッファについて有効化してください。

² メーリングリストの管理者の仕事量を最小化するために、ぜひメーリングリストの購読を検討してください。

M-x org-submit-bug-report

which will put all this information into an Emacs mail buffer so (nointent 問題箇所) あなたがバグについての説明を書き加えればよいだけの状態。Emacs を利用してメールを送信しない場合は、テンプレートの内容をメールクライアントにコピー&ペーストしてください。

もしもエラーが発生したら、バックトレースが非常に便利です (どのように作るのかは以下を参照してください)。しばしば次の明瞭な情報を含む小さな凡例ファイルが手助けになります。

1. 正確に何を実行したのか
2. 何が起きることを期待していたのか
3. 期待と異なり何が起こったのか

Thank you for helping to improve this program.

便利なバックトレースを生成する方法

Org-mode を利用している時に、理解できないメッセージのエラーが発生したら、バグを発見した可能性があります。エラーを報告する最良の方法は、すでに説明したバグレポートの書式に加えて、*backtrace* を提供することです。バックトレースは、ビルトインされたデバッガーによるエラーの発生箇所とどのように発生したかについての情報です。以下に、有用なバックトレースを生成する手順を示します。

1. コンパイルされていない Org-mode の Lisp ファイル群を再読み込みする。バックトレースは、コンパイルされていないコードを利用して生成したバックトレースは、より多くの情報を含みます。これを実行するためには、次のコマンドを実行します。

C-u M-x org-reload RET

もしくは、Org -> Refresh/Reload -> Reload Org uncompiled をメニューから選択します。

2. オプションメニュー Options から、Enter Debugger on Error (XEmacs では、このオプションは Troubleshooting サブメニューにあります) を選択する。
3. エラーを再現するために必要な操作を行なってください。実行した操作を忘れずにメモしておいてください。
4. エラーが再現されると、***Backtrace*** バッファが画面上に表示されます。このバッファを別のファイルとして保存し (例えば C-x C-w を使って)、バグレポートに添付します。

1.5 本マニュアルで使われる植字の置き換え

Org-mode は、3 種類のキーワードを使います。TODO キーワード、タグ、プロパティです。このマニュアルでは次のように植字を使い分けます。

TODO

WAITING TODO キーワードは、すべて大文字で記述されます。ユーザが定義する場合も同様です。

boss

ARCHIVE ユーザ定義のタグは、小文字で記述されます。特別な意味を持つビルトインされたタグは、すべて大文字で記述されます。

Release

PRIORITY ユーザ定義のプロパティは、大文字で始めて他を小文字で記述されます。特別な意味を持つビルトインされたプロパティは、すべて大文字で記述されます。

このマニュアルでは、Org-mode の機能を利用するためのキーバインドと、対応するコマンドの両方を表記します。Org-mode は、しばしば異なる関数に対して同じキーバインドを使います (これ

はコンテキストに依存しています)。そのようなキーバインドが割り振られたコマンドには、`org-metaright`のような一般的な名称があります。このマニュアルでは、可能な限り一般的なコマンドを用いて内部的に呼び出される関数の名称を提示します。例えば、ドキュメントの構造についての章では、`M-right`は`org-do-demote`を呼び出すように表記します。一方で、テーブルについての章では、`org-table-move-column-right`を呼び出すように表記します。

もし望むならば、`'org.texi'`にある `cmdnames` フラグの設定を外すことで、コマンドの名称を表示しないようにマニュアルをコンパイルすることができます。

2 ドキュメントの構造

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

2.1 Outlines

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

2.2 Headlines

Headlines define the structure of an outline tree. The headlines in Org start with one or more stars, on the left margin¹. For example:

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

* Another top level headline
```

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

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

2.3 Visibility cycling

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

<code>TAB</code>	<i>Subtree cycling:</i> Rotate current subtree among the states <code>, -> FOLDED -> CHILDREN -> SUBTREE --.</code> <code>'-----'</code>	<code>org-cycle</code>
------------------	---	------------------------

The cursor must be on a headline for this to work². When the cursor is at the beginning of the buffer and the first line is not a headline, then `TAB` actually

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

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

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

S-TAB org-global-cycle

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

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

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

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

Show all, including drawers.

C-c C-r org-reveal

Reveal context around point, showing the current entry, the following heading and the hierarchy above. Useful for working near a location that has been exposed by a sparse tree command (see Section 2.6 [Sparse trees], page 12) or an agenda command (see Section 10.5 [Agenda commands], page 101). With a prefix argument *show*, on each level, all sibling headings. With double prefix *arg*, also show the entire subtree of the parent.

C-c C-k show-branches

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

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

Show the current subtree in an indirect buffer⁴. With a numeric prefix argument *N*, go up to level *N* and then take that tree. If *N* is negative then go up that many levels. With a *C-u* prefix, do not remove the previously used indirect buffer.

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

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

Furthermore, any entries with a ‘VISIBILITY’ property (see Chapter 7 [Properties and Columns], page 58) will get their visibility adapted accordingly. Allowed values for this property are `folded`, `children`, `content`, and `all`.

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

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

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

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

2.4 Motion

The following commands jump to other headlines in the buffer.

<code>C-c C-n</code>		<code>outline-next-visible-heading</code>
	Next heading.	
<code>C-c C-p</code>		<code>outline-previous-visible-heading</code>
	Previous heading.	
<code>C-c C-f</code>		<code>org-forward-same-level</code>
	Next heading same level.	
<code>C-c C-b</code>		<code>org-backward-same-level</code>
	Previous heading same level.	
<code>C-c C-u</code>		<code>outline-up-heading</code>
	Backward to higher level heading.	
<code>C-c C-j</code>		<code>org-goto</code>
	Jump to a different place without changing the current outline visibility. Shows the document structure in a temporary buffer, where you can use the following keys to find your destination:	
	<code>TAB</code>	Cycle visibility.
	<code>down / up</code>	Next/previous visible headline.
	<code>RET</code>	Select this location.
	<code>/</code>	Do a Sparse-tree search
	The following keys work if you turn off <code>org-goto-auto-isearch</code>	
	<code>n / p</code>	Next/previous visible headline.
	<code>f / b</code>	Next/previous headline same level.
	<code>u</code>	One level up.
	<code>0-9</code>	Digit argument.
	<code>q</code>	Quit
	See also the variable <code>org-goto-interface</code> .	

2.5 Structure editing

<code>M-RET</code>	<code>org-insert-heading</code>
Insert new heading with same level as current. If the cursor is in a plain list item, a new item is created (see Section 2.7 [Plain lists], page 13). To force creation of a new headline, use a prefix argument. When this command is used in the middle of a line, the line is split and the rest of the line becomes the new headline ⁵ . If the command is used at the beginning of a headline, the new headline is created before the current line. If at the beginning of any other line, the content of that line is made the new heading. If the command is used at the end of a folded subtree (i.e. behind the ellipses at the end of a headline), then a headline like the current one will be inserted after the end of the subtree.	

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

<i>C-RET</i>	<code>org-insert-heading-respect-content</code> Just like <i>M-RET</i> , except when adding a new heading below the current heading, the new heading is placed after the body instead of before it. This command works from anywhere in the entry.
<i>M-S-RET</i>	<code>org-insert-todo-heading</code> Insert new TODO entry with same level as current heading. See also the variable <code>org-treat-insert-todo-heading-as-state-change</code> .
<i>C-S-RET</i>	<code>org-insert-todo-heading-respect-content</code> Insert new TODO entry with same level as current heading. Like <i>C-RET</i> , the new headline will be inserted after the current subtree.
<i>TAB</i>	<code>org-cycle</code> In a new entry with no text yet, the first <i>TAB</i> demotes the entry to become a child of the previous one. The next <i>TAB</i> makes it a parent, and so on, all the way to top level. Yet another <i>TAB</i> , and you are back to the initial level.
<i>M-left</i>	<code>org-do-promote</code> Promote current heading by one level.
<i>M-right</i>	<code>org-do-demote</code> Demote current heading by one level.
<i>M-S-left</i>	<code>org-promote-subtree</code> Promote the current subtree by one level.
<i>M-S-right</i>	<code>org-demote-subtree</code> Demote the current subtree by one level.
<i>M-S-up</i>	<code>org-move-subtree-up</code> Move subtree up (swap with previous subtree of same level).
<i>M-S-down</i>	<code>org-move-subtree-down</code> Move subtree down (swap with next subtree of same level).
<i>C-c C-x C-w</i>	<code>org-cut-subtree</code> Kill subtree, i.e. remove it from buffer but save in kill ring. With a numeric prefix argument N, kill N sequential subtrees.
<i>C-c C-x M-w</i>	<code>org-copy-subtree</code> Copy subtree to kill ring. With a numeric prefix argument N, copy the N sequential subtrees.
<i>C-c C-x C-y</i>	<code>org-paste-subtree</code> Yank subtree from kill ring. This does modify the level of the subtree to make sure the tree fits in nicely at the yank position. The yank level can also be specified with a numeric prefix argument, or by yanking after a headline marker like <code>****</code> .
<i>C-y</i>	<code>org-yank</code> Depending on the variables <code>org-yank-adjusted-subtrees</code> and <code>org-yank-folded-subtrees</code> , Org's internal <code>yank</code> command will paste subtrees folded and in a clever way, using the same command as <i>C-c C-x C-y</i> . With the

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

C-c C-x c	org-clone-subtree-with-time-shift
Clone a subtree by making a number of sibling copies of it. You will be prompted for the number of copies to make, and you can also specify if any timestamps in the entry should be shifted. This can be useful, for example, to create a number of tasks related to a series of lectures to prepare. For more details, see the docstring of the command org-clone-subtree-with-time-shift .	
C-c C-w	org-refile
Refile entry or region to a different location. See Section 9.5 [Refiling notes], page 87.	
C-c ^	org-sort-entries-or-items
Sort same-level entries. When there is an active region, all entries in the region will be sorted. Otherwise the children of the current headline are sorted. The command prompts for the sorting method, which can be alphabetically, numerically, by time (first timestamp with active preferred, creation time, scheduled time, deadline time), by priority, by TODO keyword (in the sequence the keywords have been defined in the setup) or by the value of a property. Reverse sorting is possible as well. You can also supply your own function to extract the sorting key. With a C-u prefix, sorting will be case-sensitive. With two C-u prefixes, duplicate entries will also be removed.	
C-x n s	org-narrow-to-subtree
Narrow buffer to current subtree.	
C-x n b	org-narrow-to-block
Narrow buffer to current block.	
C-x n w	widen
Widen buffer to remove narrowing.	
C-c *	org-toggle-heading
Turn a normal line or plain list item into a headline (so that it becomes a subheading at its location). Also turn a headline into a normal line by removing the stars. If there is an active region, turn all lines in the region into headlines. If the first line in the region was an item, turn only the item lines into headlines. Finally, if the first line is a headline, remove the stars from all headlines in the region.	

When there is an active region (Transient Mark mode), promotion and demotion work on all headlines in the region. To select a region of headlines, it is best to place both point and mark at the beginning of a line, mark at the beginning of the first headline, and point at the line just after the last headline to change. Note that when the cursor is inside a table (see Chapter 3 [Tables], page 19), the Meta-Cursor keys have different functionality.

2.6 Sparse trees

An important feature of Org-mode is the ability to construct *sparse trees* for selected information in an outline tree, so that the entire document is folded as much as possible, but the selected information is made visible along with the headline structure above it⁶. Just try it out and you will see immediately how it works.

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

`C-c /` `org-sparse-tree`

This prompts for an extra key to select a sparse-tree creating command.

`C-c / r` `org-occur`

Occur. Prompts for a regexp and shows a sparse tree with all matches. If the match is in a headline, the headline is made visible. If the match is in the body of an entry, headline and body are made visible. In order to provide minimal context, also the full hierarchy of headlines above the match is shown, as well as the headline following the match. Each match is also highlighted; the highlights disappear when the buffer is changed by an editing command⁷, or by pressing `C-c C-c`. When called with a `C-u` prefix argument, previous highlights are kept, so several calls to this command can be stacked.

`M-g n` or `M-g M-n` `next-error`

Jump to the next sparse tree match in this buffer.

`M-g p` or `M-g M-p` `previous-error`

Jump to the previous sparse tree match in this buffer.

For frequently used sparse trees of specific search strings, you can use the variable `org-agenda-custom-commands` to define fast keyboard access to specific sparse trees. These commands will then be accessible through the agenda dispatcher (see Section 10.2 [Agenda dispatcher], page 91). For example:

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

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

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

To print a sparse tree, you can use the Emacs command `ps-print-buffer-with-faces` which does not print invisible parts of the document⁸. Or you can use the command `C-c C-e v` to export only the visible part of the document and print the resulting file.

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

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

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

2.7 Plain lists

Within an entry of the outline tree, hand-formatted lists can provide additional structure. They also provide a way to create lists of checkboxes (see Section 5.6 [Checkboxes], page 51). Org supports editing such lists, and every exporter (see Chapter 12 [Exporting], page 125) can parse and format them.

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

- *Unordered* list items start with ‘-’, ‘+’, or ‘*’⁹ as bullets.
- *Ordered* list items start with a numeral followed by either a period or a right parenthesis¹⁰, such as ‘1.’ or ‘1)’¹¹. If you want a list to start with a different value (e.g. 20), start the text of the item with [20]¹². Those constructs can be used in any item of the list in order to enforce a particular numbering.
- *Description* list items are unordered list items, and contain the separator ‘::’ to distinguish the description *term* from the description.

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

Two methods¹³ are provided to terminate lists. A list ends whenever every item has ended, which means before any line less or equally indented than items at top level. It also ends before two blank lines¹⁴. In that case, all items are closed. For finer control, you can end lists with any pattern set in `org-list-end-regexp`. Here is an example:

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

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

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

¹² If there’s a checkbox in the item, the cookie must be put *before* the checkbox. If you have activated alphabetical lists, you can also use counters like [2b].

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

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

```

** Lord of the Rings
My favorite scenes are (in this order)
1. The attack of the Rohirrim
2. Eowyn's fight with the witch king
  + this was already my favorite scene in the book
  + I really like Miranda Otto.
3. Peter Jackson being shot by Legolas
  - on DVD only
  He makes a really funny face when it happens.
But in the end, no individual scenes matter but the film as a whole.
Important actors in this film are:
- Elijah Wood :: He plays Frodo
- Sean Austin :: He plays Sam, Frodo's friend. I still remember
  him very well from his role as Mikey Walsh in The Goonies.

```

Org supports these lists by tuning filling and wrapping commands to deal with them correctly¹⁵, and by exporting them properly (see Chapter 12 [Exporting], page 125). Since indentation is what governs the structure of these lists, many structural constructs like `#+BEGIN_...` blocks can be indented to signal that they belong to a particular item.

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

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

TAB `org-cycle`

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

M-RET `org-insert-heading`

Insert new item at current level. With a prefix argument, force a new heading (see Section 2.5 [Structure editing], page 9). If this command is used in the middle of an item, that item is *split* in two, and the second part becomes the new item¹⁶. If this command is executed *before item's body*, the new item is created *before* the current one.

M-S-RET Insert a new item with a checkbox (see Section 5.6 [Checkboxes], page 51).

TAB `org-cycle`

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

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

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

- S-up**
S-down Jump to the previous/next item in the current list, but only if `org-support-shift-select` is off. If not, you can still use paragraph jumping commands like `C-up` and `C-down` to quite similar effect.
- M-S-up**
M-S-down Move the item including subitems up/down (swap with previous/next item of same indentation). If the list is ordered, renumbering is automatic.
- M-left**
M-right Decrease/increase the indentation of an item, leaving children alone.
- M-S-left**
M-S-right Decrease/increase the indentation of the item, including subitems. Initially, the item tree is selected based on current indentation. When these commands are executed several times in direct succession, the initially selected region is used, even if the new indentation would imply a different hierarchy. To use the new hierarchy, break the command chain with a cursor motion or so.
- As a special case, using this command on the very first item of a list will move the whole list. This behavior can be disabled by configuring `org-list-automatic-rules`. The global indentation of a list has no influence on the text *after* the list.
- C-c C-c** If there is a checkbox (see Section 5.6 [Checkboxes], page 51) in the item line, toggle the state of the checkbox. In any case, verify bullets and indentation consistency in the whole list.
- C-c -** Cycle the entire list level through the different itemize/enumerate bullets ('-', '+', '*', '1.', '1') or a subset of them, depending on `org-plain-list-ordered-item-terminator`, the type of list, and its position¹⁷. With a numeric prefix argument N, select the Nth bullet from this list. If there is an active region when calling this, selected text will be changed into an item. With a prefix argument, all lines will be converted to list items. If the first line already was a list item, any item marker will be removed from the list. Finally, even without an active region, a normal line will be converted into a list item.
- C-c *** Turn a plain list item into a headline (so that it becomes a subheading at its location). See Section 2.5 [Structure editing], page 9, for a detailed explanation.
- C-c C-*** Turn the whole plain list into a subtree of the current heading. Checkboxes (see Section 5.6 [Checkboxes], page 51) will become TODO (resp. DONE) keywords when unchecked (resp. checked).
- S-left/right** This command also cycles bullet styles when the cursor is on the bullet or anywhere in an item line, details depending on `org-support-shift-select`.
- C-c ^** Sort the plain list. You will be prompted for the sorting method: numerically, alphabetically, by time, or by custom function.

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

2.8 Drawers

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

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

Visibility cycling (see Section 2.3 [Visibility cycling], page 7) on the headline will hide and show the entry, but keep the drawer collapsed to a single line. In order to look inside the drawer, you need to move the cursor to the drawer line and press TAB there. Org-mode uses the PROPERTIES drawer for storing properties (see Chapter 7 [Properties and Columns], page 58), and you can also arrange for state change notes (see Section 5.3.2 [Tracking TODO state changes], page 47) and clock times (see Section 8.4 [Clocking work time], page 73) to be stored in a drawer LOGBOOK. If you want to store a quick note in the LOGBOOK drawer, in a similar way to state changes, use

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

2.9 Blocks

Org-mode uses begin...end blocks for various purposes from including source code examples (see Section 11.3 [Literal examples], page 118) to capturing time logging information (see Section 8.4 [Clocking work time], page 73). These blocks can be folded and unfolded by pressing TAB in the begin line. You can also get all blocks folded at startup by configuring the variable `org-hide-block-startup` or on a per-file basis by using

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

2.10 Footnotes

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

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

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

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

compatibility, but not encouraged because of possible conflicts with \LaTeX snippets (see Section 11.7 [Embedded LaTeX], page 121). Here are the valid references:

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

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

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

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

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

The following command handles footnotes:

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

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

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

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

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

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

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

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

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

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

C-c C-o or *mouse-1/2*

Footnote labels are also links to the corresponding definition/reference, and you can use the usual commands to follow these links.

2.11 The Orgstruct minor mode

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

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

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

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

3 Tables

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

3.1 The built-in table editor

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

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

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

```
|Name|Phone|Age|
|–
```

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

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

Creation and conversion

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

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

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

Re-aligning and field motion

`C-c C-c` `org-table-align`

Re-align the table without moving the cursor.

<i><TAB></i>	<i>org-table-next-field</i>
Re-align the table, move to the next field. Creates a new row if necessary.	
<i>S-TAB</i>	<i>org-table-previous-field</i>
Re-align, move to previous field.	
<i>RET</i>	<i>org-table-next-row</i>
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.	
<i>M-a</i>	<i>org-table-beginning-of-field</i>
Move to beginning of the current table field, or on to the previous field.	
<i>M-e</i>	<i>org-table-end-of-field</i>
Move to end of the current table field, or on to the next field.	
Column and row editing	
<i>M-left</i>	<i>org-table-move-column-left</i>
<i>M-right</i>	<i>org-table-move-column-right</i>
Move the current column left/right.	
<i>M-S-left</i>	<i>org-table-delete-column</i>
Kill the current column.	
<i>M-S-right</i>	<i>org-table-insert-column</i>
Insert a new column to the left of the cursor position.	
<i>M-up</i>	<i>org-table-move-row-up</i>
<i>M-down</i>	<i>org-table-move-row-down</i>
Move the current row up/down.	
<i>M-S-up</i>	<i>org-table-kill-row</i>
Kill the current row or horizontal line.	
<i>M-S-down</i>	<i>org-table-insert-row</i>
Insert a new row above the current row. With a prefix argument, the line is created below the current one.	
<i>C-c -</i>	<i>org-table-insert-hline</i>
Insert a horizontal line below current row. With a prefix argument, the line is created above the current line.	
<i>C-c RET</i>	<i>org-table-hline-and-move</i>
Insert a horizontal line below current row, and move the cursor into the row below that line.	
<i>C-c ^</i>	<i>org-table-sort-lines</i>
Sort the table lines in the region. The position of point indicates the column to be used for sorting, and the range of lines is the range between the nearest horizontal separator lines, or the entire table. If point is before the first column, you will be prompted for the sorting column. If there is an active region, the mark specifies the first line and the sorting column, while point should be in the last line to be included into the sorting. The command prompts for the	

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

Regions

C-c C-x M-w **org-table-copy-region**

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

C-c C-x C-w **org-table-cut-region**

Copy a rectangular region from a table to a special clipboard, and blank all fields in the rectangle. So this is the “cut” operation.

C-c C-x C-y **org-table-paste-rectangle**

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

M-RET **org-table-wrap-region**

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

Calculations

C-c + **org-table-sum**

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

S-RET **org-table-copy-down**

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

Miscellaneous

C-c ‘ **org-table-edit-field**

Edit the current field in a separate window. This is useful for fields that are not fully visible (see Section 3.2 [Column width and alignment], page 22). When called with a *C-u* prefix, just make the full field visible, so that it can be edited in place.

M-x org-table-import

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

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

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

M-x org-table-export

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

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

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

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

3.2 Column width and alignment

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

Sometimes a single field or a few fields need to carry more text, leading to inconveniently wide columns. Or maybe you want to make a table with several columns having a fixed width, regardless of content. To set¹ the width of a column, one field anywhere in the column may contain just the string '<N>' where 'N' is an integer specifying the width of the column in characters. The next re-align will then set the width of this column to this value.

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

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

¹ This feature does not work on XEmacs.

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

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

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

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

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

3.3 Column groups

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

N N^2 N^3 N^4 sqrt(n) sqrt[4](N)
/ < > < >
1 1 1 1 1 1
2 4 8 16 1.4142 1.1892
3 9 27 81 1.7321 1.3161

```
#+TBLFM: $2=$1^2::$3=$1^3::$4=$1^4::$5=sqrt($1)::6=sqrt(sqrt(($1)))
```

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

N N^2 N^3 N^4 sqrt(n) sqrt[4](N)
/ < <

3.4 The Orgtbl minor mode

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

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


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

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

3.5 The spreadsheet

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

3.5.1 References

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

Field references

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

Org prefers³ to use another, more general operator that looks like this:

```
@row$column
```

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

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

The row specification only counts data lines and ignores horizontal separator lines (hlines). Like with columns, you can use absolute row numbers ‘1’...‘N’, and row numbers relative to the current row like ‘+3’ or ‘-1’, and `@>` references the last row in the table⁴. You may also specify the row relative to one of the hlines: ‘I’ refers to the first hline⁵, ‘II’

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

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

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

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

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

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

Here are a few examples:

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

Range references

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

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

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

Field coordinates in formulas

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

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

For the second example, table FOO must have at least as many rows as the current table. Note that this is inefficient⁶ for large number of rows.

Named references

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

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

Also properties (see Chapter 7 [Properties and Columns], page 58) can be used as constants in table formulas: for a property ‘:Xyz:’ use the name ‘\$PROP_Xyz’, and the property will be searched in the current outline entry and in the hierarchy above it. If you have the ‘`constants.el`’ package, it will also be used to resolve constants, including natural constants like ‘\$h’ for Planck’s constant, and units like ‘\$km’ for kilometers⁷. Column names and parameters can be specified in special table lines. These are described below, see Section 3.5.8 [Advanced features], page 31. All names must start with a letter, and further consist of letters and numbers.

Remote references

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

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

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

3.5.2 Formula syntax for Calc

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

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

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

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

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

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

Unless you use large integer numbers or high-precision-calculation and `-display` for floating point numbers you may alternatively provide a `printf` format specifier to reformat the Calc result after it has been passed back to Org instead of letting Calc already do the formatting⁸. A few examples:

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

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

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

3.5.3 Emacs Lisp forms as formulas

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

Swap the first two characters of the content of column 1

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

```
'(concat (substring $1 1 2) (substring $1 0 1) (substring $1 2))
Add columns 1 and 2, equivalent to Calcfls $1+$2
'(+ $1 $2);N
Compute the sum of columns 1-4, like Calcfls vsum($1..$4)
'(apply '+ '($1..$4));N
```

3.5.4 Field and range formulas

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

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

C-u C-c = org-table-eval-formula
 Install a new formula for the current field. The command prompts for a formula with default taken from the ‘#+TBLFM:’ line, applies it to the current field, and stores it.

The left-hand side of a formula can also be a special expression in order to assign the formula to a number of different fields. There is no keyboard shortcut to enter such range formulas. To add them, use the formula editor (see Section 3.5.6 [Editing and debugging formulas], page 29) or edit the #+TBLFM: line directly.

\$2= Column formula, valid for the entire column. This is so common that Org treats these formulas in a special way, see Section 3.5.5 [Column formulas], page 28.

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

@1\$2..@4\$3= Range formula, applies to all fields in the given rectangular range. This can also be used to assign a formula to some but not all fields in a row.

\$name= Named field, see Section 3.5.8 [Advanced features], page 31.

3.5.5 Column formulas

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

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

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

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

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

3.5.6 Editing and debugging formulas

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

C-c = or **C-u C-c =** **org-table-eval-formula**
 Edit the formula associated with the current column/field in the minibuffer. See Section 3.5.5 [Column formulas], page 28, and Section 3.5.4 [Field and range formulas], page 28.

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

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

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

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

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

<i>C-c C-c</i> or <i>C-x C-s</i>	<code>org-table-fedit-finish</code>
Exit the formula editor and store the modified formulas. With <i>C-u</i> prefix, also apply the new formulas to the entire table.	
<i>C-c C-q</i>	<code>org-table-fedit-abort</code>
Exit the formula editor without installing changes.	
<i>C-c C-r</i>	<code>org-table-fedit-toggle-ref-type</code>
Toggle all references in the formula editor between standard (like B3) and internal (like @3\$2).	
<i>TAB</i>	<code>org-table-fedit-lisp-indent</code>
Pretty-print or indent Lisp formula at point. When in a line containing a Lisp formula, format the formula according to Emacs Lisp rules. Another <i>TAB</i> collapses the formula back again. In the open formula, <i>TAB</i> re-indents just like in Emacs Lisp mode.	
<i>M-TAB</i>	<code>lisp-complete-symbol</code>
Complete Lisp symbols, just like in Emacs Lisp mode.	
<i>S-up/down/left/right</i>	
Shift the reference at point. For example, if the reference is B3 and you press <i>S-right</i> , it will become C3. This also works for relative references and for hline references.	
<i>M-S-up</i>	<code>org-table-fedit-line-up</code>
<i>M-S-down</i>	<code>org-table-fedit-line-down</code>
Move the test line for column formulas in the Org buffer up and down.	
<i>M-up</i>	<code>org-table-fedit-scroll-down</code>
<i>M-down</i>	<code>org-table-fedit-scroll-up</code>
Scroll the window displaying the table.	
<i>C-c }</i>	
Turn the coordinate grid in the table on and off.	

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

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

Debugging formulas

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

3.5.7 Updating the table

Recalculation of a table is normally not automatic, but needs to be triggered by a command. See Section 3.5.8 [Advanced features], page 31, for a way to make recalculation at least semi-automatic.

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

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

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

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

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

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

3.5.8 Advanced features

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

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

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

	Student	Prob 1	Prob 2	Prob 3	Total	Note
!		P1	P2	P3	Tot	
#	Maximum	10	15	25	50	10.0
^		m1	m2	m3	mt	
#	Peter	10	8	23	41	8.2
#	Sam	2	4	3	9	1.8
	Average				29.7	
^					at	
\$	max=50					

#+TBLFM: \$6=vsum(\$P1..\$P3):: \$7=10*\$Tot/\$max;%.1f:: \$at=vmean(@-II..@-I);%.1f

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

The marking characters have the following meaning:

- ‘!’ The fields in this line define names for the columns, so that you may refer to a column as ‘\$Tot’ instead of ‘\$6’.
- ‘^’ This row defines names for the fields *above* the row. With such a definition, any formula in the table may use ‘\$m1’ to refer to the value ‘10’. Also, if you assign a formula to a names field, it will be stored as ‘\$name=...’.
- ‘_’ Similar to ‘^’, but defines names for the fields in the row *below*.
- ‘\$’ Fields in this row can define *parameters* for formulas. For example, if a field in a ‘\$’ row contains ‘max=50’, then formulas in this table can refer to the value 50 using ‘\$max’. Parameters work exactly like constants, only that they can be defined on a per-table basis.
- ‘#’ Fields in this row are automatically recalculated when pressing TAB or RET or `S-TAB` in this row. Also, this row is selected for a global recalculation with `C-u C-c *`. Unmarked lines will be left alone by this command.
- ‘*’ Selects this line for global recalculation with `C-u C-c *`, but not for automatic recalculation. Use this when automatic recalculation slows down editing too much.
- ‘’ Unmarked lines are exempt from recalculation with `C-u C-c *`. All lines that should be recalculated should be marked with ‘#’ or ‘*’.
- ‘/’ Do not export this line. Useful for lines that contain the narrowing ‘<N>’ markers or column group markers.

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

	Func	n	x	Result
#	exp(x)	1	x	1 + x
#	exp(x)	2	x	1 + x + x ² / 2
#	exp(x)	3	x	1 + x + x ² / 2 + x ³ / 6
#	x ² +sqrt(x)	2	x=0	x*(0.5 / 0) + x ² (2 - 0.25 / 0) / 2
#	x ² +sqrt(x)	2	x=1	2 + 2.5 x - 2.5 + 0.875 (x - 1) ²
*	tan(x)	3	x	0.0175 x + 1.77e-6 x ³

#+TBLFM: \$5=taylor(\$2,\$4,\$3);n3

3.6 Org-Plot

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

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

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

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

Plot Options

set	Specify any <code>gnuplot</code> option to be set when graphing.
title	Specify the title of the plot.
ind	Specify which column of the table to use as the <code>x</code> axis.
deps	Specify the columns to graph as a Lisp style list, surrounded by parentheses and separated by spaces for example <code>dep:(3 4)</code> to graph the third and fourth columns (defaults to graphing all other columns aside from the <code>ind</code> column).
type	Specify whether the plot will be <code>2d</code> , <code>3d</code> , or <code>grid</code> .
with	Specify a <code>with</code> option to be inserted for every col being plotted (e.g. <code>lines</code> , <code>points</code> , <code>boxes</code> , <code>impulses</code> , etc...). Defaults to <code>lines</code> .
file	If you want to plot to a file, specify <code>"path/to/desired/output-file"</code> .
labels	List of labels to be used for the <code>deps</code> (defaults to the column headers if they exist).
line	Specify an entire line to be inserted in the Gnuplot script.
map	When plotting <code>3d</code> or <code>grid</code> types, set this to <code>t</code> to graph a flat mapping rather than a <code>3d</code> slope.
timefmt	Specify format of Org-mode timestamps as they will be parsed by Gnuplot. Defaults to <code>'%Y-%m-%d-%H:%M:%S'</code> .
script	If you want total control, you can specify a script file (place the file name between double-quotes) which will be used to plot. Before plotting, every instance of <code>\$datafile</code> in the specified script will be replaced with the path to the generated data file. Note: even if you set this option, you may still want to specify the plot type, as that can impact the content of the data file.

4 Hyperlinks

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

4.1 Link format

Org-mode は URL のようなリンクを認識して、クリック可能なリンクにしてくれます。Org-mode 上での一般的なリンクのフォーマットは以下のようになります：

`[[リンク][項目名]]` または `[[リンク]]\n`

すべての括弧を入力し終えてリンクが完成すると、Org-mode は、`[[リンク][項目名]]` のかわりに「項目名」を、`[[リンク]]` のかわりに「リンク」を表示します。リンクは「org-link」で設定されたフェイスでハイライトされます。なおデフォルトのフェイスはアンダーラインになっています。表示されている部分については、直接編集することができます。項目名がない場合はリンクの編集になり、項目名がある場合は項目名の編集になることに注意してください。表示されていないリンクを編集するには、該当するリンクの上で「C-c C-l」を実行します。

表示されたテキストの始端もしくは終端にカーソルを置いて `BACKSPACE` を押すと、その場所にある（表示されていない）括弧を取り除くことができます。これによってリンクは不完全なものになり、リンクの内部は再びプレーンなテキストとして表示されます。取り除かれた括弧を再度挿入することでリンクは再び内部化され隠されます。すべてのリンクの内部的な構造を表示するには、Org-`>Hyperlinks->Literal links` というメニューを使用します。

4.2 Internal links

もしもリンクが URL のようなものではない場合、現在使用しているファイル内へのリンクだとみなされます。最も重要なケースは、`'[[#my-custom-id]]'` のようなリンクで、`'my-custom-id'` という `CUSTOM_ID` プロパティのついた見出しへリンクします。このようなカスタム ID は、HTML でのエクスポート (see Section 12.5 [HTML export], page 128) 時にセクション毎のリンクを書き出してくれるので、大変便利です。ただし、各カスタム ID 名がファイル内で重複しないようにする必要があります。

`'[[My Target]]'` や `'[[My Target][Find my target]]'` のようなリンクは、現在使用しているファイル内でのテキスト検索になります。

マウスのクリック、またはリンク上にカーソルがあるときは `C-c C-o` で、リンクを開くことができます (see Section 4.4 [Handling links], page 36)。カスタム ID へのリンクは、対応する見出しを指し示します。テキストへのリンクには、専用のターゲットを用意する方が良いでしょう。同じ文字列を二重の角付きの括弧でくくるとか。ターゲットはどこに置かれていてもかまいません。しかし、コメント行のようなところに置いたほうが便利なのが多いでしょう。たとえば、以下のように。

```
# <<My Target>>\n
```

In HTML export (see Section 12.5 [HTML export], page 128), such targets will become HTML でのエクスポート (see Section 12.5 [HTML export], page 128) では、このようなターゲットは、`'http'` で直接アクセスできる名前付きのアンカーになります¹。

¹ 一番最初の見出しより前の文はエクスポートされないことに注意してください。このためそのようなターゲットの一番最初のものは、最初の見出しより後にある必要があります。

もしも専用のターゲットが存在しない場合には、Org-mode はそのリンクにある語句と同じ見出しを検索しますが、TODO キーワードやタグ² も検索されます。Org-mode 以外のファイルでは、リンクのテキストにある語句を検索します。上記の例では、‘my target’が検索されます。

リンクをたどると、Org-mode 自身のマークリングにマークが格納されます。C-c & で、ひとつ前のポジションに戻ります。このコマンドを連続して使うことにより、更に前のポジションへと戻ることができます。

4.2.1 Radio targets

Org-mode は、通常のテキスト部分に書かれたターゲット名を、自動でリンクにすることができます。ですから、わざわざ個別のリンクを作成することなく、それぞれのテキストがターゲットにリンクされます。ラジオターゲットは、3つの角括弧で囲まれたものです。例えば、‘<<<My Target>>>’というターゲットは、通常のテキスト部分に‘my target’が登場する度にアクティブなリンクにしてくれます。Org-mode ファイル内のラジオターゲットは、最初の読み込み時のみ自動的にスキャンされます。編集中にターゲットをアップデートしてリンクするには、ターゲットのところで C-c C-c を実行します。

4.3 External links

Org-mode は次のようなリンクをサポートしています。ファイル、web サイト、ネットニュース、email のメッセージ、BBDB データベースの項目、IRC の会話とログなど。外部リンクは URL を記載するような方法でリンクします。それらはコロンの前に短い定義用の文字列をつけて表記します。コロンのあとに空白をとってはいけません。以下に例とそれぞれのリンクのタイプの一覧を示します。

<code>http://www.astro.uva.nl/~dominik</code>	ウェブへのリンク
<code>doi:10.1000/182</code>	電子文献のための DOI
<code>file:/home/dominik/images/jupiter.jpg</code>	ファイルへの絶対パス
<code>/home/dominik/images/jupiter.jpg</code>	上に同じ
<code>file:papers/last.pdf</code>	ファイルへの相対パス
<code>./papers/last.pdf</code>	same as above
<code>file:/myself@some.where:papers/last.pdf</code>	リモートマシン上のファイルへのパス
<code>/myself@some.where:papers/last.pdf</code>	same as above
<code>file:sometextfile::NNN</code>	ジャンプ先の行番号付きファイル
<code>file:projects.org</code>	他の Org-mode ファイルへのリンク
<code>file:projects.org::some words</code>	Org-mode ファイル内でのテキスト検索
<code>file:projects.org::*task title</code>	Org-mode ファイル内での見出し検索
<code>docview:papers/last.pdf::NNN</code>	ファイルをページ指定して開く
<code>id:B7423F4D-2E8A-471B-8810-C40F074717E9</code>	ID のついた見出しへのリンク
<code>news:comp.emacs</code>	Usenet へのリンク
<code>mailto:adent@galaxy.net</code>	メールリンク
<code>vm:folder</code>	VM のフォルダーへのリンク
<code>vm:folder#id</code>	VM のメールメッセージへのリンク
<code>vm://myself@some.where.org/folder#id</code>	リモートマシン上の VM のメールメッセ
<code>セ</code>	

² 見出しへのリンクを挿入するときは、バッファ内補完をすることができます。*印の後にいくつか文字を入力し、M-TAB を実行してください。現在のバッファ内にあるすべての見出しが補完候補として表示されます。

ージへのリンク	
wl:folder ンク	WANDERLUST のフォルダーへのリ
wl:folder#id へのリ ンク	WANDERLUST のメールメッセージ
mhe:folder	MH-E のフォルダーへのリンク
mhe:folder#id	MH-E のメールメッセージへのリンク
rmail:folder	RMAIL のフォルダーへのリンク
rmail:folder#id ク	RMAIL のメールメッセージへのリン
gnus:group	Gnus グループへのリンク
gnus:group#id	Gnus の記事へのリンク
bbdb:R.*Stallman	BBDB へのリンク (正規表現)
irc:/irc.com/#emacs/bob	IRC へのリンク
info:org#External%20links	Info ノードへのリンク (空白をエンコー
ド)	
shell:ls *.org	シェルコマンドへのリンク
elisp:org-agenda	Elisp コマンドへのリンク
elisp:(find-file-other-frame "Elisp.org")	Elisp フォームを評価

Org-mode をカスタマイズして、新しいリンクのタイプを付け加えるには、Section A.3 [Adding hyperlink types], page 186 を参照してください。

リンクは二重括弧で囲んだ方が良いでしょう。URL の代わりにテキストを表示することもできます (see Section 4.1 [Link format], page 34)。たとえば、以下のように。

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

項目名が画像を指し示すファイル名や URL の場合、HTML エクスポート (see Section 12.5 [HTML export], page 128) によって、画像はクリック可能なボタンとして書き出されます。項目名がない画像の場合には、インライン画像として書き出されます。

Org-mode は、通常のテキスト内のリンクを見つけ出し、外部リンクとします。もしスペースがリンクの一部として必要な場合 (たとえば、‘bbdb:Richard Stallman’) や、リンクの末端であいまいさをなくしたい場合、角括弧で囲みます。

4.4 Handling links

Org-mode は正しい構文でリンクを作成したり、Org-mode のファイルにリンクを挿入したり、そのリンクをたどったりする方法を提供しています。

C-c l

org-store-link

現在の位置へのリンクを一時保管します。これはリンクを作成するときに、どのバッファでも使用できるグローバルなコマンド (あなた自身でキーバインドを作成しなければなりません。) です。リンクは一時保管され、後から Org-mode のバッファへ挿入することができます (下記参照)。どの種類のリンクが作成されるかは、現在のバッファが何であるかによります。

Org-mode バッファ

もしカーソル位置に ‘<<target>>’ がある場合、リンクはそのターゲットを指します。それ以外の場合は、見出しを指し、見出しは項目名にもなります。

もし見出しに `CUSTOM_ID` プロパティがある場合、このカスタム ID プロパティへのリンクが一時保管されます。また、(`org-link-to-org-use-id`の値によりますが) グローバルに固有の ID プロパティが、リンクを作成するために作られます。ですから、このコマンドを Org-mode バッファ内で使うと、潜在的にふたつのリンクを作成することになります。つまり、Org-mode バッファ内でこのコマンドを使用することにより、人間が読むことのできるカスタム ID と、グローバルに固有で、エントリーがファイル間を移動しても動作するリンクが作成されます。後にリンクを挿入するときには、どのリンクを挿入するかを決めなければなりません。

メール/ニュースクライアント:**VM, Rmail, Wanderlust, MH-E, Gnus**

ほとんどすべての Emacs のメールクライアントがサポートされています。リンクは現在の記事を指し示します。Gnus バッファでは、グループを指し示すこともあるでしょう。項目名は筆者名と題名から作成されます。

ウェブブラウザ:**W3 and W3M**

現在の URL がリンクになり、ページのタイトルが項目名になります。

連絡先:**BBDB**

BBDB バッファ内で作成されたリンクは、現在のエントリーへのリンクになります。

Chat: **IRC**

IRC リンクの場合、`org-irc-link-to-logs`変数を `t`にした場合は、ログファイル内に、現在の会話に相当する部分への `'file:/'`形式のリンクが作成されます。それ以外の場合は、`user/channel/server` への `'irc:/'`スタイルのリンクが一時保管されます。

その他のファイル

その他のファイルの場合、リンクは、現在の行を指ししめす検索語句 (see Section 4.7 [Search options], page 40) を伴って作成されます。もし、アクティブなリージョンがある場合は、選択された言葉が検索語句の基本となります。自動的に作成されたリンクがうまく働かなかったり、不正確であったりする場合は、カスタム関数を書いて、検索語句を選択したり、特定のファイル形式を検索したりすることがきます。Section 4.8 [Custom searches], page 41 を参照してください。C-c l というキーバインディングはひとつの提案に過ぎません Section 1.2 [Installation], page 3)。

アジェンダビュー

カーソルがアジェンダビューにあるときは、作成されたリンクは現在の行が参照するエントリーを指し示します。

C-c C-l

org-insert-link

リンク³を挿入します。そうすると、プロンプトによって、バッファに挿入するリンクをたずねられます。テキストを使った内部リンクや、既に述べましたリンクタイプのいずれかへのリンクを入力するだけです。リンクは項目名とともに、バッファ⁴に挿入されます。もし、このコマンドの呼び出し時にテキストが選択されていた場合には、選択されたテキストがデフォルトの項目名になります。

一時保管されたリンクを挿入

<現在のセッションで一時保管されたすべてのリンクは、このプロンプトの履歴となって

³ リンクを挿入するのに、このコマンドを使わなければならないわけではないことに注意してください。Org-mode でのリンクはプレーンテキストですので、リンクはタイプしたりペーストしたりして直接バッファへ入力することができます。このコマンドを使うことによって、リンクは自動的に二重括弧に入れられ、オプションとして項目名を入力するかどうかをたずねられます。

⁴ 一時保管されたリンクを挿入した後は、そのリンクは一時保管リストから削除されます。後で使用するためにリンクを保存したままにしておきたい場合は、C-c C-lの前に 3 回 C-u をタイプするか、`org-keep-stored-link-after-insertion` オプションを設定してください。

いますので、`up`や`down`(あるいは、`M-p/n`)を使ってこれらにアクセスすることができます。

補完の支援

TABを使用した補完機能によって、リンクの省略記法で定義された接頭辞 (see Section 4.6 [Link abbreviations], page 39) を含む、`'http'`や`'ftp'`などのリンクを適切に挿入することができます。もし、接頭辞のみをタイプした後にRETを押すと、Org-modeは、いくつかのリンク形式⁵に対して詳細な補完の支援を行います。たとえば、`file RET`をタイプすると、ファイル名の補完 ((または、`C-u C-c C-l`をタイプします。以下を参照。))を行い、`bbdb RET`をタイプすると、連絡先の名前を補完することができます。

`C-u C-c C-l`

接頭辞 `C-u`を付けて `C-c C-l`が呼び出されたときは、ファイルへのリンクが挿入され、ファイル名の補完を利用することができます。リンクされたファイルがカレントディレクトリにあるときや、カレントディレクトリのサブディレクトリにあるとき、あるいはパスが`../`を使って相対パスで書かれているときは、ファイルへのパスは、現在のOrg-mode ファイルからの相対パスとして挿入されます。それ以外の場合は絶対パスが使われ、可能であれば、ホームディレクトリには`~/`が使われます。2つの `C-u`を付けることによって、絶対パス表記を明確に指定することができます。

`C-c C-l` (カーソルがリンク上にある場合)

リンク上にカーソルがある場合、`C-c C-l`を実行すると、リンクと項目名を編集することができます。

`C-c C-o`

`org-open-at-point`

その場所にあるリンクを開きます。リンクがURL ならば、(`browse-url-at-point`を使って) ウェブブラウザを開きますし、それぞれ対応するリンクにより、VM/MH-E/Wanderlust/Rmail/Gnus/BBDB が起動し、シェルへのリンクの場合はコマンドを実行します。カーソルが内部リンク上にあるときは、対応する検索を行います。カーソルが見出しのタグ上にあるときは、対応するタグビューを作成します。カーソルがタイムスタンプ上にあるときは、その日のアジェンダを表示します。さらに、`'file:'`リンクの場合、テキストファイルやリモートマシン上のファイルは Emacs で、非テキストファイルは適切なアプリケーションで、ファイルを開きます。ファイルの分類は、拡張子のみによって判断されます。`org-file-apps`を参照してください。もし、デフォルトのアプリケーションではなく Emacs でファイルを開きたい場合は、接頭辞 `C-u`を付け、Emacs で開くことを避けたい場合は、接頭辞 `C-u C-u`を付けてください。カーソルがリンクではない見出し上にあるときは、見出し上のすべてのリンクとエントリーテキストを表示します。

RET `org-return-follows-link`が設定されているときは、RETもポイント上のリンクを開きます。

`mouse-2`

mouse-1 リンク上では、`mouse-2`は `C-c C-o`と同様にリンクを開きます。Emacs 22 以降では、`mouse-1`もリンクを開きます。

mouse-3 `mouse-2`と同様にリンクを開きますが、ファイルのリンクを強制的に Emacs で開き、内部リンクは別のウインドウ⁶で開きます。

⁵ これは `org-PREFIX-complete-link`という特別な関数を呼び出すことによって行います。

⁶ `org-display-internal-link-with-indirect-buffer` 変数を参照してください。

C-c C-x C-v **org-toggle-inline-images**
 リンクされた画像のインライン表示をトグルします。通常これはリンクに項目名がない画像のみをインライン表示するものです。すなわち、エクスポート時にインラインになる画像のことです。**org-startup-with-inline-images**変数⁷を設定することにより、インライン画像を起動時に表示されることができます。

C-c % **org-mark-ring-push**
 現在のポジションをマークリングに格納し、現在のポジションに簡単に戻ってこられるようにします。ファイル内部でのリンクをたどるときは、自動的にこれが行われます。

C-c & **org-mark-ring-goto**
 記録されたポジションへ戻ります。ポジションは、内部リンクをたどるコマンドと **C-c %**によって記録されます。このコマンドを連続して何回か使うと、記録されたポジション間を移動することができます。

C-c C-x C-n **org-next-link**
C-c C-x C-p **org-previous-link**
 バッファ内の前後のリンクへ移動します。バッファの端では、移動は、いったんエラーになり、もう一度行くと回り込みます。このキーバインドはとても長いので、**C-n**と **C-p**に設定したいと考えるかもしれません。

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

4.5 Using links outside Org

Org-mode だけでなく、どの Emacs のバッファでも、Org-mode 構文を持つリンクを挿入し、たどることができます。このためには、次のような 2 つのグローバルコマンドを作成しなければなりません (自分に適したグローバルなキーを設定してください)。

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

4.6 Link abbreviations

長い URL をタイプするのは面倒ですが、往々にしてひとつの文章には似たようなリンクが数多く登場するものです。このような場合には、リンクの省略記法を使うことができます。省略記法されたリンクは次のようなものです。

[[リンク語句:タグ][項目名]]\n

タグはなくても構いません。リンク語句は文字で始まる語句、数字、`'-'`、`'_'`を使うことができます。省略記法は、リンク語句とリンクテキストを関連づける **org-link-abbrev-alist**変数の値にしたがって展開されます。以下に例を示します。

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

⁷ 対応する `#+STARTUPinlineimages` と `inlineimages` と共に。

置き換えるテキストに‘%s’が含まれている場合は、タグに置き換えられます。それ以外の場合は、タグはリンクを作成するために文字列に付け加えられます。リンクを作成する引数としてタグと一緒に呼び出される機能を指定したほうがよいかもしれません。

上記の設定だと、`[[bugzilla:129]]`で特定のバグへリンクすることができ、`[[google:OrgMode]]`で‘OrgMode’をウェブ検索することができ、`[[gmap:51 Franklin Street, Boston]]`で Free Software Foundation の、`[[omap:Science Park 904, Amsterdam, The Netherlands]]`で Carsten のオフィスの地図上の位置を表示することができ、`[[ads:Dominik,C]]`で Org-mode の作者が Emacs のハッキングの他に何をしているかを見つけることができます。

ある特定の Org-mode バッファでだけリンクの省略記法を使いたい場合は、次のようにすることで定義できます。

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

In-buffer completion (see Section 15.1 [Completion], page 174) can be used after ‘[’ to complete link abbreviations. You may also define a function `org-PREFIX-complete-link` that implements special (e.g. completion) support for inserting such a link with `C-c C-l`. Such a function should not accept any arguments, and return the full link with prefix.

4.7 ファイルリンクにおける検索オプション

ファイルへのリンクには、ファイル内の特定の場所へジャンプするリンクを含ませることができます。これは、ダブルコロン⁸の後に行番号や検索語句を置くことによって行います。たとえば、`C-c l` コマンドをタイプして、あるファイルへのリンクを作成する場合 (see Section 4.4 [Handling links], page 36)、現在の行の言葉を検索語句としてリンクに含めることができ、`C-c C-o` コマンドで開くことができます。

説明と共に、あるファイルリンクへの検索語句を付加する様々な構文の方法を示します。

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

255 255 行目へジャンプします。

My Target 内部リンクの検索と同様に、‘<<My Target>>’という名前のリンクターゲット、あるいは ‘my target’ というテキストを検索します (Section 4.2 [Internal links], page 34 を参照してください。)。HTML エクスポート (see Section 12.5 [HTML export], page 128) では、このようなファイルへのリンクは、リンク先のファイル内にある、一致する名前のアンカーへの HTML リンクになります。

***My Target**

Org-mode ファイルの中で見出しの検索に限定されます。

#my-custom-id

CUSTOM_ID プロパティを持つ見出しへリンクします。

⁸ 下方互換性のために、行番号はシングルコロンの後に置くこともできます。

/regexp/ **regexp**を正規表現検索します。これは、Emacs の **occur** コマンドを使って、一致するすべてを別ウィンドウでリスト表示します。ターゲットが Org-mode ファイルならば、**org-occur**が使われ、一致した部分について、ツリーの抽出を行います。

特殊なケースとして、ファイル名が指定されていないファイルのリンクは、現在のファイルの検索となります。たとえば、`[[file::find me]]` は、`'[[find me]]'` と同様に、現在のファイルで `'find me'` を検索します。

4.8 カスタム検索

デフォルトの検索文字列作成のメカニズムと、実際のファイル検索のメカニズムは、すべての場合でうまく動作するとは限りません。たとえば、BibTeX データベースのファイルは、`'year=\"1993\"'` のようなエントリーをたくさん有していますが、これは良い検索文字列であるとは言えません。なぜならば、BibTeX のエントリーでは、唯一の識別情報は引用キーだからです。

このような問題に直面した場合は、特定のファイルタイプに適した検索文字列を設定し、そのファイルで検索を行うカスタム関数を書くことができます。**add-hook** を使用して、これらの関数は、**org-create-file-search-functions**、**org-execute-file-search-functions** というフック変数に付け加えられる必要があります。これらの変数についてのより詳しい情報は、ドキュメント文字列を参照してください。Org-mode は、実際にこのメカニズムを BibTeX データベースファイルに使用しており、該当するコードを実装のサンプルとして使うことができます。`'org-bibtex.el'` というファイルを参照してください。

5 TODO アイテム

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

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

5.1 基本的な TODO の機能

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

```
*** TODO サム フォーチュンに手紙を書く。
```

TODO 項目を入力するときの重要なコマンドは以下のとおりです。

C-c C-t **org-todo**

現在の TODO の状態を次のように切り替えます。

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

同じような状態の切り替えは、タイムラインとアジェンダバッファで **t** コマンドキー (see Section 10.5 [Agenda commands], page 101 参照) を入力することで「リモートで」完了にすることもできます。

C-u C-c C-t

補完や「すでに設定されていれば」さらに速い選択方法を提供するインターフェイスを使用して特定のキーワードを選択します。後者の方法では、TODO の状態に対してキーを割り振る必要があります。詳細は、Section 5.2.5 [Per-file keywords], page 45 と Section 6.2 [Setting tags], page 54 を参照してください。

S-right / S-left

切り替えの機能に似て、後にくる TODO の状態、あるいは前にくるものを選択します。もっとも役に立つのは TODO の状態が 2 段階以上の場合です。(see Section 5.2 [TODO extensions], page 43). **shift-selection-mode** との連携については、Section 15.10.2 [Conflicts], page 184 も参照してください。変数 **org-treat-S-cursor-todo-selection-as-state-change**。

C-c / t

org-show-todo-key

も参照してください。ツリーの抽出機能を使って TODO を確認します (see Section 2.6 [Sparse trees], page 12) 参照。バッファ全体を折り畳みますが、全ての TODO 項目「DONE 状態以外の」とその階層の見出しを表示します。接頭辞をつけることで (もしくは、キーバインド **C-c / T**)、ある特定の DONE 状態の項目も表示させることができます。検索用のキーワードを入力するためのプロンプトが表示されます。さらにキーワードのリストを次のように入力することもでき **KWD1|KWD2|...**、この内のどれかに一致するものが表示されます。前置引数 **N** を使って、変数 **org-todo-keywords** 内の **N** 番

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

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

C-c a t

org-todo-list

グローバル TODO リストを表示します。すべての「DONE 状態以外の」TODO アイテムをすべてのアジェンダファイル (see Chapter 10 [Agenda Views], page 90) から集めて、一つのバッファに表示します。その新しくできたバッファは、**agenda-mode**で表示され、確認や修正を加えるためのコマンドも提供されます。(see Section 10.5 [Agenda commands], page 101). See Section 10.3.2 [Global TODO list], page 94. を参照してください。

S-M-RET

org-insert-todo-heading

新しい TODO を現在の位置に入力します。

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

5.2 TODO キーワードの拡張的な使い方

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

注記、タグは見出しと特に TODO アイテムの分類のもう一つの方法です。(see Chapter 6 [Tags], page 54).

5.2.1 ワークフローの状態としての TODO キーワード

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

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

縦線は、TODO キーワード「アクションが必要な状態」と DONE 状態「アクションが不要な状態」からを分離します。縦線が指定されていない場合は、最後の状態が、DONE 状態として使用されます。この設定により、コマンド **C-c C-t**で、TODO、FEEDBACK、それから VERIFY、最後に DONE、DELEGATED というように順番切り替えます。前置引数を使用することで、特定の状態を即座に選択することもできます。例えば **C-3 C-c C-t**と入力すると、ダイレクトに 3 番目の VERIFY に変更することができます。もしくは、**S-left**により、逆の方向に順番に切り替えることもできます。もしも、たくさんのキーワードを定義した場合は、入力補完機能 (see Section 15.1 [Completion], page 174) か特別な一つのキーによる選択によりバッファに入力することができます。(see Section 5.2.4 [Fast access to TODO states], page 44)TODO の状態の変更は、タイムスタンプと共にログをとることができます。(より詳しい情報は Section 5.3.2 [Tracking TODO state changes], page 47 参照)

5.2.2 種類としての TODO キーワード

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

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

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

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

5.2.3 同一ファイル内での複数のキーワードセット

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

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

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

C-u C-u C-c C-t

C-S-right

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

S-right

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

5.2.4 Fast access to TODO states

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

³ タイムラインやアジェンダのバッファでは、「t」コマンドも同じ仕様です。

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

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

5.2.5 ファイル別にキーワードを設定する

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

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

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

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

同時に複数のキーワードセットの設定には:

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

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

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

5.2.6 Faces for TODO keywords

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

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

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

⁴ 変数 `org-fast-tag-selection-include-todo` も見てください、この変数は、タグを使って状態の変更を可能にします (see Section 6.2 [Setting tags], page 54)、この二つを混ざって使いたいならですが。この場合、それぞれのキーワードセットに単一のキーを準備する必要があります。

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

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

5.2.7 TODO dependencies

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

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

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

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

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

If you set the variable `org-agenda-dim-blocked-tasks`, TODO entries that cannot be closed because of such dependencies will be shown in a dimmed font or even made invisible in agenda views (see Chapter 10 [Agenda Views], page 90).

You can also block changes of TODO states by looking at checkboxes (see Section 5.6 [Checkboxes], page 51). If you set the variable `org-enforce-todo-checkbox-dependencies`, an entry that has unchecked checkboxes will be blocked from switching to DONE.

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

5.3 Progress logging

Org-mode can automatically record a timestamp and possibly a note when you mark a TODO item as DONE, or even each time you change the state of a TODO item. This system is highly configurable, settings can be on a per-keyword basis and can be localized to a file or even a subtree. For information on how to clock working time for a task, see Section 8.4 [Clocking work time], page 73.

5.3.1 Closing items

The most basic logging is to keep track of *when* a certain TODO item was finished. This is achieved with¹

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

Then each time you turn an entry from a TODO (not-done) state into any of the DONE states, a line 'CLOSED: [timestamp]' will be inserted just after the headline. If you turn the entry back into a TODO item through further state cycling, that line will be removed again. If you want to record a note along with the timestamp, use²

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

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

In the timeline (see Section 10.3.4 [Timeline], page 97) and in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 92), you can then use the `l` key to display the TODO items with a 'CLOSED' timestamp on each day, giving you an overview of what has been done.

5.3.2 Tracking TODO state changes

When TODO keywords are used as workflow states (see Section 5.2.1 [Workflow states], page 43), you might want to keep track of when a state change occurred and maybe take a note about this change. You can either record just a timestamp, or a time-stamped note for a change. These records will be inserted after the headline as an itemized list, newest first³. When taking a lot of notes, you might want to get the notes out of the way into a drawer (see Section 2.8 [Drawers], page 16). Customize the variable `org-log-into-drawer` to get this behavior—the recommended drawer for this is called `LOGBOOK`. You can also overrule the setting of this variable for a subtree by setting a `LOG_INTRO_DRAWER` property.

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

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

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

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

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

you not only define global TODO keywords and fast access keys, but also request that a time is recorded when the entry is set to DONE⁴, and that a note is recorded when switching to WAIT or CANCELED. The setting for WAIT is even more special: the ‘!’ after the slash means that in addition to the note taken when entering the state, a timestamp should be recorded when *leaving* the WAIT state, if and only if the *target* state does not configure logging for entering it. So it has no effect when switching from WAIT to DONE, because DONE is configured to record a timestamp only. But when switching from WAIT back to TODO, the ‘/!’ in the WAIT setting now triggers a timestamp even though TODO has no logging configured.

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

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

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

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

5.3.3 習慣の追跡

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

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

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

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

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

```

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

```

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

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

Blue	If the task wasn't to be done yet on that day.
Green	If the task could have been done on that day.
Yellow	If the task was going to be overdue the next day.
Red	If the task was overdue on that day.

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

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

org-habit-graph-column

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

The amount of history, in days before today, to appear in consistency graphs.

The number of days after today that will appear in consistency graphs.

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

5.4 Priorities

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

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

<i>S-up</i>	org-priority-up
<i>S-down</i>	org-priority-down

Increase/decrease priority of current headline⁵. Note that these keys are also used to modify timestamps (see Section 8.2 [Creating timestamps], page 67). See also Section 15.10.2 [Conflicts], page 184, for a discussion of the interaction with `shift-selection-mode`.

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

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

5.5 Breaking tasks down into subtasks

It is often advisable to break down large tasks into smaller, manageable subtasks. You can do this by creating an outline tree below a TODO item, with detailed subtasks on the tree⁶. To keep the overview over the fraction of subtasks that are already completed, insert either ‘[/]’ or ‘[%]’ anywhere in the headline. These cookies will be updated each time the TODO status of a child changes, or when pressing `C-c C-c` on the cookie. For example:

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

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

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

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

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

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

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

Another possibility is the use of checkboxes to identify (a hierarchy of) a large number of subtasks (see Section 5.6 [Checkboxes], page 51).

5.6 Checkboxes

Every item in a plain list⁷ (see Section 2.7 [Plain lists], page 13) can be made into a checkbox by starting it with the string ‘[]’. This feature is similar to TODO items (see Chapter 5 [TODO Items], page 42), but is more lightweight. Checkboxes are not included into the global TODO list, so they are often great to split a task into a number of simple steps. Or you can use them in a shopping list. To toggle a checkbox, use `C-c C-c`, or use the mouse (thanks to Piotr Zielinski’s ‘org-mouse.el’).

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

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

Here is an example of a checkbox list.

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

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

The ‘[2/4]’ and ‘[1/3]’ in the first and second line are cookies indicating how many checkboxes present in this entry have been checked off, and the total number of checkboxes present. This can give you an idea on how many checkboxes remain, even without opening a folded entry. The cookies can be placed into a headline or into (the first line of) a plain list item. Each cookie covers checkboxes of direct children structurally below the headline/item on which the cookie appears⁸. You have to insert the cookie yourself by typing either ‘[/]’ or ‘[%]’. With ‘[/]’ you get an ‘n out of m’ result, as in the examples above. With ‘[%]’ you get information about the percentage of checkboxes checked (in the above example, this would be ‘[50%]’ and ‘[33%]’, respectively). In a headline, a cookie can count either checkboxes below the heading or TODO states of children, and it will display whatever was changed last. Set the property `COOKIE_DATA` to either ‘checkbox’ or ‘todo’ to resolve this issue.

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

The following commands work with checkboxes:

C-c C-c **org-toggle-checkbox**
 Toggle checkbox status or (with prefix arg) checkbox presence at point. With double prefix argument, set it to ‘[-]’, which is considered to be an intermediate state.

C-c C-x C-b **org-toggle-checkbox**
 Toggle checkbox status or (with prefix arg) checkbox presence at point. With double prefix argument, set it to ‘[-]’, which is considered to be an intermediate state.

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

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

- M-S-RET*** **org-insert-todo-heading**
Insert a new item with a checkbox. This works only if the cursor is already in a plain list item (see Section 2.7 [Plain lists], page 13).
- C-c C-x o*** **org-toggle-ordered-property**
Toggle the **ORDERED** property of the entry, to toggle if checkboxes must be checked off in sequence. A property is used for this behavior because this should be local to the current entry, not inherited like a tag. However, if you would like to *track* the value of this property with a tag for better visibility, customize the variable **org-track-ordered-property-with-tag**.
- C-c #*** **org-update-statistics-cookies**
Update the statistics cookie in the current outline entry. When called with a ***C-u*** prefix, update the entire file. Checkbox statistic cookies are updated automatically if you toggle checkboxes with ***C-c C-c*** and make new ones with ***M-S-RET***. TODO statistics cookies update when changing TODO states. If you delete boxes/entries or add/change them by hand, use this command to get things back into sync. Or simply toggle any entry twice (checkboxes with ***C-c C-c***).

6 Tags

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

Every headline can contain a list of tags; they occur at the end of the headline. Tags are normal words containing letters, numbers, ‘_’, and ‘@’. Tags must be preceded and followed by a single colon, e.g., ‘:work:’. Several tags can be specified, as in ‘:work:urgent:’. Tags will by default be in bold face with the same color as the headline. You may specify special faces for specific tags using the variable `org-tag-faces`, in much the same way as you can for TODO keywords (see Section 5.2.6 [Faces for TODO keywords], page 45).

6.1 Tag inheritance

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

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

the final heading will have the tags ‘:work:’, ‘:boss:’, ‘:notes:’, and ‘:action:’ even though the final heading is not explicitly marked with those tags. You can also set tags that all entries in a file should inherit just as if these tags were defined in a hypothetical level zero that surrounds the entire file. Use a line like this¹:

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

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

When a headline matches during a tags search while tag inheritance is turned on, all the sublevels in the same tree will (for a simple match form) match as well². The list of matches may then become very long. If you only want to see the first tags match in a subtree, configure the variable `org-tags-match-list-sublevels` (not recommended).

6.2 Setting tags

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

C-c C-q **org-set-tags-command**
Enter new tags for the current headline. Org-mode will either offer completion or a special single-key interface for setting tags, see below. After pressing RET, the tags will be inserted and aligned to `org-tags-column`. When called with a *C-u* prefix, all tags in the current buffer will be aligned to that column, just to make things look nice. TAGS are automatically realigned after promotion, demotion, and TODO state changes (see Section 5.1 [TODO basics], page 42).

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

² This is only true if the search does not involve more complex tests including properties (see Section 7.3 [Property searches], page 60).

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

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

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

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

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

```
#+TAGS:
```

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

```
#+STARTUP: noptag
```

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

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

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

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

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

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

or write them in two lines:

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

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

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

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

Don't forget to press `C-c C-c` with the cursor in one of these lines to activate any changes. To set these mutually exclusive groups in the variable `org-tag-alist`, you must use the dummy tags `:startgroup` and `:endgroup` instead of the braces. Similarly, you can use `:newline` to indicate a line break. The previous example would be set globally by the following configuration:


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

If at least one tag has a selection key then pressing `C-c C-c` will automatically present you with a special interface, listing inherited tags, the tags of the current headline, and a list of all valid tags with corresponding keys³. In this interface, you can use the following keys:

<code>a-z...</code>	Pressing keys assigned to tags will add or remove them from the list of tags in the current line. Selecting a tag in a group of mutually exclusive tags will turn off any other tags from that group.
<code>TAB</code>	Enter a tag in the minibuffer, even if the tag is not in the predefined list. You will be able to complete on all tags present in the buffer. You can also add several tags: just separate them with a comma.
<code>SPC</code>	Clear all tags for this line.
<code>RET</code>	Accept the modified set.
<code>C-g</code>	Abort without installing changes.
<code>q</code>	If <code>q</code> is not assigned to a tag, it aborts like <code>C-g</code> .
<code>!</code>	Turn off groups of mutually exclusive tags. Use this to (as an exception) assign several tags from such a group.
<code>C-c</code>	Toggle auto-exit after the next change (see below). If you are using expert mode, the first <code>C-c</code> will display the selection window.

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

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

6.3 Tag searches

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

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

- `C-c / m` or `C-c \` `org-match-sparse-tree`
Create a sparse tree with all headlines matching a tags search. With a `C-u` prefix argument, ignore headlines that are not a TODO line.
- `C-c a m` `org-tags-view`
Create a global list of tag matches from all agenda files. See Section 10.3.3 [Matching tags and properties], page 95.
- `C-c a M` `org-tags-view`
Create a global list of tag matches from all agenda files, but check only TODO items and force checking subitems (see variable `org-tags-match-list-sublevels`).

These commands all prompt for a match string which allows basic Boolean logic like `+boss+urgent-project1`, to find entries with tags `boss` and `urgent`, but not `project1`, or `Kathy|Sally` to find entries which are tagged, like `Kathy` or `Sally`. The full syntax of the search string is rich and allows also matching against TODO keywords, entry levels and properties. For a complete description with many examples, see Section 10.3.3 [Matching tags and properties], page 95.

7 プロパティ (属性) とカラム (列)

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

Properties can be conveniently edited and viewed in column view (see Section 7.5 [Column view], page 61).

7.1 Property syntax

Properties are key-value pairs. They need to be inserted into a special drawer (see Section 2.8 [Drawers], page 16) with the name `PROPERTIES`. Each property is specified on a single line, with the key (surrounded by colons) first, and the value after it. Here is an example:

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

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

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

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

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

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

The following commands help to work with properties:

<i>M-TAB</i>	pcomplete
After an initial colon in a line, complete property keys. All keys used in the current file will be offered as possible completions.	
<i>C-c C-x p</i>	org-set-property
Set a property. This prompts for a property name and a value. If necessary, the property drawer is created as well.	
<i>M-x org-insert-property-drawer</i>	
Insert a property drawer into the current entry. The drawer will be inserted early in the entry, but after the lines with planning information like deadlines.	
<i>C-c C-c</i>	org-property-action
With the cursor in a property drawer, this executes property commands.	
<i>C-c C-c s</i>	org-set-property
Set a property in the current entry. Both the property and the value can be inserted using completion.	
<i>S-right</i>	org-property-next-allowed-value
<i>S-left</i>	org-property-previous-allowed-value
Switch property at point to the next/previous allowed value.	
<i>C-c C-c d</i>	org-delete-property
Remove a property from the current entry.	
<i>C-c C-c D</i>	org-delete-property-globally
Globally remove a property, from all entries in the current file.	
<i>C-c C-c c</i>	org-compute-property-at-point
Compute the property at point, using the operator and scope from the nearest column format definition.	

7.2 Special properties

Special properties provide an alternative access method to Org-mode features, like the TODO state or the priority of an entry, discussed in the previous chapters. This interface exists so that you can include these states in a column view (see Section 7.5 [Column view], page 61), or to use them in queries. The following property names are special and (except for `:CATEGORY:`) should not be used as keys in the properties drawer:

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

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

7.3 Property searches

To create sparse trees and special lists with selection based on properties, the same commands are used as for tag searches (see Section 6.3 [Tag searches], page 56).

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

C-c a m org-tags-view
Create a global list of tag/property matches from all agenda files. See Section 10.3.3 [Matching tags and properties], page 95.

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

The syntax for the search string is described in Section 10.3.3 [Matching tags and properties], page 95.

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

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

7.4 Property Inheritance

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

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

COLUMNS The `:COLUMNS:` property defines the format of column view (see Section 7.5 [Column view], page 61). It is inherited in the sense that the level where a `:COLUMNS:` property is defined is used as the starting point for a column view table, independently of the location in the subtree from where columns view is turned on.

- CATEGORY** For agenda view, a category set through a `:CATEGORY:` property applies to the entire subtree.
- ARCHIVE** For archiving, the `:ARCHIVE:` property may define the archive location for the entire subtree (see Section 9.6.1 [Moving subtrees], page 88).
- LOGGING** The `LOGGING` property may define logging settings for an entry or a subtree (see Section 5.3.2 [Tracking TODO state changes], page 47).

7.5 Column view

A great way to view and edit properties in an outline tree is *column view*. In column view, each outline node is turned into a table row. Columns in this table provide access to properties of the entries. Org-mode implements columns by overlaying a tabular structure over the headline of each item. While the headlines have been turned into a table row, you can still change the visibility of the outline tree. For example, you get a compact table by switching to `CONTENTS` view (`S-TAB S-TAB`, or simply `c` while column view is active), but you can still open, read, and edit the entry below each headline. Or, you can switch to column view after executing a sparse tree command and in this way get a table only for the selected items. Column view also works in agenda buffers (see Chapter 10 [Agenda Views], page 90) where queries have collected selected items, possibly from a number of files.

7.5.1 Defining columns

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

7.5.1.1 Scope of column definitions

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

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

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

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

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

7.5.1.2 Column attributes

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

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

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

<i>width</i>	An integer specifying the width of the column in characters. If omitted, the width will be determined automatically.
<i>property</i>	The property that should be edited in this column. Special properties representing meta data are allowed here as well (see Section 7.2 [Special properties], page 59)
<i>title</i>	The header text for the column. If omitted, the property name is used.
<i>{summary-type}</i>	The summary type. If specified, the column values for parent nodes are computed from the children. Supported summary types are: <ul style="list-style-type: none"> <i>{+}</i> Sum numbers in this column. <i>{+;%.1f}</i> Like '+', but format result with '%.1f'. <i>{\$}</i> Currency, short for '+;%.2f'. <i>{:}</i> Sum times, HH:MM, plain numbers are hours. <i>{X}</i> Checkbox status, '[X]' if all children are '[X]'. <i>{X/}</i> Checkbox status, '[n/m]'. <i>{X%}</i> Checkbox status, '[n%]'. <i>{min}</i> Smallest number in column. <i>{max}</i> Largest number. <i>{mean}</i> Arithmetic mean of numbers. <i>{:min}</i> Smallest time value in column. <i>{:max}</i> Largest time value. <i>{:mean}</i> Arithmetic mean of time values. <i>{@min}</i> Minimum age (in days/hours/mins/seconds). <i>{@max}</i> Maximum age (in days/hours/mins/seconds). <i>{@mean}</i> Arithmetic mean of ages (in days/hours/mins/seconds). <i>{est+}</i> Add low-high estimates.

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

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

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

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

```
:COLUMNS: %25ITEM %9Approved(Approved?){X} %0wner %11Status \1
```

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

```

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

```

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

7.5.2 Using column view

Turning column view on and off

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

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

r org-columns-redo
 Recreate the column view, to include recent changes made in the buffer.

g org-columns-redo
 Same as *r*.

q org-columns-quit
 Exit column view.

Editing values

left right up down

Move through the column view from field to field.

S-left/right

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

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

n org-columns-next-allowed-value

p org-columns-previous-allowed-value

Same as *S-left/right*

<code>e</code>	<code>org-columns-edit-value</code> Edit the property at point. For the special properties, this will invoke the same interface that you normally use to change that property. For example, when editing a TAGS property, the tag completion or fast selection interface will pop up.
<code>C-c C-c</code>	<code>org-columns-set-tags-or-toggle</code> When there is a checkbox at point, toggle it.
<code>v</code>	<code>org-columns-show-value</code> View the full value of this property. This is useful if the width of the column is smaller than that of the value.
<code>a</code>	<code>org-columns-edit-allowed</code> Edit the list of allowed values for this property. If the list is found in the hierarchy, the modified values is stored there. If no list is found, the new value is stored in the first entry that is part of the current column view.

Modifying the table structure

<code><</code>	<code>org-columns-narrow</code>
<code>></code>	<code>org-columns-widen</code>
	Make the column narrower/wider by one character.
<code>S-M-right</code>	<code>org-columns-new</code>
	Insert a new column, to the left of the current column.
<code>S-M-left</code>	<code>org-columns-delete</code>
	Delete the current column.

7.5.3 カラム表示の保存

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

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

#+END:
```

This dynamic block has the following parameters:

<code>:id</code>	This is the most important parameter. Column view is a feature that is often localized to a certain (sub)tree, and the capture block might be at a different location in the file. To identify the tree whose view to capture, you can use 4 values:
<code>local</code>	use the tree in which the capture block is located
<code>global</code>	make a global view, including all headings in the file
<code>"file:path-to-file"</code>	
	run column view at the top of this file
<code>"ID"</code>	call column view in the tree that has an <code>:ID:</code> property with the value <i>label</i> . You can use <i>M-x org-id-copy</i> to create a globally unique ID for the current entry and copy it to the kill-ring.

:hlines When **t**, insert an hline after every line. When a number *N*, insert an hline before each headline with level $\leq N$.

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

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

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

The following commands insert or update the dynamic block:

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

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

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

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

An alternative way to capture and process property values into a table is provided by Eric Schulte's '**org-collector.el**' which is a contributed package². It provides a general API to collect properties from entries in a certain scope, and arbitrary Lisp expressions to process these values before inserting them into a table or a dynamic block.

7.6 The Property API

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

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

8 日付と時刻

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

8.1 Timestamps, deadlines, and scheduling

A timestamp is a specification of a date (possibly with a time or a range of times) in a special format, either ‘<2003-09-16 Tue>’ or ‘<2003-09-16 Tue 09:39>’ or ‘<2003-09-16 Tue 12:00-12:30>’¹. A timestamp can appear anywhere in the headline or body of an Org tree entry. Its presence causes entries to be shown on specific dates in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 92). We distinguish:

Plain timestamp; Event; Appointment

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

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

Timestamp with repeater interval

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

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

Diary-style sexp entries

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

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

Time/Date range

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

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

Inactive timestamp

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

¹ This is inspired by the standard ISO 8601 date/time format. To use an alternative format, see Section 8.2.2 [Custom time format], page 69.

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

8.2 Creating timestamps

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

<i>C-c .</i>	<i>org-time-stamp</i>
Prompt for a date and insert a corresponding timestamp. When the cursor is at an existing timestamp in the buffer, the command is used to modify this timestamp instead of inserting a new one. When this command is used twice in succession, a time range is inserted.	
<i>C-c !</i>	<i>org-time-stamp-inactive</i>
Like <i>C-c .</i> , but insert an inactive timestamp that will not cause an agenda entry.	
<i>C-u C-c .</i>	
<i>C-u C-c !</i>	Like <i>C-c .</i> and <i>C-c !</i> , but use the alternative format which contains date and time. The default time can be rounded to multiples of 5 minutes, see the option <i>org-time-stamp-rounding-minutes</i> .
<i>C-c <</i>	<i>org-date-from-calendar</i>
Insert a timestamp corresponding to the cursor date in the Calendar.	
<i>C-c ></i>	<i>org-goto-calendar</i>
Access the Emacs calendar for the current date. If there is a timestamp in the current line, go to the corresponding date instead.	
<i>C-c C-o</i>	<i>org-open-at-point</i>
Access the agenda for the date given by the timestamp or -range at point (see Section 10.3.1 [Weekly/daily agenda], page 92).	
<i>S-left</i>	<i>org-timestamp-down-day</i>
<i>S-right</i>	<i>org-timestamp-up-day</i>
Change date at cursor by one day. These key bindings conflict with shift-selection and related modes (see Section 15.10.2 [Conflicts], page 184).	
<i>S-up</i>	<i>org-timestamp-up</i>
<i>S-down</i>	<i>org-timestamp-down-down</i>
Change the item under the cursor in a timestamp. The cursor can be on a year, month, day, hour or minute. When the timestamp contains a time range like '15:30-16:30', modifying the first time will also shift the second, shifting the time block with constant length. To change the length, modify the second time. Note that if the cursor is in a headline and not at a timestamp, these same keys modify the priority of an item. (see Section 5.4 [Priorities], page 50). The key bindings also conflict with shift-selection and related modes (see Section 15.10.2 [Conflicts], page 184).	
<i>C-c C-y</i>	<i>org-evaluate-time-range</i>
Evaluate a time range by computing the difference between start and end. With a prefix argument, insert result after the time range (in a table: into the following column).	

8.2.1 The date/time prompt

When Org-mode prompts for a date/time, the default is shown in default date/time format, and the prompt therefore seems to ask for a specific format. But it will in fact accept any string containing some date and/or time information, and it is really smart about interpreting your input. You can, for example, use `C-y` to paste a (possibly multi-line) string copied from an email message. Org-mode will find whatever information is in there and derive anything you have not specified from the *default date and time*. The default is usually the current date and time, but when modifying an existing timestamp, or when entering the second stamp of a range, it is taken from the stamp in the buffer. When filling in information, Org-mode assumes that most of the time you will want to enter a date in the future: if you omit the month/year and the given day/month is *before* today, it will assume that you mean a future date². If the date has been automatically shifted into the future, the time prompt will show this with ‘(=>F).’

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

```

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

```

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

```

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

```

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

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

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

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

Parallel to the minibuffer prompt, a calendar is popped up³. When you exit the date prompt, either by clicking on a date in the calendar, or by pressing RET, the date selected in the calendar will be combined with the information entered at the prompt. You can control the calendar fully from the minibuffer:

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

The actions of the date/time prompt may seem complex, but I assure you they will grow on you, and you will start getting annoyed by pretty much any other way of entering a date/time out there. To help you understand what is going on, the current interpretation of your input will be displayed live in the minibuffer⁴.

8.2.2 Custom time format

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

`C-c C-x C-t` `org-toggle-time-stamp-overlays`
Toggle the display of custom formats for dates and times.

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

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

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

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

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

8.3 Deadlines and scheduling

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

DEADLINE

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

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

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

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

SCHEDULED

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

The headline will be listed under the given date⁵. In addition, a reminder that the scheduled date has passed will be present in the compilation for *today*, until the entry is marked DONE, i.e. the task will automatically be forwarded until completed.

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

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

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

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

8.3.1 Inserting deadlines or schedules

The following commands allow you to quickly insert⁶ a deadline or to schedule an item:

<i>C-c C-d</i>	org-deadline
Insert ‘DEADLINE’ keyword along with a stamp. The insertion will happen in the line directly following the headline. When called with a prefix arg, an existing deadline will be removed from the entry. Depending on the variable <code>org-log-redeadline</code> ⁷ , a note will be taken when changing an existing deadline.	
<i>C-c C-s</i>	org-schedule
Insert ‘SCHEDULED’ keyword along with a stamp. The insertion will happen in the line directly following the headline. Any CLOSED timestamp will be removed. When called with a prefix argument, remove the scheduling date from the entry. Depending on the variable <code>org-log-reschedule</code> ⁸ , a note will be taken when changing an existing scheduling time.	
<i>C-c C-x C-k</i>	org-mark-entry-for-agenda-action
Mark the current entry for agenda action. After you have marked the entry like this, you can open the agenda or the calendar to find an appropriate date. With the cursor on the selected date, press <i>k s</i> or <i>k d</i> to schedule the marked item.	
<i>C-c / d</i>	org-check-deadlines
Create a sparse tree with all deadlines that are either past-due, or which will become due within <code>org-deadline-warning-days</code> . With <i>C-u</i> prefix, show all deadlines in the file. With a numeric prefix, check that many days. For example, <i>C-1 C-c / d</i> shows all deadlines due tomorrow.	
<i>C-c / b</i>	org-check-before-date
Sparse tree for deadlines and scheduled items before a given date.	
<i>C-c / a</i>	org-check-after-date
Sparse tree for deadlines and scheduled items after a given date.	

8.3.2 Repeated tasks

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

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

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

⁶ The ‘SCHEDULED’ and ‘DEADLINE’ dates are inserted on the line right below the headline. Don’t put any text between this line and the headline.

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

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

Deadlines and scheduled items produce entries in the agenda when they are over-due, so it is important to be able to mark such an entry as completed once you have done so. When you mark a DEADLINE or a SCHEDULE with the TODO keyword DONE, it will no longer produce entries in the agenda. The problem with this is, however, that then also the *next* instance of the repeated entry will not be active. Org-mode deals with this in the following way: When you try to mark such an entry DONE (using `C-c C-t`), it will shift the base date of the repeating timestamp by the repeater interval, and immediately set the entry state back to TODO⁹. In the example above, setting the state to DONE would actually switch the date like this:

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

A timestamp¹⁰ will be added under the deadline, to keep a record that you actually acted on the previous instance of this deadline.

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

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

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

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

An alternative to using a repeater is to create a number of copies of a task subtree, with dates shifted in each copy. The command `C-c C-x c` was created for this purpose, it is described in Section 2.5 [Structure editing], page 9.

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

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

8.4 Clocking work time

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

To save the clock history across Emacs sessions, use

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

When you clock into a new task after resuming Emacs, the incomplete clock¹¹ will be found (see Section 8.4.3 [Resolving idle time], page 76) and you will be prompted about what to do with it.

8.4.1 Clocking commands

C-c C-x C-i **org-clock-in**
 Start the clock on the current item (clock-in). This inserts the **CLOCK** keyword together with a timestamp. If this is not the first clocking of this item, the multiple **CLOCK** lines will be wrapped into a **:LOGBOOK:** drawer (see also the variable **org-clock-into-drawer**). When called with a **C-u** prefix argument, select the task from a list of recently clocked tasks. With two **C-u C-u** prefixes, clock into the task at point and mark it as the default task. The default task will always be available when selecting a clocking task, with letter **d**.
 While the clock is running, the current clocking time is shown in the mode line, along with the title of the task. The clock time shown will be all time ever clocked for this task and its children. If the task has an effort estimate (see Section 8.5 [Effort estimates], page 77), the mode line displays the current clocking time against it¹². If the task is a repeating one (see Section 8.3.2 [Repeated tasks], page 71), only the time since the last reset of the task¹³ will be shown. More control over what time is shown can be exercised with the **CLOCK_MODELINE_TOTAL** property. It may have the values **current** to show only the current clocking instance, **today** to show all time clocked on this tasks today (see also the variable **org-extend-today-until**), **all** to include all time, or **auto** which is the default¹⁴.
 Clicking with **mouse-1** onto the mode line entry will pop up a menu with clocking options.

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

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

¹² To add an effort estimate “on the fly”, hook a function doing this to **org-clock-in-prepare-hook**.

¹³ as recorded by the **LAST_REPEAT** property

¹⁴ See also the variable **org-clock-modeline-total**.

inserts it after the time range as ‘=> HH:MM’. See the variable `org-log-note-clock-out` for the possibility to record an additional note together with the clock-out timestamp¹⁵.

- `C-c C-x C-e` `org-clock-modify-effort-estimate`
 Update the effort estimate for the current clock task.
- `C-c C-c` or `C-c C-y` `org-evaluate-time-range`
 Recompute the time interval after changing one of the timestamps. This is only necessary if you edit the timestamps directly. If you change them with *S-cursor* keys, the update is automatic.
- `C-c C-t` `org-todo`
 Changing the TODO state of an item to DONE automatically stops the clock if it is running in this same item.
- `C-c C-x C-x` `org-clock-cancel`
 Cancel the current clock. This is useful if a clock was started by mistake, or if you ended up working on something else.
- `C-c C-x C-j` `org-clock-goto`
 Jump to the headline of the currently clocked in task. With a *C-u* prefix arg, select the target task from a list of recently clocked tasks.
- `C-c C-x C-d` `org-clock-display`
 Display time summaries for each subtree in the current buffer. This puts overlays at the end of each headline, showing the total time recorded under that heading, including the time of any subheadings. You can use visibility cycling to study the tree, but the overlays disappear when you change the buffer (see variable `org-remove-highlights-with-change`) or press `C-c C-c`.

The *l* key may be used in the timeline (see Section 10.3.4 [Timeline], page 97) and in the agenda (see Section 10.3.1 [Weekly/daily agenda], page 92) to show which tasks have been worked on or closed during a day.

8.4.2 The clock table

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

- `C-c C-x C-r` `org-clock-report`
 Insert a dynamic block (see Section A.6 [Dynamic blocks], page 192) containing a clock report as an Org-mode table into the current file. When the cursor is at an existing clock table, just update it. When called with a prefix argument, jump to the first clock report in the current document and update it.
- `C-c C-c` or `C-c C-x C-u` `org-dblock-update`
 Update dynamic block at point. The cursor needs to be in the `#+BEGIN` line of the dynamic block.
- `C-u C-c C-x C-u`
 Update all dynamic blocks (see Section A.6 [Dynamic blocks], page 192). This is useful if you have several clock table blocks in a buffer.

¹⁵ The corresponding in-buffer setting is: `#+STARTUP: lognoteclock-out`

*S-left**S-right***org-clocktable-try-shift**

Shift the current **:block** interval and update the table. The cursor needs to be in the **#+BEGIN: clocktable** line for this command. If **:block** is **today**, it will be shifted to **today-1** etc.

Here is an example of the frame for a clock table as it is inserted into the buffer with the **C-c C-x C-r** command:

```
#+BEGIN: clocktable :maxlevel 2 :emphasize nil :scope file
#+END: clocktable
```

The ‘BEGIN’ line and specify a number of options to define the scope, structure, and formatting of the report. Defaults for all these options can be configured in the variable **org-clocktable-defaults**.

First there are options that determine which clock entries are to be selected:

:maxlevel	Maximum level depth to which times are listed in the table. Clocks at deeper levels will be summed into the upper level.
:scope	The scope to consider. This can be any of the following: <ul style="list-style-type: none"> nil the current buffer or narrowed region file the full current buffer subtree the subtree where the clocktable is located treeN the surrounding level <i>N</i> tree, for example tree3 tree the surrounding level 1 tree agenda all agenda files ("file"..) scan these files file-with-archives current file and its archives agenda-with-archives all agenda files, including archives
:block	The time block to consider. This block is specified either absolute, or relative to the current time and may be any of these formats: <ul style="list-style-type: none"> 2007-12-31 New year eve 2007 2007-12 December 2007 2007-W50 ISO-week 50 in 2007 2007-Q2 2nd quarter in 2007 2007 the year 2007 today, yesterday, today-N a relative day thisweek, lastweek, thisweek-N a relative week thismonth, lastmonth, thismonth-N a relative month thisyear, lastyear, thisyear-N a relative year Use <i>S-left/right</i> keys to shift the time interval.
:tstart	A time string specifying when to start considering times.
:tend	A time string specifying when to stop considering times.
:step	week or day , to split the table into chunks. To use this, :block or :tstart , :tend are needed.
:stepskip0	Do not show steps that have zero time.
:fileskip0	Do not show table sections from files which did not contribute.
:tags	A tags match to select entries that should contribute.

Then there are options which determine the formatting of the table. There options are interpreted by the function `org-clocktable-write-default`, but you can specify your own function using the `:formatter` parameter.

<code>:emphasize</code>	When <code>t</code> , emphasize level one and level two items.
<code>:lang</code>	Language ¹⁶ to use for descriptive cells like "Task".
<code>:link</code>	Link the item headlines in the table to their origins.
<code>:narrow</code>	An integer to limit the width of the headline column in the org table. If you write it like '50!', then the headline will also be shortened in export.
<code>:indent</code>	Indent each headline field according to its level.
<code>:tcolumns</code>	Number of columns to be used for times. If this is smaller than <code>:maxlevel</code> , lower levels will be lumped into one column.
<code>:level</code>	Should a level number column be included?
<code>:compact</code>	Abbreviation for <code>:level nil :indent t :narrow 40! :tcolumns 1</code> All are overwritten except if there is an explicit <code>:narrow</code>
<code>:timestamp</code>	A timestamp for the entry, when available. Look for SCHEDULED, DEADLINE, TIMESTAMP and TIMESTAMP_IA, in this order.
<code>:formula</code>	Content of a <code>#+TBLFM</code> line to be added and evaluated. As a special case, <code>:formula %</code> adds a column with % time. If you do not specify a formula here, any existing formula below the clock table will survive updates and be evaluated.
<code>:formatter</code>	A function to format clock data and insert it into the buffer.

To get a clock summary of the current level 1 tree, for the current day, you could write

```
#+BEGIN: clocktable :maxlevel 2 :block today :scope tree1 :link t
#+END: clocktable
```

and to use a specific time range you could write¹⁷

```
#+BEGIN: clocktable :tstart "<2006-08-10 Thu 10:00>"
                  :tend "<2006-08-10 Thu 12:00>"
#+END: clocktable
```

A summary of the current subtree with % times would be

```
#+BEGIN: clocktable :scope subtree :link t :formula %
#+END: clocktable
```

A horizontally compact representation of everything clocked during last week would be

```
#+BEGIN: clocktable :scope agenda :block lastweek :compact t
#+END: clocktable
```

8.4.3 Resolving idle time

If you clock in on a work item, and then walk away from your computer—perhaps to take a phone call—you often need to “resolve” the time you were away by either subtracting it from the current clock, or applying it to another one.

¹⁶ Language terms can be set through the variable `org-clock-clocktable-language-setup`.

¹⁷ Note that all parameters must be specified in a single line—the line is broken here only to fit it into the manual.

By customizing the variable `org-clock-idle-time` to some integer, such as 10 or 15, Emacs can alert you when you get back to your computer after being idle for that many minutes¹⁸, and ask what you want to do with the idle time. There will be a question waiting for you when you get back, indicating how much idle time has passed (constantly updated with the current amount), as well as a set of choices to correct the discrepancy:

- k** To keep some or all of the minutes and stay clocked in, press **k**. Org will ask how many of the minutes to keep. Press **RET** to keep them all, effectively changing nothing, or enter a number to keep that many minutes.
- K** If you use the shift key and press **K**, it will keep however many minutes you request and then immediately clock out of that task. If you keep all of the minutes, this is the same as just clocking out of the current task.
- s** To keep none of the minutes, use **s** to subtract all the away time from the clock, and then check back in from the moment you returned.
- S** To keep none of the minutes and just clock out at the start of the away time, use the shift key and press **S**. Remember that using shift will always leave you clocked out, no matter which option you choose.
- C** To cancel the clock altogether, use **C**. Note that if instead of canceling you subtract the away time, and the resulting clock amount is less than a minute, the clock will still be canceled rather than clutter up the log with an empty entry.

What if you subtracted those away minutes from the current clock, and now want to apply them to a new clock? Simply clock in to any task immediately after the subtraction. Org will notice that you have subtracted time “on the books”, so to speak, and will ask if you want to apply those minutes to the next task you clock in on.

There is one other instance when this clock resolution magic occurs. Say you were clocked in and hacking away, and suddenly your cat chased a mouse who scared a hamster that crashed into your UPS’s power button! You suddenly lose all your buffers, but thanks to auto-save you still have your recent Org mode changes, including your last clock in.

If you restart Emacs and clock into any task, Org will notice that you have a dangling clock which was never clocked out from your last session. Using that clock’s starting time as the beginning of the unaccounted-for period, Org will ask how you want to resolve that time. The logic and behavior is identical to dealing with away time due to idleness; it’s just happening due to a recovery event rather than a set amount of idle time.

You can also check all the files visited by your Org agenda for dangling clocks at any time using *M-x org-resolve-clocks*.

8.5 Effort estimates

If you want to plan your work in a very detailed way, or if you need to produce offers with quotations of the estimated work effort, you may want to assign effort estimates to entries.

¹⁸ On computers using Mac OS X, idleness is based on actual user idleness, not just Emacs’ idle time. For X11, you can install a utility program ‘`x11idle.c`’, available in the UTILITIES directory of the Org git distribution, to get the same general treatment of idleness. On other systems, idle time refers to Emacs idle time only.

If you are also clocking your work, you may later want to compare the planned effort with the actual working time, a great way to improve planning estimates. Effort estimates are stored in a special property ‘**Effort**’¹⁹. You can set the effort for an entry with the following commands:

C-c C-x e **org-set-effort**
 Set the effort estimate for the current entry. With a numeric prefix argument, set it to the Nth allowed value (see below). This command is also accessible from the agenda with the **e** key.

C-c C-x C-e **org-clock-modify-effort-estimate**
 Modify the effort estimate of the item currently being clocked.

Clearly the best way to work with effort estimates is through column view (see Section 7.5 [Column view], page 61). You should start by setting up discrete values for effort estimates, and a **COLUMNS** format that displays these values together with clock sums (if you want to clock your time). For a specific buffer you can use

```
#+PROPERTY: Effort_ALL 0 0:10 0:30 1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00
#+COLUMNS: %40ITEM(Task) %17Effort(Estimated Effort){:} %CLOCKSUM
```

or, even better, you can set up these values globally by customizing the variables **org-global-properties** and **org-columns-default-format**. In particular if you want to use this setup also in the agenda, a global setup may be advised.

The way to assign estimates to individual items is then to switch to column mode, and to use **S-right** and **S-left** to change the value. The values you enter will immediately be summed up in the hierarchy. In the column next to it, any clocked time will be displayed.

If you switch to column view in the daily/weekly agenda, the effort column will summarize the estimated work effort for each day²⁰, and you can use this to find space in your schedule. To get an overview of the entire part of the day that is committed, you can set the option **org-agenda-columns-add-appointments-to-effort-sum**. The appointments on a day that take place over a specified time interval will then also be added to the load estimate of the day.

Effort estimates can be used in secondary agenda filtering that is triggered with the **/** key in the agenda (see Section 10.5 [Agenda commands], page 101). If you have these estimates defined consistently, two or three key presses will narrow down the list to stuff that fits into an available time slot.

8.6 Taking notes with a relative timer

When taking notes during, for example, a meeting or a video viewing, it can be useful to have access to times relative to a starting time. Org provides such a relative timer and make it easy to create timed notes.

C-c C-x . **org-timer**
 Insert a relative time into the buffer. The first time you use this, the timer will be started. When called with a prefix argument, the timer is restarted.

¹⁹ You may change the property being used with the variable **org-effort-property**.

²⁰ Please note the pitfalls of summing hierarchical data in a flat list (see Section 10.8 [Agenda column view], page 115).

- C-c C-x -** **org-timer-item**
 Insert a description list item with the current relative time. With a prefix argument, first reset the timer to 0.
- M-RET** **org-insert-heading**
 Once the timer list is started, you can also use **M-RET** to insert new timer items.
- C-c C-x ,** Pause the timer, or continue it if it is already paused (**org-timer-pause-or-continue**).
- C-u C-c C-x ,**
 Stop the timer. After this, you can only start a new timer, not continue the old one. This command also removes the timer from the mode line.
- C-c C-x 0** **org-timer-start**
 Reset the timer without inserting anything into the buffer. By default, the timer is reset to 0. When called with a **C-u** prefix, reset the timer to specific starting offset. The user is prompted for the offset, with a default taken from a timer string at point, if any. So this can be used to restart taking notes after a break in the process. When called with a double prefix argument **C-u C-u**, change all timer strings in the active region by a certain amount. This can be used to fix timer strings if the timer was not started at exactly the right moment.

8.7 カウントダウンタイマ

Calling **org-timer-set-timer** from an Org-mode buffer runs a countdown timer. Use ; from agenda buffers, **C-c C-x ;** everywhere else.

org-timer-set-timer prompts the user for a duration and displays a countdown timer in the modeline. **org-timer-default-timer** sets the default countdown value. Giving a prefix numeric argument overrides this default value.

9 Capture - Refile - Archive

An important part of any organization system is the ability to quickly capture new ideas and tasks, and to associate reference material with them. Org does this using a process called *capture*. It also can store files related to a task (*attachments*) in a special directory. Once in the system, tasks and projects need to be moved around. Moving completed project trees to an archive file keeps the system compact and fast.

9.1 Capture

Org's method for capturing new items is heavily inspired by John Wiegley excellent remember package. Up to version 6.36 Org used a special setup for 'remember.el'. 'org-remember.el' is still part of Org-mode for backward compatibility with existing setups. You can find the documentation for org-remember at <http://orgmode.org/org-remember.pdf>.

The new capturing setup described here is preferred and should be used by new users. To convert your `org-remember-templates`, run the command

```
M-x org-capture-import-remember-templates RET
```

and then customize the new variable with `M-x customize-variable org-capture-templates`, check the result, and save the customization. You can then use both remember and capture until you are familiar with the new mechanism.

Capture lets you quickly store notes with little interruption of your work flow. The basic process of capturing is very similar to remember, but Org does enhance it with templates and more.

9.1.1 Setting up capture

The following customization sets a default target file for notes, and defines a global key¹ for capturing new material.

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

9.1.2 Using capture

`C-c c` `org-capture`
Call the command `org-capture`. Note that this keybinding is global and not active by default - you need to install it. If you have templates defined see Section 9.1.3 [Capture templates], page 81, it will offer these templates for selection or use a new Org outline node as the default template. It will insert the template into the target file and switch to an indirect buffer narrowed to this new node. You may then insert the information you want.

`C-c C-c` `org-capture-finalize`
Once you have finished entering information into the capture buffer, `C-c C-c` will return you to the window configuration before the capture process, so that you can resume your work without further distraction. When called with a prefix arg, finalize and then jump to the captured item.

¹ Please select your own key, `C-c c` is only a suggestion.

C-c C-w **org-capture-refile**
 Finalize the capture process by refile (see Section 9.5 [Refiling notes], page 87) the note to a different place. Please realize that this is a normal refile command that will be executed—so the cursor position at the moment you run this command is important. If you have inserted a tree with a parent and children, first move the cursor back to the parent. Any prefix argument given to this command will be passed on to the **org-refile** command.

C-c C-k **org-capture-kill**
 Abort the capture process and return to the previous state.

You can also call **org-capture** in a special way from the agenda, using the **k c** key combination. With this access, any timestamps inserted by the selected capture template will default to the cursor date in the agenda, rather than to the current date.

To find the locations of the last stored capture, use **org-capture** with prefix commands:

C-u C-c c
 Visit the target location of a capture template. You get to select the template in the usual way.

C-u C-u C-c c
 Visit the last stored capture item in its buffer.

9.1.3 Capture templates

You can use templates for different types of capture items, and for different target locations. The easiest way to create such templates is through the customize interface.

C-c c C Customize the variable **org-capture-templates**.

Before we give the formal description of template definitions, let's look at an example. Say you would like to use one template to create general TODO entries, and you want to put these entries under the heading 'Tasks' in your file '**~/org/gtd.org**'. Also, a date tree in the file '**journal.org**' should capture journal entries. A possible configuration would look like:

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

If you then press **C-c c t**, Org will prepare the template for you like this:

```
* TODO
[[file:link to where you initiated capture]]
```

During expansion of the template, **%a** has been replaced by a link to the location from where you called the capture command. This can be extremely useful for deriving tasks from emails, for example. You fill in the task definition, press **C-c C-c** and Org returns you to the same place where you started the capture process.

To define special keys to capture to a particular template without going through the interactive template selection, you can create your key binding like this:

```
(define-key global-map "\C-cx"
  (lambda () (interactive) (org-capture nil "x")))
```

9.1.3.1 Template elements

Now let's look at the elements of a template definition. Each entry in `org-capture-templates` is a list with the following items:

keys The keys that will select the template, as a string, characters only, for example "a" for a template to be selected with a single key, or "bt" for selection with two keys. When using several keys, keys using the same prefix key must be sequential in the list and preceded by a 2-element entry explaining the prefix key, for example

("b" "Templates for marking stuff to buy")

If you do not define a template for the `C` key, this key will be used to open the customize buffer for this complex variable.

description

A short string describing the template, which will be shown during selection.

type

The type of entry, a symbol. Valid values are:

entry An Org-mode node, with a headline. Will be filed as the child of the target entry or as a top-level entry. The target file should be an Org-mode file.

item A plain list item, placed in the first plain list at the target location. Again the target file should be an Org file.

checkboxitem

A checkbox item. This only differs from the plain list item by the default template.

table-line

a new line in the first table at the target location. Where exactly the line will be inserted depends on the properties `:prepend` and `:table-line-pos` (see below).

plain

Text to be inserted as it is.

target

Specification of where the captured item should be placed. In Org-mode files, targets usually define a node. Entries will become children of this node. Other types will be added to the table or list in the body of this node. Most target specifications contain a file name. If that file name is the empty string, it defaults to `org-default-notes-file`. A file can also be given as a variable, function, or Emacs Lisp form.

Valid values are:

(file "path/to/file")

Text will be placed at the beginning or end of that file.

(id "id of existing org entry")

Filing as child of this entry, or in the body of the entry.

(file+headline "path/to/file" "node headline")

Fast configuration if the target heading is unique in the file.

(file+olp "path/to/file" "Level 1 heading" "Level 2" ...)

For non-unique headings, the full path is safer.

(file+regexp "path/to/file" "regexp to find location")

Use a regular expression to position the cursor.

(file+datetree "path/to/file")

Will create a heading in a date tree for today's date.

(file+datetree+prompt "path/to/file")

Will create a heading in a date tree, but will prompt for the date.

(file+function "path/to/file" function-finding-location)

A function to find the right location in the file.

(clock) File to the entry that is currently being clocked.

(function function-finding-location)

Most general way, write your own function to find both file and location.

template The template for creating the capture item. If you leave this empty, an appropriate default template will be used. Otherwise this is a string with escape codes, which will be replaced depending on time and context of the capture call. The string with escapes may be loaded from a template file, using the special syntax (file "path/to/template"). See below for more details.

properties The rest of the entry is a property list of additional options. Recognized properties are:

:prepend Normally new captured information will be appended at the target location (last child, last table line, last list item...). Setting this property will change that.

:immediate-finish

When set, do not offer to edit the information, just file it away immediately. This makes sense if the template only needs information that can be added automatically.

:empty-lines

Set this to the number of lines to insert before and after the new item. Default 0, only common other value is 1.

:clock-in

Start the clock in this item.

:clock-keep

Keep the clock running when filing the captured entry.

:clock-resume

If starting the capture interrupted a clock, restart that clock when finished with the capture. Note that **:clock-keep** has precedence over **:clock-resume**. When setting both to **t**, the current clock will run and the previous one will not be resumed.

:unnarrowed

Do not narrow the target buffer, simply show the full buffer. Default is to narrow it so that you only see the new material.

:kill-buffer

If the target file was not yet visited when capture was invoked, kill the buffer again after capture is completed.

9.1.3.2 テンプレートの拡張

In the template itself, special %-escapes² allow dynamic insertion of content:

%{prompt}	prompt the user for a string and replace this sequence with it. You may specify a default value and a completion table with %{prompt default completion2 completion3...} The arrow keys access a prompt-specific history.
%a	annotation, normally the link created with org-store-link
%A	like %a , but prompt for the description part
%i	initial content, the region when capture is called while the region is active. The entire text will be indented like %i itself.
%t	timestamp, date only
%T	timestamp with date and time
%u, %U	like the above, but inactive timestamps
%^t	like %t , but prompt for date. Similarly %^T, %^u, %^U You may define a prompt like %{Birthday}t
%n	user name (taken from user-full-name)
%c	Current kill ring head.
%x	Content of the X clipboard.
%C	Interactive selection of which kill or clip to use.
%^L	Like %C , but insert as link.
%k	title of the currently clocked task
%K	link to the currently clocked task
%f	file visited by current buffer when org-capture was called
%F	like %f , but include full path
%^g	prompt for tags, with completion on tags in target file.
%^G	prompt for tags, with completion all tags in all agenda files.
%{prop}p	Prompt the user for a value for property <i>prop</i>
%:keyword	specific information for certain link types, see below
%[file]	insert the contents of the file given by <i>file</i>
%(sexp)	evaluate Elisp <i>sexp</i> and replace with the result

For specific link types, the following keywords will be defined³:

Link type	Available keywords
-----+	-----
bbdb	%:name %:company
irc	%:server %:port %:nick
vm, wl, mh, mew, rmail	%:type %:subject %:message-id
	%:from %:fromname %:fromaddress
	%:to %:toname %:toaddress
	%:date (message date header field)
	%:date-timestamp (date as active timestamp)
	%:date-timestamp-inactive (date as inactive timestamp)
	%:fromto (either "to NAME" or "from NAME") ⁴

² If you need one of these sequences literally, escape the % with a backslash.

³ If you define your own link types (see Section A.3 [Adding hyperlink types], page 186), any property you store with **org-store-link-props** can be accessed in capture templates in a similar way.

⁴ This will always be the other, not the user. See the variable **org-from-is-user-regexp**.

<code>gnus</code>		<code>:%group</code> , for messages also all email fields
<code>w3, w3m</code>		<code>:%url</code>
<code>info</code>		<code>:%file %:node</code>
<code>calendar</code>		<code>:%date</code>

To place the cursor after template expansion use:

`%?` After completing the template, position cursor here.

9.2 Attachments

It is often useful to associate reference material with an outline node/task. Small chunks of plain text can simply be stored in the subtree of a project. Hyperlinks (see Chapter 4 [Hyperlinks], page 34) can establish associations with files that live elsewhere on your computer or in the cloud, like emails or source code files belonging to a project. Another method is *attachments*, which are files located in a directory belonging to an outline node. Org uses directories named by the unique ID of each entry. These directories are located in the ‘`data`’ directory which lives in the same directory where your Org file lives⁵. If you initialize this directory with `git init`, Org will automatically commit changes when it sees them. The attachment system has been contributed to Org by John Wiegley.

In cases where it seems better to do so, you can also attach a directory of your choice to an entry. You can also make children inherit the attachment directory from a parent, so that an entire subtree uses the same attached directory.

The following commands deal with attachments:

C-c C-a	org-attach
The dispatcher for commands related to the attachment system. After these keys, a list of commands is displayed and you must press an additional key to select a command:	
a	org-attach-attach
Select a file and move it into the task’s attachment directory. The file will be copied, moved, or linked, depending on org-attach-method . Note that hard links are not supported on all systems.	
c/m/l	
Attach a file using the copy/move/link method. Note that hard links are not supported on all systems.	
n	org-attach-new
Create a new attachment as an Emacs buffer.	
z	org-attach-sync
Synchronize the current task with its attachment directory, in case you added attachments yourself.	
o	org-attach-open
Open current task’s attachment. If there is more than one, prompt for a file name first. Opening will follow the rules set by org-file-apps . For more details, see the information on following hyperlinks (see Section 4.4 [Handling links], page 36).	

⁵ If you move entries or Org files from one directory to another, you may want to configure **org-attach-directory** to contain an absolute path.

<i>O</i>	<code>org-attach-open-in-emacs</code> Also open the attachment, but force opening the file in Emacs.
<i>f</i>	<code>org-attach-reveal</code> Open the current task's attachment directory.
<i>F</i>	<code>org-attach-reveal-in-emacs</code> Also open the directory, but force using <code>dired</code> in Emacs.
<i>d</i>	<code>org-attach-delete-one</code> Select and delete a single attachment.
<i>D</i>	<code>org-attach-delete-all</code> Delete all of a task's attachments. A safer way is to open the directory in <code>dired</code> and delete from there.
<i>s</i>	<code>org-attach-set-directory</code> Set a specific directory as the entry's attachment directory. This works by putting the directory path into the <code>ATTACH_DIR</code> property.
<i>i</i>	<code>org-attach-set-inherit</code> Set the <code>ATTACH_DIR_INHERIT</code> property, so that children will use the same directory for attachments as the parent does.

9.3 RSS フィード

Org can add and change entries based on information found in RSS feeds and Atom feeds. You could use this to make a task out of each new podcast in a podcast feed. Or you could use a phone-based note-creating service on the web to import tasks into Org. To access feeds, configure the variable `org-feed-alist`. The docstring of this variable has detailed information. Here is just an example:

```
(setq org-feed-alist
  '(("Slashdot"
    "http://rss.slashdot.org/Slashdot/slashdot"
    "~/txt/org/feeds.org" "Slashdot Entries")))
```

will configure that new items from the feed provided by `rss.slashdot.org` will result in new entries in the file `~/org/feeds.org` under the heading `'Slashdot Entries'`, whenever the following command is used:

`C-c C-x g` `org-feed-update-all`
`C-c C-x g` Collect items from the feeds configured in `org-feed-alist` and act upon them.

`C-c C-x G` `org-feed-goto-inbox`
 Prompt for a feed name and go to the inbox configured for this feed.

Under the same headline, Org will create a drawer `'FEEDSTATUS'` in which it will store information about the status of items in the feed, to avoid adding the same item several times. You should add `'FEEDSTATUS'` to the list of drawers in that file:

```
#+DRAWERS: LOGBOOK PROPERTIES FEEDSTATUS
```

For more information, including how to read atom feeds, see `'org-feed.el'` and the docstring of `org-feed-alist`.

9.4 Protocols for external access

You can set up Org for handling protocol calls from outside applications that are passed to Emacs through the ‘`emacsserver`’. For example, you can configure bookmarks in your web browser to send a link to the current page to Org and create a note from it using capture (see Section 9.1 [Capture], page 80). Or you could create a bookmark that will tell Emacs to open the local source file of a remote website you are looking at with the browser. See <http://orgmode.org/worg/org-contrib/org-protocol.php> for detailed documentation and setup instructions.

9.5 Refiling notes

When reviewing the captured data, you may want to refile some of the entries into a different list, for example into a project. Cutting, finding the right location, and then pasting the note is cumbersome. To simplify this process, you can use the following special command:

`C-c C-w` `org-refile`

Refile the entry or region at point. This command offers possible locations for refileing the entry and lets you select one with completion. The item (or all items in the region) is filed below the target heading as a subitem. Depending on `org-reverse-note-order`, it will be either the first or last subitem.

By default, all level 1 headlines in the current buffer are considered to be targets, but you can have more complex definitions across a number of files. See the variable `org-refile-targets` for details. If you would like to select a location via a file-path-like completion along the outline path, see the variables `org-refile-use-outline-path` and `org-outline-path-complete-in-steps`. If you would like to be able to create new nodes as new parents for refileing on the fly, check the variable `org-refile-allow-creating-parent-nodes`. When the variable `org-log-refile`⁶ is set, a timestamp or a note will be recorded when an entry has been refiled.

`C-u C-c C-w`

Use the refile interface to jump to a heading.

`C-u C-u C-c C-w` `org-refile-goto-last-stored`

Jump to the location where `org-refile` last moved a tree to.

`C-2 C-c C-w`

Refile as the child of the item currently being clocked.

`C-0 C-c C-w` or `C-u C-u C-u C-c C-w`

`C-0 C-c C-w` or `C-u C-u C-u C-c C-w` `org-refile-cache-clear`

Clear the target cache. Caching of refile targets can be turned on by setting `org-refile-use-cache`. To make the command see new possible targets, you have to clear the cache with this command.

⁶ with corresponding `#+STARTUP` keywords `logrefile`, `lognoterefile`, and `nologrefile`

9.6 Archiving

When a project represented by a (sub)tree is finished, you may want to move the tree out of the way and to stop it from contributing to the agenda. Archiving is important to keep your working files compact and global searches like the construction of agenda views fast.

`C-c C-x C-a` `org-archive-subtree-default`
 Archive the current entry using the command specified in the variable `org-archive-default-command`.

9.6.1 Moving a tree to the archive file

The most common archiving action is to move a project tree to another file, the archive file.

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

`C-u C-c C-x C-s`
 Check if any direct children of the current headline could be moved to the archive. To do this, each subtree is checked for open TODO entries. If none are found, the command offers to move it to the archive location. If the cursor is *not* on a headline when this command is invoked, the level 1 trees will be checked.

The default archive location is a file in the same directory as the current file, with the name derived by appending ‘`_archive`’ to the current file name. For information and examples on how to change this, see the documentation string of the variable `org-archive-location`. There is also an in-buffer option for setting this variable, for example⁷:

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

If you would like to have a special ARCHIVE location for a single entry or a (sub)tree, give the entry an `:ARCHIVE:` property with the location as the value (see Chapter 7 [Properties and Columns], page 58).

When a subtree is moved, it receives a number of special properties that record context information like the file from where the entry came, its outline path the archiving time etc. Configure the variable `org-archive-save-context-info` to adjust the amount of information added.

9.6.2 ファイル内部でのアーカイブ

If you want to just switch off (for agenda views) certain subtrees without moving them to a different file, you can use the **ARCHIVE tag**.

A headline that is marked with the ARCHIVE tag (see Chapter 6 [Tags], page 54) stays at its location in the outline tree, but behaves in the following way:

⁷ For backward compatibility, the following also works: If there are several such lines in a file, each specifies the archive location for the text below it. The first such line also applies to any text before its definition. However, using this method is *strongly* deprecated as it is incompatible with the outline structure of the document. The correct method for setting multiple archive locations in a buffer is using properties.

- It does not open when you attempt to do so with a visibility cycling command (see Section 2.3 [Visibility cycling], page 7). You can force cycling archived subtrees with `C-TAB`, or by setting the option `org-cycle-open-archived-trees`. Also normal outline commands like `show-all` will open archived subtrees.
- During sparse tree construction (see Section 2.6 [Sparse trees], page 12), matches in archived subtrees are not exposed, unless you configure the option `org-sparse-tree-open-archived-trees`.
- During agenda view construction (see Chapter 10 [Agenda Views], page 90), the content of archived trees is ignored unless you configure the option `org-agenda-skip-archived-trees`, in which case these trees will always be included. In the agenda you can press `v a` to get archives temporarily included.
- Archived trees are not exported (see Chapter 12 [Exporting], page 125), only the headline is. Configure the details using the variable `org-export-with-archived-trees`.
- Archived trees are excluded from column view unless the variable `org-columns-skip-archived-trees` is configured to `nil`.

The following commands help manage the ARCHIVE tag:

- `C-c C-x a` `org-toggle-archive-tag`
 Toggle the ARCHIVE tag for the current headline. When the tag is set, the headline changes to a shadowed face, and the subtree below it is hidden.
- `C-u C-c C-x a`
 Check if any direct children of the current headline should be archived. To do this, each subtree is checked for open TODO entries. If none are found, the command offers to set the ARCHIVE tag for the child. If the cursor is *not* on a headline when this command is invoked, the level 1 trees will be checked.
- `C-TAB` `org-force-cycle-archived`
 Cycle a tree even if it is tagged with ARCHIVE.
- `C-c C-x A` `org-archive-to-archive-sibling`
 Move the current entry to the *Archive Sibling*. This is a sibling of the entry with the heading ‘Archive’ and the tag ‘ARCHIVE’. The entry becomes a child of that sibling and in this way retains a lot of its original context, including inherited tags and approximate position in the outline.

10 アジェンダビュー

Org-mode で作業した結果、TODO アイテム、タイムスタンプのついたアイテム、タグの付いた見出しなどが、1つのファイル、あるいはいくつものファイルにまたがって、撒き散らされることとなります。ある特定の日に重要な、実際に動いているアイテムやイベントの全体像を把握するためには、ひとつの管理された方法で、これらの情報を集めたり、並び替えたりしながら、表示することが必要です。

Org-mode では、いろいろな基準によってアイテムを選択することが可能であり、独立したバッファにそれらのアイテムを表示させることができます。7つの異なるビューのタイプが用意されています。:

- アジェンダ カレンダーのように指定した日付の情報を表示します、
- *TODO* リスト 未完了のアクションアイテムをカバーします、
- マッチビュー 関連づけられているタグやプロパティ、TODO の状態に基づいて見出しを表示します、
- タイムラインビュー 1つの Org-mode のファイルの中に含まれている全てのイベントを時間順のビューに表示します、
- a テキストの検索ビュー 複数のファイルの中かから、指定したキーワードを含んでいるすべてのエントリを表示します、
- a 詳細が未決定のプロジェクトビュー 現在作業が進んでいないプロジェクトを表示します。そして、
- カスタムビュー 特別な検索や異なるビューの組合せによるビューです。

抽出された情報は特別なアジェンダバッファに表示されます。このバッファはリードオンリーですが、オリジナルの Org-mode ファイルにジャンプしたり、オリジナルのファイルを間接的に編集することができます。

2つの変数によって、アジェンダバッファをどのように表示するか、アジェンダが存在したときに、ウインドウの設定を元に戻すかどうかをコントロールします。; `org-agenda-window-setup` と `org-agenda-restore-windows-after-quit`。

10.1 Agenda files

表示される情報は、通常すべてのアジェンダファイルから収集されます。アジェンダファイルは `org-agenda-files`¹ 変数にリスト化されたファイルが対象となります。もしもこのリストの中にディレクトリ名が記載されていたら、そのディレクトリの中にある `‘.org’` という拡張子がついた全てのファイルが、アジェンダファイルの対象となります。

したがって、たとえあなたが1つの Org-mode ファイルでしか作業をしていなくても、このファイルをそのリスト²に記載したことになるでしょう。`org-agenda-files`をカスタマイズすることが可能で、しかも以下に述べるコマンドを通して簡単な方法で維持することができます。

¹ もしもその変数の値がリストではなく、単独のファイル名の場合には、その外部ファイルの中に記載されているアジェンダファイルの名前となります。

² コマンド選択画面を使用しているときに、コマンドを選択する前に、<を押すと、編集中的のファイルに対するコマンドが制限されて、次のコマンド選択画面でコマンドが入力されるまで、`org-agenda-files`は無視されます。

C-c [**org-agenda-file-to-front**
 アジェンダファイルのリストに編集中のファイルを追加する。そのファイルは、リストの先頭に追加される。もしも既にリストに存在していたら、先頭に移動する。前置引数をつけることで、リストの最後に追加／移動する。

C-c] **org-remove-file**
 編集中のファイルをアジェンダファイルのリストから削除する。

C-' **org-cycle-agenda-files**
C-, アジェンダファイルのリストに従って、1つのファイルから次のファイルへと切り替える。

M-x org-iswitchb
iswitchbと似たようなインターフェースで Org-mode のバッファの間を切り替えるコマンド。

Org-mode メニューには、現時点のファイルのリストが含まれており、その中のファイルに移動するのに役立ちます。

もしもこのリストに載っているファイルではなく、作業中のアジェンダファイルに焦点をあてたかったり、リストにあるファイルのまさにひとつのファイルに焦点をあてたかったり、はたまたあるファイルの中のあるサブツリーに焦点をあてたかったりしたいときは、いくつかの方法が用意されています。単一のアジェンダコマンドとして、コマンド選択画面上 (see Section 10.2 [Agenda dispatcher], page 91) で **<** を 1 回ないし数回押すとよいのです。アジェンダの対象をある限定した期間に絞込むために以下のコマンドが用意されています。:

C-c C-x < **org-agenda-set-restriction-lock**
 アジェンダの対象を現在カーソルが置かれているサブツリーに固定的に制限します。前置引数をつけたり、ファイルの最初の見出しよりも前にカーソルが置かれているときには、アジェンダの対象範囲はファイル全体になります。この制約は **C-c C-x >** を実行して取り除くか、**<** または **>** をアジェンダのコマンド選択画面上で入力するまでは維持します。もしもウィンドウ上にアジェンダビューが表示されているならば、あたらしい制約が即座に効果を及ぼします。

C-c C-x > **org-agenda-remove-restriction-lock**
C-c C-x < で作成された固定する制限を削除します。

‘**speedbar.el**’を併用しているときは、Speedbar のフレームの中で以下のコマンドを使用することができます。

< in the speedbar frame **org-speedbar-set-agenda-restriction**
 Speedbar のフレームの中で、1つの Org-mode ファイルか、そのファイルのサブツリーの一つか、カーソルの置かれているアイテムに対応してアジェンダを恒久的に限定します。もしもアジェンダビューが表示されているウィンドウがあるならば、限定箇所が変更されると即座に反映する。

> in the speedbar frame **org-agenda-remove-restriction-lock**
 制限をふたたび解除する。

10.2 アジェンダのコマンド選択画面

グローバルなキーと結びついている、コマンド選択画面を通してそのビューは作成されます。—例えば、**C-c a** (see Section 1.2 [Installation], page 3) のように。以下のように、コマンド選択画面に

アクセスする方法として `C-c a` を想定しており、キーボードでコマンドにアクセスするためのリストが表示されています。`C-c a` を入力した後、コマンドを実行するために、次に入力する文字を要求します。コマンド選択画面では以下に記載するデフォルトのコマンドが提供されています。

- `a` カレンダーのようなアジェンダを作成します。(see Section 10.3.1 [Weekly/daily agenda], page 92)
- `t / T` すべての TODO アイテムのリストを作成します。(see Section 10.3.2 [Global TODO list], page 94)
- `m / M` タグの表記にマッチした見出しのリストを作成します。(see Section 10.3.3 [Matching tags and properties], page 95)
- `L` カレントバッファ用のタイムラインのビューを作成します。(see Section 10.3.4 [Timeline], page 97)
- `s` そのエントリーに存在するしないにかかわらず、and/or という正規表現によるキーワードの論理式で選択したエントリのリストを作成します。
- `/` すべてのアジェンダファイルと `org-agenda-text-search-extra-files` の中でリスト化されているファイルの中から正規表現を用いて検索します。これは Emacs の `multi-occur` というコマンドを使用します。前置引数をつけると、それぞれのマッチした行の状況の数をしていますことができます。デフォルトは 1 となっています。
- `# / !` 詳細が未決定のプロジェクトのリストを作成します。(see Section 10.3.6 [Stuck projects], page 98)
- `<` カレントバッファ³ に対してアジェンダコマンドを制限します。`<` を入力したあと、コマンドを選択するために文字を入力する必要があります。
- `<<` もしもアクティブなリージョンがあるときは、以下のようなアジェンダコマンドがそのリージョンに限定されます。一方、カレントのサブツリー⁴ に限定することもできます。`<<` を入力したあと、コマンドを選択する文字を入力する必要があります。

あなたは、あたかもデフォルトのコマンドのように、コマンド選択画面でアクセスするカスタムコマンドを定義することもできます。複数のブロックを同時に含めた拡張されたアジェンダバッファを作成する可能性を含んでいます。例えば週のアジェンダ、グローバルな TODO リスト、そして多数の特定タグの検索など。See Section 10.6 [Custom agenda views], page 110.

10.3 agenda に組み込まれているビュー

このセクションではビルトインビューについて説明します。

10.3.1 1 週間 / 1 日のアジェンダ

1 週間の / 1 日のアジェンダの目的は、その週あるいはその日のタスクをすべて表示して、紙のアジェンダのページのように、実行に移すことです。

³ 逆の互換性として、1 をカレントバッファを制限するために入力することもできます。

⁴ 逆の互換性として、カレントリージョンまたはカレントサブツリーに限定するために 0 を入力することもできます。

`C-c a a``org-agenda-list`

Org-mode のファイルのリストの中からその週の予定を収集するものです。予定はそれぞれの日に表示されます。(C-u 2 1 C-c a aのように) 前置引数に数字をつけて⁵ 表示する日数を設定することができます。

表示されるデフォルトの日数は、`org-agenda-span`(あるいは古くさくなってしまいましたが`org-agenda-ndays`) という変数で設定します。この変数は、アジェンダの中でデフォルトとして確認したい日数、あるいは、期間を示す `day`、`week`、`month` や `year` といった期間を示す名前をつけて設定します。

アジェンダバッファからリモートで編集するとは、例えば、アジェンダバッファの中でデッドラインやアポイントメントの日付を変更することができるという意味です。アジェンダバッファの中で利用できるコマンドは、Section 10.5 [Agenda commands], page 101 の中で一覧表にしています。

カレンダー／日記の統合

Emacs には、Edward M. Reingold によって開発されたカレンダーと日記の機能があります。カレンダーでは、国や文化の異なる祝祭日を備えた 3ヵ月分のカレンダーが表示されます。日記には記念日、月の満ち欠け、日の出日の入り、繰り返しの予定 (隔週、隔月) などを記録しておくことができます。このような機能は、Org-mode に対して大変補完的な関係にあります。日記と Org-mode の出力を結びつけることは大変有益です。

Emacs の日記から Org-mode のアジェンダに項目を落とし込むために、あなたは次のように変数を設定するだけです。

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

After that, everything will happen automatically. All diary 祝祭日や記念日などを含むすべての項目は、Org-mode で作成されるジェンダバッファに取り込むことができます。日記に記録されている項目を編集するために、アジェンダバッファ上で `SPC`、`TAB`、及び `RET` を入力することで、日記のファイルにジャンプすることができます。その日に新しいエントリーを挿入する `i` というコマンドはアジェンダバッファ上で動作します。あたかも、日の出日の入りの時刻を表示したり、月の満ち欠けの状態を表示したり、他の暦に変換するための、`S`、`M`、および `C` というコマンドと同様です。`c` はカレンダーとアジェンダの間を行ったり来たりすることができます。

もしもあなたが日記を `S` 式項目と祝祭日だけで使用しているのならば、上のような設定をするよりも、Org-mode ファイルに直接コピーしたり移動したりしたほうが手っ取り早いです。Org-mode は日記形式の `S` 式項目を評価し、しかもより早く、というのは、最初にカレンダーを表示するという負荷がかからないからです。`S` 式項目は左端から記述し、式の前にスペースが入ってはいけないことに注意してください。たとえば、ある Org-mode ファイルについての、以下にのべるセグメントが処理され、項目がアジェンダの中に作成されます。

```
* Birthdays and similar stuff
#+CATEGORY: Holiday
%%(org-calendar-holiday) ; special function for holiday names
#+CATEGORY: Ann
%%(diary-anniversary 5 14 1956)6 Arthur Dent is %d years old
%%(diary-anniversary 10 2 1869) Mahatma Gandhi would be %d years old
```

⁵ 逆方向の互換性のために、普遍的な前置引数 `C-u` をつけることでアジェンダ (予定表) より上に、TODO リストを書き出すことができます。この機能は軽視されており、専用の TODO リストやブロックアジェンダ (see Section 10.6.2 [Block agenda], page 111). をその代わりに利用することが多いです。

⁶ Note that the order of the arguments (month, day, year) depends on the setting of `calendar-date-style`.

Anniversaries from BBDB

もしも Big Brothers Database を使用して連絡先を管理しているのならば、あなたは先に述べたのと同様に、独立した Org-mode のファイルや日記のファイルに登録するよりも、BBDB の中に記念日を登録したいと考えるでしょう。Org-mode はこれもサポートしており、アジェンダの一部として BBDB の記念日を表示することができます。そのために必要なことは、以下のような記述をアジェンダファイルに行うことです。

```
* Anniversaries
:PROPERTIES:
:CATEGORY: Anniv
:END:
%%(org-bbdb-anniversaries)
```

それから BBDB のデータレコードのための記念日の定義に取り掛かることができます。基本的には、BBDB のレコードの中にカーソルを置いて、`C-o anniversary RET` を実行し、それから日付を YYYY-MM-DD または MM-DD の形式で記入し、半角スペースに続けて記念日の種類 ('birthday'、'wedding'、または定型句) のクラスを記入します。もしもクラスを省略した場合は、デフォルトでは 'birthday' であるとみなします。いくつかの例を書いてみました。'org-bbdb.el' ファイルの先頭のところにもう少し詳しい説明が書いてあります。

```
1973-06-22
06-22
1955-08-02 wedding
2008-04-14 %s released version 6.01 of org-mode, %d years ago
```

BBDB を変更したり、Emacs のセッションで最初にアジェンダを表示したとき後は、アジェンダの表示が少し遅くなるかもしれません。というのは Org-mode が記念日のハッシュデータを更新するからです。しかしながら、そのことについていうと非常に早いといえます。実際 Org-mode の日記ファイルに '%%(diary-anniversary)' のエントリーを長々と書き連ねた場合よりもずっと早いと言えるでしょう。

Appointment reminders

Org-mode は Emacs の予定を通知する機能と連携しています。あなたのアジェンダファイルに含まれているすべてのアポイントを追加するために、`org-agenda-to-appt` コマンドを使います。このコマンドはあなたの予定のリストにフィルターをかけ、特別なカテゴリーに属しているものや正規表現の検索に合致したものを追加します。詳細はドキュメント文字列を参照してください。

10.3.2 The global TODO list

グローバルな TODO リストには、形式を整えられ、1 つの場所に集められたすべての未完了の TODO アイテムが含まれています。

```
C-c a t                                org-todo-list
```

グローバルな TODO リストを表示します。これはすべてのアジェンダファイル (see Chapter 10 [Agenda Views], page 90) から TODO アイテムを 1 つのバッファに集約します。デフォルトでは、このアイテムのリストは DONE という状態ではないアイテムです。そのバッファは `agenda-mode` となり、そのバッファから TODO アイテムを直接調べたり操作したりするコマンドが用意されています (see Section 10.5 [Agenda commands], page 101)。上と似ていますが、指定した TODO キーワードと合致したものを表示します。同じことを前置引数をつけて `C-c a t` を実行することでも指定できます。キーワードの入力を促す指示が表示され、そして複数のキーワードを論理式 OR

という意味で「|」で区切って指定することができます。数字付きの前置引数をつけると `org-todo-keywords` 中の N 番目のキーワードを選択することができます。`r` キーをアジェンダバッファで使用するとバッファの再構成が行われます。たとえば `3 r` というように、前置引数をつけてこのコマンドを実行すると選択した TODO キーワードが変更することができます。もしも特定のキーワードを使って検索することが多い場合は、カスタムコマンドを定義することもできます (see Section 10.2 [Agenda dispatcher], page 91)。

特定の TODO キーワードと合致するものを検索するのは、タグ検索の 1 機能として行うこともできます (see Section 6.3 [Tag searches], page 56)。

リモートで TODO アイテムを編集するというこの意味は、1 つのキーを入力することで TODO エントリーの状態を変更できるということです。TODO リストの中で利用できるコマンドは Section 10.5 [Agenda commands], page 101 の記述を参考にしてください。

通常グローバルな TODO リストには、TODO キーワードのついたすべて見出しが表示されます。このリストは大変長いものになる場合もあります。それをコンパクトにするには 2 つの方法があります。

- TODO アイテムが、実行するために *scheduled* となっている、あるいは、もはや *open* となっている *deadline* (see Section 8.1 [Timestamps], page 66) を持っているかどうかを確認したい人もいるでしょう。`org-agenda-todo-ignore-scheduled`、`org-agenda-todo-ignore-deadlines`、`org-agenda-todo-ignore-timestamp` および／または `org-agenda-todo-ignore-with-date` という変数を設定し、グローバルな TODO リストから取り除くことができます。
- TODO アイテムがサブタスクにブレイクダウンされた下位のレベルを持っているかもしれません。そういった場合は、最上位の TODO の見出しを表示すれば十分で、グローバルなリストからは下位のレベルの項目は省略してもよい場合があります。そういったときは `org-agenda-todo-list-sublevels` 変数を設定することで可能となります

10.3.3 Matching tags and properties

アジェンダファイルの中の見出しに *tags* (see Chapter 6 [Tags], page 54) がついていた、あるいは属性 (see Chapter 7 [Properties and Columns], page 58) がついていたときは、このメタデータに基づいて見出しを選択し、アジェンダバッファに収集することができます。この項で述べている検索構文は `C-c / m` を用いたツリーの抽出を行うときも適用できます。

`C-c a m`

`org-tags-view`

一組のタグのセットに合致したすべての見出しのリストを作成します。選択の基準の入力を指示するコマンドでタグのついた論理式による表現で記入します。例えば、`‘+work+urgent-withboss’` あるいは `‘work|home’` というように (see Chapter 6 [Tags], page 54)。もしも特定の検索をよく行うならばそのためのカスタムコマンドを定義することができます (see Section 10.2 [Agenda dispatcher], page 91)。

`C-c a M`

`org-tags-view`

`C-c a m` と似ていますが、`not-DONE` の状態にある TODO アイテムの見出しから選択するもので、自動的にサブアイテムもチェックします (`org-tags-match-list-sublevels` 変数参照)。予定／期限のついたアイテムを除外するには `org-agenda-tags-todo-honor-ignore-options` の変数を参照してください。特定の TODO キーワードをタグの一致と一緒に指定することも可能です。Section 6.3 [Tag searches], page 56 を参照してください。

タグのリストで利用できるコマンドは Section 10.5 [Agenda commands], page 101 のところで説明しています。

Match syntax

検索文字列では AND の意味で '&'、OR の意味で '|' という論理式を使うことができます。'&' は '|' よりも強く結びつけます。括弧 () は現在準備されていません。検索のどの要素も、タグそのものか、正規表現でマッチしたタグか、あるいは PROPERTY OPERATOR VALUE のような属性値にアクセスして比較操作のできる値のいずれかになります。どの要素も '-' を先頭につけてそれ以外のものを表現するか、'+' を先頭につけてポジティブな選択を行う、というような糖衣構文（簡便な構文）で表現します。'&' で AND を取り扱うことは '+'、'-' で表現できるもののオプションです。下にタグだけをつかったいくつかの例を挙げておきました。

'+work-boss'

'work:' というタグがついているが、':boss:' というタグがついていない見出しを選択します。

'work|laptop'

'work:' または ':laptop:' というタグがついたものを選択します。

'work|laptop+night'

前の文と同じですが、':laptop:' の行には、同時に ':night:' というタグが付いている必要があります。

タグの代わりに、大括弧でくくられた正規表現により指定をすることもできます。例えば、'work+{^boss.*}' と指定すると、':work:' というタグのついた見出しで 'boss' という単語で *starting* するタグがついているものに一致します。

タグとマッチするものを探すと同時に属性 (see Chapter 7 [Properties and Columns], page 58) の検索をすることも可能です。属性としては実際の属性のほかに、他のメタデータで表現された特別な属性 (see Section 7.2 [Special properties], page 59) にも対応しています。例えば、そのエントリーの中の TODO キーワードで表現された TODO という「属性」。あるいは、そのエントリーの階層を示す LEVEL という「属性」などです。そのため、'+LEVEL=3+boss-TODO="DONE"' という検索式は、第 3 階層のすべての見出しの中で、'boss' というタグがついており、TODO キーワードが DONE では「ない」もののリストを表示します。org-odd-levels-only という設定がなされているバッファでは 'LEVEL' は * の数を数えるのではなく、'LEVEL=2' (2 番目) の階層は * が 3 つある階層が該当します。

いくつかの例を紹介します。

'work+TODO="WAITING"'

'work:' というタグがある TODO 行のうち、特に TODO キーワードが 'WAITING' となっている行を選択します。

'work+TODO="WAITING"|home+TODO="WAITING"'

work と home というタグがついている Waiting となっているタスク

属性の検索では、多数の異なる操作で属性の値をテストすることができます。複雑な例を挙げます。

```
+work-boss+PRIORITY="A"+Coffee="unlimited"+Effort<2 \
+With={Sarah\|Denny}+SCHEDULED>="<2008-10-11>"
```

比較のタイプは比較の値がどのように書かれているかによります。

- 比較する値が普通の数字ならば、数値の比較が行われ、'<'、'='、'>'、'<='、'>='、および '<>' という操作が可能です。

- 比較する対象がダブルクォーテーションで囲まれている場合は、文字列の比較が行われ、前項と同じ操作が可能です。
- もしも比較対象が、('DEADLINE<="<2008-12-24 18:30>"'のように)、ダブルクォーテーションおよび角括弧<>で囲まれていた場合は、両方の値が Org-mode 流の標準的な日付・時刻の指定であると仮定し、それにそって比較を行います。いくつかの特別な値があります。"<now>"は(時刻も含めた)現在を示し、"<today>"、"<tomorrow>"はそれらの日の 0:00 つまり、時刻の指定がないことを表します。同様に、"<+5d>"または"<-2m>"というような文字列は、それぞれ日、週、月、年を示す、d、w、m、y という単位がついているものとして使用されます。
- もしも比較対象が中括弧 {} でくくられていて、正規表現での比較がなされるときは、'='は一致していることを示し、'<'は一致していないことを示します。

そのため、例に掲げた検索文字列の意味は、':work:'というタグがつけられているが、':boss:'というタグはついておらず、また、優先順位の値が'A'であり、':Coffee:'が'unlimited'という値であり、'Effort' 属性が数値で2より小さく、':With:'の値が'Sarah\|Denny'であり、スケジュールが2008年10月11日もしくはそれ以降に予約されたものを示しています。

TODO、LEVEL、CATEGORY を検索するときは短時間ですみます。それ以外の属性を検索するときはいささか時間がかかります。しかしながら、一度高い代償を払って1つのプロパティを検索したら、他の属性を追加して再び検索するときは安くあがります。

検索の際に Org-mode で属性の継承という機能を使用するように設定することができますが、相当検索スピードが落ちることを覚悟してください。詳細は Section 7.4 [Property inheritance], page 60 参照。

逆互換として、さらにまたタイプのスピードを上げるために、検索において TODO の状態をテストする別の方法があります。このためには、検索文字列(それは'|'で結合された複数の用語が含まれていると思いますが)のタグ・属性検索の部分を '/' を使って終了させ、TODO キーワードを論理式で結んで指定します。その構文はタグの検索で使ったのと似ていますが、よく考えて適用する必要があります。例えば、複数の TODO キーワードが存在することを検索するには論理式の AND で結びつけても意味がありません。しかしながら、*negative selection* (存在しないことを選択する場合)では「AND」で結合することは意味を持ちます。これを確かめるには、実際にいくつかの TODO キーワードで、C-c a M を用いて確認するだけです(そのほうがスピードアップできます)。あるいはスラッシュのあとに '!' を記入して同時に TODO の部分を開始します。C-c a M または '/!' を使用したときは、DONE の状態にある TODO キーワードを検索することはできません。例えば、

'work/WAITING'

'work+TODO="WAITING"'と同じ

'work/!-WAITING-NEXT'

' :work:' を選択。ただし TODO 行では 'WAITING' と 'NEXT' のどちらのタグもついていないもの

'work/!+WAITING|+NEXT'

' :work:' を選択。TODO 行に 'WAITING' か 'NEXT' かどちらかのタグがついているもの。

10.3.4 Timeline for a single file

タイムラインはひとつの Org-mode ファイルの中から *time-sorted view* (時間順のビュー) ですべてのタイムスタンプのついたアイテムをまとめて表示します。このコマンドの主な目的は、あるプロジェクトに含まれているイベント全体の概要をつかむためにあります。

C-c a L**org-timeline**

すべてのタイムスタンプの付いたアイテムについて、Org-mode ファイルの中で時間順のビューを提供します。**C-u**という前置引数をつけて呼び出したときは、現在の日付の時点で、すべての未完了の TODO エントリー（予約されているものも、そうでないものも）を一覧にします。

タイムラインのバッファで利用できるコマンドは、Section 10.5 [Agenda commands], page 101 にリスト化されています。

10.3.5 Search view

アジェンダのビューでは Org-mode のエントリーに対する一般的なテキスト検索機能を持っています。これはノートを探すのに特に役に立ちます。

C-c a s**org-search-view**

このコマンドは特別な検索のためのもので、論理式を使って、文字列または特定の単語に合致するエントリーを選択します。

例えば、`'computer equipment'`という検索文字列は、`'computer equipment'`という 1 つの文字列が含まれているエントリーを検索するでしょう。もしも、2 つの単語が、1 つ以上のスペースまたは改行で分かれていても、依然として一致するものを検索するでしょう。検索ビューでは、エントリーの中にある特別なキーワードについて論理式を使って検索することもできます。`'+computer +wifi -ethernet -{8\11[bg]}'`という検索文字列では、次のようなノートエントリーを検索します。`computer`と `wifi`というキーワードを含んでおり、`ethernet`というキーワードは含まれておらず、`8\11[bg]`という正規表現を含んでいない、すなわち 8.11b および 8.11g ともに含まれていないという意味ですが、エントリーを検索します。最初の `+` は単語検索を開始するために必要ですが、ほかの `+` はオプションです。詳しく知りたい場合は、`org-search-view`というコマンドのドキュメント文字列を参照してください。

アジェンダファイルに加えて、このコマンドは `org-agenda-text-search-extra-files` の中で一覧になっているファイルもまた検索するということに注意してください。

10.3.6 Stuck projects

もしもあなたが、以下に述べるような David Allen 氏の GTD のようなシステムであなたの仕事を管理しているならば、あなたが抱えている「義務」のひとつは、すべてのプロジェクトが進んでいるかを明確にするために、レビューを定期的に行うことです。詳細が未決定のプロジェクトは、次の行動が何も定義がされていないため、Org-mode が提示する TODO リストに、全く何も表示されることがないので。レビューをする際に、そういったプロジェクトを明確にし、それらのプロジェクトのための次の行動を定義することが必要です。

C-c a #**org-agenda-list-stuck-projects**

詳細が未決定のプロジェクトリスト

C-c a !

`org-stuck-projects` の変数をカスタマイズすることで何が詳細が未決定のプロジェクトで、どうやったらそういうプロジェクトを発見できるかを定義することができます。

あなたは九分九厘このコマンドが機能するために、このビューを定義する必要があります。あらかじめビルトインされているデフォルトの設定では、すべてのあなたのプロジェクトは第 2 階層の見出しに記述されており、あるプロジェクトが未決定であるとはいえない状況とは、すくなくとも 1 つのエントリーに TODO または NEXT または NEXTACTION という印がつけられている場合です。

Org-mode を使う際に、あなた自身の方法でアプローチするとして、PROJECT というタグがあるものをプロジェクトと定義し、プロジェクトがまだ検討する段階にないということを示すために

TODO キーワードで MAYBE と書いているものと仮定しましょう。さらに TODO キーワードで DONE という印の付いたものは完了したプロジェクトであると仮定しましょう。そしてまた NEXT もしくは TODO と書かれたものは NextAction であると仮定しましょう。@SHOP というタグがついたときは NEXT というタグが付いていなくても、ショッピングに行くという次の行動を示しているとします。最終的に、もしもプロジェクトに IGNORE (無視) という特別なキーワードがどこかについていたら、それはリストに表示されないものとします。このようなケースの場合、タグ・TODO⁷ が '+PROJECT/-MAYBE-DONE' とマッチし、さらにサブツリーに TODO、NEXT、@SHOP、および IGNORE というタグが付いているようなプロジェクトは、詳細が未決定のプロジェクトではないといえます。このようなカスタマイズを正しく定義するには、

```
(setq org-stuck-projects
  '("+PROJECT/-MAYBE-DONE" ("NEXT" "TODO") ("@SHOP")
    "\\<IGNORE\\>"))
```

もしもあるプロジェクトが詳細が未決定のプロジェクトではないと定義されたならば、そのエントリーのサブツリーは依然として詳細が未決定のプロジェクトとして検索されるということに注意してください。

10.4 Presentation and sorting

アジェンダビューにアイテムが表示される前に、Org-mode ではそのアイテムを表示し並び替える準備を行っています。それぞれのアイテムは 1 行を占めます。その行にはその項目の *category* (see Section 10.4.1 [Categories], page 99) を含んだ *prefix* とそれ以外の重要な情報を含んでいます。あなたは `org-agenda-tags-column` を使って表示されるコラムタグをカスタマイズすることができます。`org-agenda-prefix-format` のオプションを使用して前置引数をカスタマイズすることができます。この前置引数は、そのアイテムに関連するアウトラインの見出しの最新のバージョンに従います。

10.4.1 Categories

カテゴリーとは、それぞれのアジェンダアイテムに割り当てられた幅の広いラベルです。デフォルトでは、カテゴリーはファイルの名前から単純に作成されます。しかし、バッファ上で特別な行を足すことでそれを指定することができます。⁸

```
#+CATEGORY: Thesis
```

もしもあなたが、1 つのエントリーもしくは 1 つの (サブ) ツリーに特別な CATEGORY を持たせたいと望むのなら、そのエントリーに、値として適用したいと思っている特別なカテゴリーを `:CATEGORY:` という属性に設定しなさい。

アジェンダバッファの表示は、そのカテゴリーが 10 文字以上長くしない方が見栄えが良いです。

あなたは `org-agenda-category-icon-alist` 変数をカスタマイズすることで、カテゴリーにアイコンを設定することができます。

⁷ See Section 6.3 [Tag searches], page 56.

⁸ 逆に言うと、以下のような動作も生じます。もしも 1 つのファイルの中に、いくつかのそういう行が存在するならば、それよりも下の行にあるテキストに、そのカテゴリーをそれぞれ指定することになります。最初のカテゴリーは、その最初のカテゴリーの行はよりも前にあるどのテキストにも適用されます。しかしながら、*strongly* という手法を使うことは、文書のアウトライン構造と非互換であることを、強く非難することになります。複数のカテゴリーをバッファの中で設定する正しい方法は属性を使用することです。

10.4.2 Time-of-day specifications

Org-mode は時刻の仕様に基づいて、それぞれのアジェンダアイテムをチェックします。時刻は、例えば、‘<2005-05-10 Tue 19:00>’ のように、アジェンダの中に含まれているものをトリガーとしたタイムスタンプの一部です。時間の幅は 2 つのタイムスタンプで指定され、例えば ‘<2005-05-10 Tue 20:30>--<2005-05-10 Tue 22:15>’ のように記載されます。

そのエントリー自身の見出しの中で、時刻（時間）はプレーンなテキストとして（‘12:45’ や ‘8:30-1pm’）のように表示されます。もしもアジェンダが Emacs のダイアリー (see Section 10.3.1 [Weekly/daily agenda], page 92) と一体化されていたときは、ダイアリーのエントリーの中で指定した時間は、同様に認識されます。

アジェンダの表示のために、Org-mode は時間を引き出し、前置引数の一部として標準的な 24 時間のフォーマットでそれを表示します。前の段落に書かれた時間の例は、アジェンダの中で結局以下のように表示されます。

```
8:30-13:00 Arthur Dent lies in front of the bulldozer
12:45..... Ford Prefect arrives and takes Arthur to the pub
19:00..... The Vagon reads his poem
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

もしもアジェンダが一日モードであるならば、あるいは今日を表示しているならば、時間設定されたエントリーは、次のような時間のグリッドに埋め込まれます。

```
8:00..... -----
8:30-13:00 Arthur Dent lies in front of the bulldozer
10:00..... -----
12:00..... -----
12:45..... Ford Prefect arrives and takes Arthur to the pub
14:00..... -----
16:00..... -----
18:00..... -----
19:00..... The Vagon reads his poem
20:00..... -----
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

時間のグリッドは、org-agenda-use-time-grid 変数で表示したりしなかったさせることができます。そしてまた org-agenda-time-grid で設定をすることができます。

10.4.3 agenda の項目をソートする

ビューに書き出される前に、各アイテムは並び替えが行われます。この並び替えはビューのタイプによって決まります。

- 一日／一週間のアジェンダでは、それぞれの日の各アイテムは順番に並びます。デフォルトの順番は、明示的に日付と時刻の指定を含んでいるアイテムを、最初に集めます。これらのアイテムは、その日のスケジュールに応じて、リストの最初から順番に表示されます。その次に、各アイテムは org-agenda-files によって決められた順番に、カテゴリーごとにグループ分けされます。それぞれのカテゴリーの中で、各アイテムは優先順位 (see Section 5.4 [Priorities], page 50) に従って並び替えられます。優先順位は基本的な優先順位で構成されます（優先順位 ‘A’ ならば 2000、‘B’ ならば 1000、‘C’ ならば 0 として）。さらに、予定あるいはデッドラインを過ぎているアイテムのウェイトが追加されます。
- TODO リストでは、各アイテムはカテゴリーの順番に並び替えられますが、各カテゴリーの中では、優先順位 (see Section 5.4 [Priorities], page 50) によって並び替えられます。優先順位

は、優先順位の記号に従って並べ替えられます。さらに、アイテムが実行する日あるいは予約した日にどれだけ近いということも考慮されます。

- タグでの一致については、項目は並び替えは行われず、アジェンダファイルの中で一致した項目が発見された順番に従って表示されるのみです。

並び替えは、`org-agenda-sorting-strategy`変数でカスタマイズすることができます。そして、並び替えはそのエントリーの工数の見積りに基づく評価も含まれます。

10.5 Commands in the agenda buffer

アジェンダバッファでのエントリーは、その項目が作成された Org-mode ファイルと日記ファイルの間でリンクされます。アジェンダバッファでは編集することはできませんが、コマンドを使って、そのエントリーがある場所を表示したり、ジャンプして、アジェンダバッファから「遠隔的に」Org-mode ファイルを編集することができます。この方法で、すべての情報は1度書き込めばよく、あなたがアジェンダとノートのファイルが別の情報になるというリスクを避けることができます。

いくつかのコマンドはアジェンダの行上でマウスをクリックすることで実行されます。それ以外のコマンドは、必要とされる行の中にカーソルが置かれている必要があります。

Motion

`n` `org-agenda-next-line`
次の行へ (up及び `C-p`と同じ)。

`p` `org-agenda-previous-line`
次の行へ (down及び `C-n`と同じ)。

View/Go to Org file

`SPC` or `mouse-3` `org-agenda-show-and-scroll-up`
そのアイテムのオリジナルの場所を別のウィンドウで表示する。前置引数を使うことで、見出しだけでなく、アウトライン上にエントリー全体を明確に表示する。

`L` `org-agenda-recenter`
オリジナルの場所を表示し、ウィンドウのセンターに再配置する。

`TAB` or `mouse-2` `org-agenda-goto`
別のウィンドウでそのアイテムのオリジナルの場所に移動する。

`RET` `org-agenda-switch-to`
そのアイテムのオリジナルの場所に移動し、他のウィンドウは削除する。

`F` `org-agenda-follow-mode`
Follow モードをトグルする。Follow モードではアジェンダバッファ上でカーソルを動かすと、Org-mode ファイルの中で、別のウィンドウ上で対応する場所を表示する。新しいアジェンダバッファの中でこのモードの初期設定値は、`org-agenda-start-with-follow-mode`変数で設定することができる。

`C-c C-x b` `org-agenda-tree-to-indirect-buffer`
間接的なバッファの中で可憐とアイテムのサブツリー全体を表示する。数値付きの前置引数 `N` をつけると、第 `N` 階層まで階層を上がり、そのツリーを取得する。もしも `N` がマイナスならば、多くの階層まで上がる。`C-u`という前置引数を付けた場合は、既に使われた間接的バッファは消去されない。

C-c C-o**org-agenda-open-link**

エントリーの中にあるリンクをフォローする。この機能は、参照されている Org-mode のノードに属しているテキストの中に含まれているいくつかのリンクの中から選択するという機能を提供する。もしもリンクが 1 つしかない場合は、選択画面を表示せずに、そこにリンクを貼る。

Change display

- o 他のウィンドウを削除します。

v d or short **d**
v w or short **w**
v m
v y
v SPC

org-agenda-day-view
org-agenda-day-view
org-agenda-month-view
org-agenda-month-year
org-agenda-reset-view

日／週／月／年のビューを切り替えます。日または週にビューを切り替えたときは、この設定は、それに続くアジェンダの更新についてのデフォルトの設定となります。月および年のビューは、作成するために時間を要するので、デフォルトとはしていません。数字の付いた前置引数をつけると、その年、ISO の週、月、年の指定した日に直接ジャンプします。例えば **32 d**と書いたときは 2 月 1 日、**9 w**と書いたら ISO の週番号が 9 を指します。日、週あるいは月のビューを設定したときは、1 年は同様に前置引数の中でコード化されます。例えば、**200712 w**と書いたときは 2007 年の第 12 週にジャンプするでしょう。もしもそのような年の指定を、1 桁もしくは 2 桁の数字で行いたいたときは、1938 年から 2037 年の間に位置づけられます。**v SPC**によって、**org-agenda-span**での設定をリセットすることができます。

f**org-agenda-later**

時間を前の日付の表示へと遡ります。

.**org-agenda-goto-today**

今日へ移動します。

j**org-agenda-goto-date**

日付の選択画面でその日に移動します。

J**org-agenda-clock-goto**

アジェンダバッファの中で現在時間を計測中のタスクに移動します。

D**org-agenda-toggle-diary**

日記のエントリーに含めるかどうかトグルします。参照 Section 10.3.1 [Weekly/daily agenda], page 92.

v l or short **l****org-agenda-log-mode**

Logbook mode にするかどうかをトグルします。Logbook mode の中では、ログの取得中に (変数 **org-log-done**)DONE と印が付けられたエントリーが、その日の時刻を持っているエントリーとして、アジェンダの中に表示されます。**org-agenda-log-mode-items**変数を用いて log モードに含まれるエントリーのタイプを設定することができます。**C-u**という前置引数をつけて呼び出すと、状態の変化を含め、すべてのおこりうる logbook のエントリーを表示できるでしょう。**C-u C-u**という 2 つの前置引数をつけて呼び出すと、ログの情報のみが表示され、それ以外は表示されません。**v L**は、**C-u v l**と等価です。

- `v [` or short `[` `org-agenda-manipulate-query-add`
現在のビューに、不活性のタイムスタンプを含めます。週／日のアジェンダとタイムラインビューのみです。
- `v a` `org-agenda-archives-mode`
`v A` `org-agenda-archives-mode 'files`
Archives モードをトグルします。Archives モードでは、ARCHIVED と印されたツリーもまたアジェンダを作成するときにスキャンされます。大文字の `A` を使用したときは、全てのアーカイブファイルを含みます。archives mode から出るためには、再度 `v a` を押してください。
- `v R` or short `R` `org-agenda-clockreport-mode`
Clockreport モードをトグルします。Clockreport モードでは、日／週のアジェンダは、時間軸のための時刻のついた表を表示し、カレントのアジェンダビューでカバーされる範囲をファイルします。新しいアジェンダバッファの中で、このモードの初期設定は、`org-agenda-start-with-clockreport-mode` 変数で設定することができます。このモードをトグル (すなわち `C-u R`) している時に、前置引数を使用することで、アジェンダフィルター⁹ によって隠されているエントリーからの情報を表示しないでしょう。
- `v E` or short `E` `org-agenda-entry-text-mode`
entry text mode をトグルします。entry text mode では、アジェンダ行によって参照されている Org-mode のアウトラインのノードから、多数の行が、その行の下に表示されるでしょう。最大の行数は、`org-agenda-entry-text-maxlines` 変数で指定します。数値付きの前置引数を付けて、このコマンドを呼び出すと、前置引数の値の数によって、即座に修正されます。
- `G` `org-agenda-toggle-time-grid`
時間のグリッドの表示をトグルします。`org-agenda-use-time-grid` と `org-agenda-time-grid` 変数を参照してください。
- `r` `org-agenda-rod`
アジェンダバッファを再構築する。例えば、`S-left` と `S-right` を使って、アイテムのタイムスタンプを改修したあと、その変更を反映するために。そのバッファがグローバルな TODO リストの場合は、指定した TODO キーワードを選択できるリストを作成するために、前置引数を解釈します。
- `g` `org-agenda-rod`
カレントの Emacs のセッションにおいて、すべての Org-mode のバッファを保存します。あわせて ID の場所も。
- `C-c C-x C-c` `org-agenda-columns`
アジェンダバッファの中でカラムビュー (see Section 7.5 [Column view], page 61) を作成します。カラムビューのフォーマットは、その時点のエントリーから作成され、あるいは (もしも、その時点でエントリーが存在しないなら)、アジェンダビューの最初のエントリーから作成されます。そのエントリーのためのフォーマットが何であれ、(プロパティーから、`#+COLUMNS` という行から、あるいは `org-columns-default-format` 変数のデフォルトから作成された) オリジナルのバッファに存在しているエントリーのフォーマットがアジェンダで使用されます。

⁹ ここではタグフィルターだけが有効です。工数のフィルターは無視されます。

C-c C-x > **org-agenda-remove-restriction-lock**
 もしもファイルまたはサブツリーをその時点で制限しているならば、アジェンダをロックする制限を取り除きます。(see Section 10.1 [Agenda files], page 90).

Secondary filtering and query editing

/ **org-agenda-filter-by-tag**
 タグおよび（または）工数の見積りに対して、カレントのアジェンダビューにフィルターをかけます。これとカスタムなアジェンダコマンドとの間の差異は、このフィルターが非常に早いということです。このため、あなたは、アジェンダ（注1）を再表示することなく、異なるフィルターの間を素早く切り替えることができます。¹⁰

タグ選択の文字を入力しましょう。SPCはタグの全てを意味しています。入力部分でTABを押すと、選択するタグの補完機能を使用できます（すべてのタグに選択用の文字が指定されているとはかぎりません）。そして、そのコマンドは、このタグを含んでいないか継承していないエントリーを全て隠します。前置引数をつけて呼び出した場合は、そのタグを持っているエントリーを削除さえしてしまいます。入力部で2番目の/はフィルターを終了し、隠されているエントリーを再度出現させます。もしも最初に入力したキーが、+または-ならば、前のフィルターは、選択された新たなタグの要求あるいは禁止に応じて、幅を狭くします。/の後に、+あるいは-を入力する代わりに、\ コマンドを即座に使用することもできます。

工数見積のフィルターをかけるために、予め認められている汎用的な工数を設定すべきです。例えば

```
(setq org-global-properties
  '(("Effort_ALL". "0 0:10 0:30 1:00 2:00 3:00 4:00")))
```

あなたは、<、>および=のひとつの操作を最初に入力することで、工数のためのフィルターをかけることができます。それから、あらかじめ認められた値のリストの中で、工数見積りのインデックスの数字を入力します。そこでは0は10番目の値を意味します。フィルターは選択された値よりも、以下、イコール、以上であるかによって限定されます。もしも0-9のキーがタグへのアクセスキーとして使用されていないならば、単純にあなたは操作コマンドを利用することなく、直接インデックスとなる数字を入力するだけです。この場合<が假定されます。操作のアプリケーションのために、定義された工数がないエントリーでは、**org-sort-agenda-noeffort-is-high**変数の値に従って取り扱われます。工数の定義のないタスクにフィルターをかけるには、?を操作の値として入力します。

Org-mode はまた、コンテキストに対応したタグのフィルターを自動的にサポートしています。もしも、**org-agenda-auto-exclude-function**変数の値が、ユーザが定義した機能に設定されているときは、その機能によって、どのようなタグがアジェンダから自動的に排除されるかを決定します。一度この機能が設定されると、それによって、/ コマンドは、**RET** をサブのオプションキーとして受け付け、自動的に排除ロジックを走らせます。例えば、いってみれば、ネットワークへのアクセスを必要とするタスクを定義するために **Net** というタグ、街での用事のために **Errand** というタグ、電話を掛けなければならないときに **Call** というタグを使用しているとします。あなたは、インターネッ

¹⁰ カスタムコマンドによって、オプションとして **org-agenda-filter-preset** 変数と結びつけることで、フィルターを事前にセットすることができます。このフィルターは、ビューに適用されます。そして、リフレッシュや2番目のフィルターを通して、基本的なフィルターとして存続します。このフィルターは、アジェンダのブロックの中で、アジェンダビュー全体のグローバルなプロパティです。この設定を行うためには、個別のブロックのセクションではなく、グローバルオプションのセクションで行います。

トを利用できるかどうか、仕事時間外にあるかどうか、このような状況に基づいて、これらのタグを自動的に排除することができるのです。

```
(defun org-my-auto-exclude-function (tag)
  (and (cond
        ((string= tag "Net")
         (/= 0 (call-process "/sbin/ping" nil nil nil
                              "-c1" "-q" "-t1" "mail.gnu.org"))))
        ((or (string= tag "Errand") (string= tag "Call"))
         (let ((hour (nth 2 (decode-time))))
           (or (< hour 8) (> hour 21))))))
    (concat "-" tag)))

(setq org-agenda-auto-exclude-function 'org-my-auto-exclude-function)
```

`\` `org-agenda-filter-by-tag-refine`
追加の条件によってカレントのアジェンダフィルターをナローイングします。前置引数を用いてコマンドを呼び出したときは、まさにタグがついているエントリー、あるいは工数の基準にまさに合致するエントリーを削除する。/コマンドのあとの最初のキーとして、+あるいは-を押すことで同様の効果を達成することができます。

`[] { }`

in *search view*

新しい検索の単語 (`[と]`)、あるいは新しい正規表現 (`{と}`) をクエリー文字列に追加する。開いた角括弧／大括弧は、`+` という接頭辞のついたポジティブな検索用語を追加する。この検索用語は、必ずそのエントリーに発生／合致しなければならないことを示す。閉じた角括弧／大括弧は、ネガティブな検索用語を追加し、それは、選択されているエントリーの中で、絶対に発生／合致しないということである。

Remote editing

`0-9` Digit argument.

`C-_` `org-agenda-undo`
外部の編集コマンドでの変更を元に戻す。この変更はアジェンダバッファと外部のバッファの両方を元に戻す。

`t` `org-agenda-todo`
アイテムの TODO のステータスを変更する。アジェンダファイルでもオリジナルの Org ファイルでも有効である。

`C-S-right` `org-agenda-todo-nextset`
`C-S-left` `org-agenda-todo-previousset`
次／前の TODO キーワードのセットへと切り替える。

`C-k` `org-agenda-kill`
オリジナルの Org ファイルの中で、そのアイテムが属しているサブツリー全体と共に、カレントのアジェンダアイテムを削除する。もしも外部ファイルの削除するテキストが1行以上ならば、削除を行うには、ユーザーが指定する必要がある。`org-agenda-confirm-kill`変数を参照のこと。

- C-c C-w** **org-agenda-refile**
その時点でそのエントリーを差し替える。
- C-c C-x C-a** or short **a** **org-agenda-archive-default-with-confirmation**
org-archive-default-commandに設定されたデフォルトのアーカイブコマンドを使用して、その時点でエントリーに対応したサブツリーをアーカイブする。**a**キーを使用したときは、承認が必要である。
- C-c C-x a** **org-agenda-toggle-archive-tag**
カレントの見出しのための **ARCHIVE** タグをトグルする。
- C-c C-x A** **org-agenda-archive-to-archive-sibling**
カレントエントリーに対応したサブツリーを、アーカイブファイルに移動する。
- C-c C-x C-s** or short **\$** **org-agenda-archive**
カレントの見出しに対応したサブツリーをアーカイブする。これは、設定されたアーカイブの場所に、多くの場合それは異なるファイルであるが、エントリーを移動することを意味している。
- T** **org-agenda-show-tags**
カレントアイテムと関連づけられたすべてのタグを表示する。もしも、あなたが **org-agenda-show-inherited-tags** 機能を停止しているにもかかわらず、依然として、たびたび見出しのすべてのタグを確認したいというときに役に立つ。
- :** **org-agenda-set-tags**
カレントの見出しにタグを設定する。もしもアジェンダの中にアクティブなリージョンがあるときは、そのリージョンの中ですべての見出し用としてタグを変更する。
- ,** **org-agenda-priority**
カレントアイテムに優先順位を設定する。**(org-agenda-priority)** Org-mode は優先順位を表す文字を指示します。もしも、SPCを使って返答すると、優先順位のクッキーがそのエントリーから取り除かれる。
- P** **org-agenda-show-priority**
カレントアイテムの優先順位の重み付けを表示する。
- + or S-up** **org-agenda-priority-up**
カレントアイテムの優先順位を高くする。優先順位はオリジナルのバッファで変更される。しかしアジェンダ上では並び替えの更新は行われたい。このためには、**r** キーを使用する。
- or S-down** **org-agenda-priority-down**
カレントアイテムの優先順位を低くする。
- z or C-c C-z** **org-agenda-add-note**
そのエントリーにのノートを追加する。このノートは記録され、ノートが置かれている状態を変更した同じ場所にファイルされる。**org-log-into-drawer**によって、これは引き出しの中に入る。
- C-c C-a** **org-attach**
すべてのコマンドの選択画面は、付属するものに関連づけられる。
- C-c C-s** **org-agenda-schedule**
このアイテムを予約する。前置引数をつけると、予約のタイムスタンプが削除される。

C-c C-d org-agenda-deadline
 このアイテムにデッドラインを設定する。前置引数をつけるとデッドラインが削除される。

k org-agenda-action
 カーソルの置かれた日付に選択されたアイテムの日付を設定するための、アジェンダのアクション。このコマンドはカレンダーでも動作する！コマンドは追加されたキーで入力する。

m その地点でアクションのためにエントリーにマークする。複数のエントリーに対しても可能である。

Org-mode では次を伴う **C-c C-x C-k**。

d その時点の日付でマークされたエントリーのデッドラインを設定する。

s その時点の日付でマークされたエントリーを予約する。

r デフォルトの日付としてカーソルの日付とともに **org-capture** を呼び出す。

アジェンダを更新した後に、**r** を押すと、コマンドの効果を確認できる。

S-right org-agenda-do-date-later
 カレント行に関連づけられたタイムスタンプを 1 日先に変更する。数値付きの前置引数をつけると、その数字の日数分だけ先に変更する。例えば、**3 6 5 S-right** と入力すると 1 年先に変更される。**C-u** という前置引数をつけると、1 時間ずつ時間を変更する。もしもあなたが、同じコマンドを即座に繰り返したいときは、前置変数を付けなくても 1 時間単位で変化し続けるでしょう。二重の **C-u C-u** という前置引数をつけると、同様に分単位で変更される。オリジナルの Org-mode ファイルの中でタイムスタンプは変更されるが、その変更はアジェンダバッファには直接は反映されない。バッファを更新するには、**r** または **g** を使用する。

S-left org-agenda-do-date-earlier
 カレント行のに関連づけられたタイムスタンプを 1 日過去に変更する。

> org-agenda-date-prompt
 カレント行に関連づけられたタイムスタンプを変更する。**>** キーが選択される。というのは、私のキーボード上では **S-.** と同じだからである。

I org-agenda-clock-in
 カレントアイテムの時計をスタートする。もしもすでに時計が動いているのならば、まずそれが停止する。

O org-agenda-clock-out
 すでにスタートした時計を停止する。

X org-agenda-clock-cancel
 カレントで動いている時計をキャンセルする。

J org-agenda-clock-goto
 別のウインドウの中の動いている時計にジャンプする。

Bulk remote editing selected entries

m org-agenda-bulk-mark
 大量のアクションについて、その時点でエントリーにマークをつける。前置引数をつけると、多くの連続したエントリーにマークをつける。

U **org-agenda-bulk-remove-all-marks**
 大量のアクションのマークを取り除く。

U **org-agenda-bulk-remove-all-marks**
 大量のアクションのためにマークがつけられたエントリーのマークを取り除く。

B **org-agenda-bulk-action**
 大量のアクション。アジェンダの中ですべてのマークをつかれたエントリーについて実行する。この機能では、適用されるアクションを選択するために、別のキーを入力する。**B**に前置引数をつけると、**s**や**d**のコマンドをパスして、これらの特別なタイムスタンプをまとめて取り除く。

r 1つのリフィル上のターゲットに入力しすべてのエントリーを移動する。そのエントリーは
 アジェンダ上には表示されなくなる。再表示(**g**)によって再度表示される。
\$ 選択されているエントリーをすべてアーカイブする。
A エントリーをアーカイブし、それぞれを所定のアーカイブ先に移動する。
t TODOの状態を変更する。これはTODOキーワード1文字を入力し、そして
 選択されたエントリーすべての状態を変更する。それはブロックしているのを無視し
 ログのノートを抑え込んで(タイムスタンプは別です)。
+ 選択されたエントリーのすべてにタグを付加する。
- 選択されたエントリーのすべてから、タグのひとつを削除する。
s すべてのアイテムに新しい日付で予約する。すでに予約がついていれば、日数分だけ
 日付を更新する。入力欄でプラスを2つつけて何かの数字を最初に打つことで。
 例えば、**'++8d'**とか**'++2w'**のように。
S N日を指定して、それぞれをリスケジュールする。Nは入力欄で指定する。
 前置引数
 (**C-u B S**)をつけることで、平日のみに指定できる。
d 指定した日をデッドラインとして設定する。

Calendar commands

c **org-agenda-goto-calendar**
 Emacsのカレンダーを開き、アジェンダのカーソルの置かれている日付に移動します。

c **org-calendar-goto-agenda**
 すでにカレンダーの中にあるときは、カーソルの置かれている日付で計算し、Org-modeのアジェンダを表示します。

i **org-agenda-diary-entry**
 カーソルの置かれている日付および（ブロックエントリーでは）マークされた日付を使って、新しいエントリーを日記に書き込みます。この機能では Emacs の日記ファイル¹¹ に追加することになります。ある意味では、カレンダーの **i** コマンドと似た機能です。日記ファイルは別のウインドウにポップアップし、そこでエントリーを書き加えることができます。

もしも Org-mode ファイルに **org-agenda-diary-file** を指定したならば、Org-mode ではそのファイルの中に（Org-mode の構文を使って）日記の代わりに、エントリーを作成することができます。ほとんどのエントリーは、日付を元にしたアウトラインのツリーの中に記述されており、あとで過去の月／年の中から予定をアーカイブするのを簡単にします。そのツリーは、**DATE_TREE** 属性か、最上位のエントリーとして、年という属性を持ったエントリーのもとに構築されています。Emacs でエントリーのテキストを入力するようプロンプトが表示されるでしょう。もしもあなたがそれを指示するならば、さらなる連携なく、**org-agenda-diary-file** にそのエントリーを作成することになるでしょう。テキストを入力することなく、その入力欄で直接 **RET** を入力したら、そのターゲットとなるファイルがその場でエントリーを終了させ、別のウインドウが表示されるでしょう。**kr** コマンドを参照してください。

M **org-agenda-phases-of-moon**
 その日を中心として 3ヶ月間の月齢を表示する。

S **org-agenda-sunrise-sunset**
 日の出と日の入りを表示する。地理上の場所によって、カレンダーの変数が設定される。Emacs の **calendar** の章を参照のこと。

C **org-agenda-convert-date**
 カーソルの置かれている日付によって、多くの他の文化的・歴史的なカレンダーに変換する。

H **org-agenda-holidays**
 カーソルのある日付を中心に 3ヶ月間の祝祭日を表示する。

M-x org-export-icalendar-combine-agenda-files

すべてのアジェンダファイルからエントリーを含んだ iCalendar 形式のファイルにエクスポートする。これはグローバルに利用できるコマンドで、そしてまたアジェンダメニューの中で利用できるコマンドです。

Exporting to a file

C-x C-w **org-write-agenda**
 アジェンダビューを 1つのファイルに書き出します。選択したファイル名の拡張子に従って、そのビューは、HTML（拡張子が **‘.html’** または **‘.htm’**）、Postscript（拡張子 **‘.ps’**）、PDF（拡張子 **‘.pdf’**）、そしてプレーンテキスト（その他の拡張子）などにエクスポートされます。**C-u** という前置引数を用いてコマンドを呼び出したならば、即座に新しく作成されたファイルが開きます。エクスポートの間に使用されている **‘ps-print’** および **‘htmlize’** のためのオプションを設定するために、**org-agenda-exporter-settings** 変数を使用します。

Quit and Exit

¹¹ **org-agenda-include-diary** が設定されているときは、このファイルはアジェンダ用に解析されます。

- q** **org-agenda-quit**
アジェンダを終了し、アジェンダバッファを削除します **s**。
- x** **org-agenda-exit**
アジェンダを終了し、アジェンダバッファとアジェンダを編集するために Emacs で読み込まれたすべてのバッファを削除する。Org-mode ファイルを読み込むためにユーザーによって作成されたバッファは削除されない。

10.6 Custom agenda views

カスタムアジェンダコマンドは2つの目的を提供する。ひとつは TODO とタグの検索を使用して、保存と素早く頻繁にアクセスするため。もうひとつは、特別に合成したアジェンダバッファを作成するため。カスタムなアジェンダコマンドはデフォルトのコマンドと同様に、コマンド選択画面ディスパッチャー (see Section 10.2 [Agenda dispatcher], page 91) を通して利用できる。

10.6.1 Storing searches

カスタム検索の最初のアプリケーションは、よく使われる検索式のためのキーボードショートカットを定義することです。それはアジェンダバッファの作成、またはツリーの抽出（後者は言うまでもなくカレントバッファのみをカバーする）のどちらに対してでも。カスタムコマンドは、**org-agenda-custom-commands**変数で設定されます。あなたはこの変数をカスタマイズできます。例えば、**C-c a C**というように。またあなたは`‘.emacs’`に Emacs の Lisp を記述して直接設定することもできます。以下に述べる例はすべての適正な検索タイプを含んでいます。

```
(setq org-agenda-custom-commands
  '(("w" todo "WAITING")
    ("W" todo-tree "WAITING")
    ("u" tags "+boss-urgent")
    ("v" tags-todo "+boss-urgent")
    ("U" tags-tree "+boss-urgent")
    ("f" occur-tree "\\<FIXME\\>")
    ("h" . "HOME+Name tags searches") ; description for "h" prefix
    ("hl" tags "+home+Lisa")
    ("hp" tags "+home+Peter")
    ("hk" tags "+home+Kim")))
```

それぞれのエントリーの頭文字は、コマンドにアクセスするために、コマンド選択画面を呼び出す **C-c a**というコマンドの後に、入力しなければならないキーを定義します。通常、これは1文字をあてますが、もしもあなたが似たようなコマンドをたくさん持っていたら、あなたは2文字の組合せで定義することができます。その場合、いくつかの組合せでは最初の文字が同じものとなり、前置引数¹²と同じように提供されます。2番目のパラメーターは検索の種類を示し、マッチさせるために使われる文字列や正規表現がそれに続きます。上の例ではそれゆえ以下のように定義します。

- C-c a w** TODO のキーワードとして、`‘WAITING’`となっている TODO エントリーのためのグローバルな検索として。す。
- C-c a W** 同じような検索であるが、カレントバッファにのみ適用され、ツリーの抽出として検索結果を表示する。

¹² あなたは前置引数と説明をつけて、コンソールのセルを挿入することで、前置引数のキーのための説明を表示することができます。

- C-c a u** ‘:urgent:’ではなく‘:boss:’というタグがつけられた見出しのための、グローバルなタグ検索を行う。
- C-c a v** **C-c a u**と同じ検索を行うが、TODO アイテムである見出しに対してのみ検索を行うという制限がある。
- C-c a U** **C-c a u**と同じ検索を行うが、カレントバッファに対してのみ検索を行い、結果をツリーの抽出として表示する。
- C-c a f** すべてのエントリーのうちで‘FIXME’という言葉を含んでいるものを検索してツリーの抽出を行う（くどいかもしれませんが、カレントバッファだけが対象です）。
- C-c a h** HOME というタグ検索のためのコマンドの前置引数として、そこでは、タグ検索の追加として、一つの名前 (Lisa、Peter、または Kim) を選択するために、あなたはさらに (l、p、または k) というキーを追加入力する必要があります。

10.6.2 Block agenda

もう一つの可能性とは、アジェンダビューの構築です。そのビューは、様々なコマンドの結果で構成されており、それぞれのコマンドはアジェンダバッファの中の1つのブロックを作成します。利用できるコマンドは (**C-c a a**を実行して作成された) 一日または週間アジェンダのための **agenda**、(**C-c a t**を実行して作成された) グローバルな todo リストのための **alltodo**、そして上で議論してきた **todo**、**tags**、**tags-todo**などの検索コマンドに含まれています。2つの例を挙げます。

```
(setq org-agenda-custom-commands
  '(("h" "Agenda and Home-related tasks"
    ((agenda "")
     (tags-todo "home")
     (tags "garden"))))
  ("o" "Agenda and Office-related tasks"
    ((agenda "")
     (tags-todo "work")
     (tags "office")))))
```

これによって、家で精を出さなければならない用事に対するマルチブロックのビューを作成するために、**C-c a h**を定義します。アジェンダバッファには結果として、その週の、‘home’というタグが含まれているすべての TODO アイテムと、‘garden’というタグがついたすべての行のためのアジェンダを含むことになります。最後に、**C-c a o**というコマンドで、同様に、オフィスの作業についてのビューを得ることができます。

10.6.3 Setting options for custom commands

Org-mode はたくさんのアジェンダの構築や表示について調整する変数を含んでいます。グローバルな変数では、カスタムコマンドも含めて、アジェンダの全てのコマンドの動作を定義することができます。しかしながら、もしもあるひとつのカスタムビューについて、いくつかの設定を変更したいならば、それも可能です。オプションの設定は変数名のリストに書き込むことが必要で、**org-agenda-custom-commands**の中に、正しい位置に値を書き込む必要があります。例えば。


```
(setq org-agenda-custom-commands
  '(("w" todo "WAITING"
    ((org-agenda-sorting-strategy '(priority-down))
     (org-agenda-prefix-format " Mixed: ")))
    ("U" tags-tree "+boss-urgent"
     ((org-show-following-heading nil)
      (org-show-hierarchy-above nil)))
    ("N" search ""
     ((org-agenda-files '("~/org/notes.org"))
      (org-agenda-text-search-extra-files nil)))))
```

こう書き込むことによって、`C-c a w`というコマンドは、優先順位によってのみ収集したエントリーを並べ替えるでしょう。そのエントリーのカテゴリを設定する代わりに、例えば‘Mixed:’という文字を prefix の形で書くことで変更することができます。`C-c a U`というタグでツリーを抽出するコマンドは、この結果、超コンパクトとなるでしょう。なぜならば、検索に合致した項目の上の階層の見出しも、合致した項目の見出しもどちらも表示されないからです。`C-c a N`というコマンドは、1つのファイルに制限されたテキスト検索を実行します。

ブロックアジェンダを作成するコマンドセットのために、`org-agenda-custom-commands`ではオプションの設定用に2つの別の場所を用意しています。その設定の中にたったひとつのコマンドに有効なオプションを付け加えることも、その設定の中にすべてのコマンドに有効なオプションを付け加えることもできます。前者のオプションは1つのコマンドエントリーを付け加える。後者のオプションは、コマンドエントリーのリストを書き込むことが必要です。ブロックアジェンダの例に戻ると (see Section 10.6.2 [Block agenda], page 111)、`C-c a h`というコマンドで、並べ替えの順序を優先順位の降順 `priority-down`に変更することができますし、その中で「GARDEN」というタグのついたものについては反対の順序、すなわち優先順位の昇順 `priority-up`に並べ替えることができます。このことは以下のように記述できます。

```
(setq org-agenda-custom-commands
  '(("h" "Agenda and Home-related tasks"
    ((agenda)
     (tags-todo "home")
     (tags "garden")
     ((org-agenda-sorting-strategy '(priority-up)))))
    ((org-agenda-sorting-strategy '(priority-down))))
    ("o" "Agenda and Office-related tasks"
    ((agenda)
     (tags-todo "work")
     (tags "office")))))
```

おわかりだと思いますが、変数とカッコで囲んでいる設定はやや複雑なところがあります。わかりにくいときは、カスタマイズのインターフェースとしてこの変数を設定してください。これはカスタマイズの構造を完全にサポートしています。注意しなければならないのは、このインターフェースでオプションを設定するときに、変数は、Lisp による表現をとっているということです。そのため、もしもその変数が1つの文字ならば、あなた自身でその変数の値に「” (ダブルクォート)」で囲む必要があるということです。

10.7 Exporting Agenda Views

もしもあなたが自分のコンピュータから離れているときは、いくつかのアジェンダのバージョンを印刷して持ち歩くことは大変役に立ちます。Org-mode はカスタムアジェンダビューをプレーンなテキスト、HTML¹³、Postscript、PDF¹⁴、iCalendar ファイルとしてエクスポートすることができます。もしも、ときどきこのようなことを実行するのならばコマンドを使用しましょう。

C-x C-w

org-write-agenda

アジェンダビューを1つのファイルに書き出します。選択したファイル名の拡張子により、そのビューは HTML (拡張子が `‘.html’` または `‘.htm’`)、Postscript (拡張子が `‘.ps’`)、iCalendar (拡張子が `‘.ics’`)、あるいはプレーンなテキスト (何かほかの拡張子) としてエクスポートされます。エクスポートの間に、`‘ps-print’` のため、および `‘htmlize’` のためにオプションを設定するには、`org-agenda-exporter-settings` 変数を使用します。例えば

```
(setq org-agenda-exporter-settings
      '((ps-number-of-columns 2)
        (ps-landscape-mode t)
        (org-agenda-add-entry-text-maxlines 5)
        (htmlize-output-type 'css)))
```

もしも、あなたがアジェンダビューをたびたびエクスポートする必要があるのならば、アウトプットのファイルの名前¹⁵ のリストに、いくつかのカスタムなアジェンダのコマンドを関連づけることができます。ここに一つの例があります。最初のはアジェンダとグローバルな TODO リストに対するカスタムなコマンドを定義しており、それらをエクスポートするたくさんのファイルと一緒にしています。それから2つのブロックアジェンダコマンドを定義し、同様にそれらのためのファイル名を指定しています。ファイル名は、現在作業しているディレクトリに対して相対パスにすることも絶対パスにすることもできます。

```
(setq org-agenda-custom-commands
      '(("X" agenda "" nil ("agenda.html" "agenda.ps"))
        ("Y" alltodo "" nil ("todo.html" "todo.txt" "todo.ps"))
        ("h" "Agenda and Home-related tasks"
          ((agenda "")
            (tags-todo "home")
            (tags "garden")))
          nil
          ("~/views/home.html"))
        ("o" "Agenda and Office-related tasks"
          ((agenda)
            (tags-todo "work")
            (tags "office")))
          nil
          ("~/views/office.ps" "~/calendars/office.ics"))))
```

¹³ あなたは Hrvoje Niksic 氏の `‘htmlize.el’` をインストールする必要があります。

¹⁴ PDF の出力を作成するためには、Ghostscript の `‘ps2pdf’` ユーティリティがシステムにインストールされている必要があります。pdf ファイルを選択するとポストスクリプトファイルも作成されます。

¹⁵ もしもあなたが週間アジェンダやグローバルな TODO リストなどのような標準的なビューを保存したいならば、ファイル名を指定することができるようにするために、それらのビューのためにカスタムなコマンドを定義する必要があります。

ファイル名の拡張子がエクスポートのタイプを決定します。もしも拡張子が‘.html’ならば、Org-mode は‘htmlize.el’パッケージを使用し、バッファを HTML に変換し、そのファイル名で保存します。もしも拡張子が‘.ps’ならば、ps-print-buffer-with-faces が Postscript の出力のために使用されます。もしも拡張子が‘.ics’ならば、iCalendar のエクスポートは、アジェンダを構成しているすべてのファイルにわたってエクスポートを実行し、現在アジェンダの中ではリスト化されたエントリーのエクスポートに限定されます。ほかの拡張子がついた場合は、プレーンな ASCII テキストファイルが作成されます。

エクスポートファイルは、非常に負荷が高いので、これらのコマンドの一つを相互に影響するように使用している時は、出力されません。そのかわり、1 ステップですべての指定されたファイルを出力する特別なコマンドが用意されています。

C-c a e **org-store-agenda-views**
アジェンダに関連するエクスポートファイル名を持つすべてのアジェンダビューをエクスポートします。

あなたは、エクスポートコマンドのためのオプションの設定をするために。カスタムアジェンダコマンドのオプションのセクションを使用することができます。例えば、

```
(setq org-agenda-custom-commands
      '(("X" agenda ""
         ((ps-number-of-columns 2)
          (ps-landscape-mode t)
          (org-agenda-prefix-format " [ ] ")
          (org-agenda-with-colors nil)
          (org-agenda-remove-tags t))
         ("theagenda.ps"))))
```

このコマンドは、Postscript のエクスポートのために、2つのオプションを設定します。横長のフォーマットで2段のプリントを作成するためです。出力されたページは、2つにカットして、紙のアジェンダとして使えるようになります。もうひとつの設定は、行頭のカテゴリーとスケジューリング情報を省き、その代わりにチェックのついてないチェックボックスの項目となるようにアジェンダを修正します。私たちは各行をコンパクトに表示するためにタグを省略したり、白黒プリンタのためにカラーを使わない用にすることもできます。org-agenda-exporter-settingsの中で指定する設定もできますが、org-agenda-custom-commandsでの設定が優先します。

コマンドラインで次のような設定を使用することができます。

```
emacs -f org-batch-store-agenda-views -kill
```

また、いくつかのパラメーター¹⁶を修正する必要があります。

```
emacs -eval '(org-batch-store-agenda-views \
              org-agenda-span month \
              org-agenda-start-day "2007-11-01" \
              org-agenda-include-diary nil \
              org-agenda-files (quote ("~/org/project.org")))' \
      -kill
```

どちらも‘~/org/project.org’のファイルを対象として、日記のエントリーは除かれ、30日以内に限定したアジェンダビューを作成します。

¹⁶ 引用の方法はあなたの使用しているシステムに依存します。事例用の FAQ を確認してください。

あなたは、他のプログラムで将来の進行過程を認める方法で、アジェンダの情報を絞り込むことができます。詳細は Section A.8 [Extracting agenda information], page 195, のノートの情報を参照してください。

10.8 Using column view in the agenda

カラムビュー (see Section 7.5 [Column view], page 61) は、Org-mode ファイルの階層構造の中に組み込まれている属性を見たり編集したりするために通常は使われます。エントリーがある評価基準で収集されているアジェンダから、カラムビューを使用することは大変便利です。

`C-c C-x C-c`

`org-agenda-columns`

アジェンダの中でカラムビューに切り替えます。

この属性がどのようなものか理解するために、アジェンダのエントリーはもはや適切なアウトラインの環境ではなくなることを理解することが重要です。これによって以下のようなことが生じます。

1. Org-mode では、どの **COLUMNS** のフォーマットを使用するか、決定する必要があります。アジェンダの中のエントリーは、異なるファイルから集められるということと、ファイルが異なると **COLUMNS** のフォーマットも異なるということから、このことは些細な問題であるとはいえないのです。Org-mode は最初に、`org-overriding-columns-format` 変数がカレントで設定されているかどうか、またそこからフォーマットを取り出すことができるかどうかチェックします。一方、アジェンダの最初のアイテムに関連したフォーマットを使用するか、もしもそのアイテムが特別なフォーマット (属性もしくはファイルの中で定義された) を持たないならば、`org-columns-default-format` を使用します。
2. もしも、どれかカラムに要約形式 (see Section 7.5.1.2 [Column attributes], page 61) が定義されているならば、アジェンダでカラムビューに切り替えるときに、すべての関連するアジェンダ ファイルを確認して、この属性の計算の更新を確実に行います。このことは、特別な **CLOCKSUM** の属性が真であると設定されているということです。Org-mode はアジェンダの中で表示された値を合計するでしょう。一日/週間アジェンダの中で、合計は 1 日をカバーしています。他のビューでは、ブロック全体をカバーするのです。アジェンダでは同じエントリーを 2 度表示したり (例えばスケジュールと期限というように)、同じ階層 (例えば親と子) から 2 つのエントリーを表示したりするかもしれない、ということを理解することは重要なことです。これらの場合、アジェンダの中での要約は、いくつかの値が二重にカウントされるという間違った結果を導く可能性があります。
3. アジェンダの中のカラムビューが、**CLOCKSUM** を表示するときは、このアイテムのためにいつでも時間計測全体に対応します。そのため 1 日/週間アジェンダにおいて、カラムビューでリスト化された時間合計は、カレントのビューの外側の時間から発生することになるかもしれません。この機能によって、あるタスクについて、計画された総工数を 1 つのカラムにリストにして、その値を比較することができるので、優位性を持ちます。この機能はアジェンダのカラムビューにおける重要なアプリケーションのひとつです。もしもあなたが表示されている期間の中の作業時間についての情報を得たいならば、clock table mode (`R` をアジェンダの中で入力する) を使用してください。

11 Markup for rich export

Org-mode の文書をエクスポートする時、エクスポート機能は文書の構造をできるだけ正確に反映しようとしています。HTML や \LaTeX 、DocBook、その他のリッチフォーマット等のエクスポートの対象について、Org-mode は文書をリッチエクスポートに変換するルールを持ちます。このセクションは Org-mode のバッファで使われるマークアップのルールについて説明します。

11.1 Structural markup elements

Document title

エクスポートされた文書のタイトルは専用の行で設定されます。

#+TITLE: これは文書のタイトルです。

もしこの行が存在しなければ、タイトルはバッファ中の最初の空でない、コメントでない行を用います。もしまだ何も存在していない、またはあなたが最初の見出しより前のテキストをエクスポートをしないよう設定していたら、タイトルは拡張子無しのファイル名となります。

もしあなたがリージョンでマークしたサブツリーのみをエクスポートしているなら、サブツリーの見出しは文書のタイトルとなるでしょう。もしサブツリーが `EXPORT_TITLE` プロパティを持っているなら、そのプロパティの値が優先して用いられるでしょう。

Headings and sections

Chapter 2 [Document Structure], page 7 で説明されているような文書のアウトライン構造はエクスポートされた文書のセクションの定義の基準を形成しています。しかしながら、アウトライン構造はまた (例えば) タスクのリストとしても使われているので、最初の 3 アウトラインレベルのみ見出しとして使われます。

#+OPTIONS: H:4

Table of contents

目次は通常ファイルの最初の見出しの前に直接挿入されます。もしあなたが異なる場所に目次を挿入したいのなら、その場所に `[TABLE-OF-CONTENTS]` 文字列を書いてください。目次の深さはデフォルトでは見出しのレベルの数と同じですが、`org-export-with-toc` 変数を設定するか、ファイルに以下のように書くことによって、あなたはこれより小さな値に変更することも、目次を完全に表示させないようにすることも可能です。

#+OPTIONS: toc:2 (目次に表示するレベルを 2 までとする)

#+OPTIONS: toc:nil (目次を表示しない)

最初の見出しより前のテキスト

Org-mode は通常最初の見出しの前にテキストをエクスポートし、最初の行を文書のタイトルにします。テキストは完全にマークアップされているでしょう。もしあなたが HTML や \LaTeX 、DocBook のようなクリテラルを含めたい場合、独立したエクスポート機構のセクションで説明されている特別な構造を使います。

多くの人々は内部リンクの設定のためとそのために異なる方法でエクスポートされた最初の見出しの前のテキストを制御する最初の見出しの前に空白を使うことを好みます。あなたは `orgexport-skip-text-before-1st-heading` 変数を `t` にすることで設定することができます。ファイル中に設定する場合、あなたは `'#+OPTIONS: skip:t'` とすることで同等の設定を行うことができます。

もし、あなたがまだ最初の見出しの前にテキストを置きたいのであれば、**#+TEXT** 構造を使います:

```
#+OPTIONS: skip:t
#+TEXT: このテキストは*最初の*見出しの前に置かれます
#+TEXT: [TABLE-OF-CONTENTS]
#+TEXT: このテキストは目次と最初の見出しの間に置かれます
```

Lists

Section 2.7 [Plain lists], page 13 で説明されているプレーンリストは、バックエンドのリストに変換されます。多くのバックエンドがサポートしているのは記号付きリスト、番号付きリスト、見出し付きリストです。

段落、改行、引用

段落は最低 1 つの空白行で区切られます。もしあなたが強制的に段落の中で改行しないなら、‘\\\\’ を行の末尾に書いてください。

リージョンで改行を保つためには、しかしそうでなければ通常のフォーマットが使われるなら、あなたはフォーマット技法として使われるこの構文を使うことができます。

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

-- AlexSchroeder
#+END_VERSE
```

別の文書から一節を引用する時、段落の左右の余白を空けることが慣習となっています。あなたは以下を用いることで引用を Org-mode の文書に含めることができます:

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

もしあなたがテキストを中央寄せにしたいなら、以下を使うことができます:

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

Footnote markup

脚注は Section 2.10 [Footnotes], page 16 で説明されたように定義されていて、全てのバックエンドにエクスポートされます。Org-mode は同じノートに対しての複数の参照と異なるバックエンドをサポートします。

Emphasis and monospace

あなたは***code***と`verbatim`、そして必要なら~~strile-through~~を単語に適用することができます。code と verbatim 文字列の中のテキストは Org-mode の明確な構文ではありません; それは verbatim にエクスポートされます。

Horizontal rules

少なくとも 5 文字のダッシュ文字のみで行成される線は水平線 (HTML では `<hr/>`、 \LaTeX では `\hrule`) にエクスポートされます。

コメント行

行頭の文字が `#` から始まる行はコメントとして扱われ、エクスポートされません。もしあなたがコメント行をインデントしたいのであれば、`#+` から行を開始してください。`COMMENT` ワードを持つサブツリーは、サブツリー全体がエクスポートされません。最後に、`#+BEGIN_COMMENT` から `END_COMMENT` で囲まれた範囲はエクスポートされません。

`C-c ;` エントリー先頭の `COMMENT` キーワードをトグルします。

11.2 画像と表

Org-mode ネイティブなテーブル (see Chapter 3 [Tables], page 19) と `table.el` パッケージを用いたテーブルの両方が適切にエクスポートされます。Org-mode の表では、最初の水平線の前の行が表のヘッダ行となります。あなたはキャプションと相互参照の指定を表の直前に、参照のための `\ref{tab:basic-data}` オブジェクトをテキストのどこかに書くことができます。

```
#+CAPTION: これは次の表 (またはリンク) のキャプションです
#+LABEL:   tbl:basic-data
| ... | ... |
|-----|----|
```

多くのバックエンド (HTML、 \LaTeX 、DocBook) はエクスポートされた文書の中に直接画像を挿入することができます。もし、例えば、`[[./img/a.jpg]]` のような説明部分を持たない画像ファイルへのリンクがあるなら、Org-mode は画像の挿入を行います。もしあなたが画像のキャプションや内部相互参照のラベルを定義したいなら、以下のように `#+CAPTION` と `#+LABEL` をリンクの前に書きます:

```
#+CAPTION: これは次の画像 (または表) のリンクのキャプションです。
#+LABEL:   fig:SED-HR4049
[[./img/a.jpg]]
```

あなたは画像に対する追加要素を定義するかもしれません。これはバックエンドの仕様なので、さらに情報が必要なら独立したバックエンドについてのセクションを見てください。

See Section 4.4 [Handling links], page 36.

11.3 Literal examples

あなたはマークアップに依存しないリテラルの例を含めることができます。そのような例に等幅のタイプセットがあり、それはソースコードやそれに似た例向きです。

```
#+BEGIN_EXAMPLE
テキストファイルからの例。
#+END_EXAMPLE
```

そのようなブロックはインデントされたテキストをうまく整列させるためと、特にプレーンリスト構造 (See Section 2.7 [Plain lists], page 13.) のためにインデントされるでしょう。小さな例を使う時、それを簡単にするために、あなたはコロンとそれに続く空白からなる例の行を使うことができます。それらはコロンの前に空白を追加することもできます。

ここに例を書きます

: テキストファイルからの例

もし例がソースコードなら、もしくは Emacs でフォントロックによりマークアップされたテキストなら、あなたは Emacs バッファ¹を要塞化するように要請することができます。あなたが例に色付けするために使うメジャーモードの名前を指定することが必要な時、‘src’ブロックを使います:

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

example と src スニペットでは、あなたは BEGIN の行の最後に -n を追加することで、例の行番号を表示することができます。もしあなたが +n とすると、前のスニペットから現在のものに番号が引き継がれます。リテラルの例で、Org-mode は ‘(ref:name)’ をラベルとして解釈し、[[(name)]] のような特別なリンクによりそこを参照することができます (i.e. 参照名は 1 つの括弧に囲まれています)。HTML では、対応するコード行をマウスオーバーすると自動的にハイライト表示になり、少しクールです。

また、ソースコード² からラベルを消去するかどうかの切り替えのために -r を追加することもできます。-n で切り替えると、リンクされるそれらのリファレンスはコードリスティングの行番号によってラベルを付けられ、そうでなければ括弧無しのラベルにリンクされます。

```
#+BEGIN_SRC emacs-lisp -n -r
(save-excursion                (ref:sc)
  (goto-char (point-min))      (ref:jump))
#+END_SRC
In line [[(sc)]] we remember the current position. [[(jump)]] [Line (jump)]
jumps to point-min.
```

もし、ラベルの構文が言語の構文と衝突した場合、-l を使うことで ‘#+BEGIN_SRC pascal -n -r -l \ "((%s))\' のようにフォーマットを変更できます。org-coderef-label-format 変数を見てください。

HTML はエクスポート時にテキストエリア, See Section 12.5.7 [Text areas in HTML export], page 131. とすることができます

C-c ’ カーソル位置のソースコード例をそのネイティブモードで編集します。これはソースコードを一時バッファに表示し、切り替えることで働きます。あなたは C-c ’ をもう一度押すことで編集を終了します³。編集されたバージョンは Org-mode バッファ上の古いバー

¹ HTML バックエンドに対しては、この作業は自動的に行われます (Org-mode と一緒に配布されている ‘htmlize.el’ のバージョン 1.34 が必要です)。LaTeX の要塞化されたコードの塊はリスティングか、minted (<http://coe.google.com/p/minted>) パッケージによってアーカイブされます。リスティングを使うには、org-export-latex-listings 変数をオンにし、LaTeX のヘッダにリスティングパッケージが含まれているようにします (例: org-export-latex-packages-alist の設定とを使います)。色付きの出力を含む設定のオプションについて、リスティングのドキュメントを見てください。minted を使うには、pygments (<http://pygments.org>) プログラムをインストールする必要があります。org-export-latex-minted を追加で設定し、LaTeX のヘッダに minted パッケージが含まれていることと -shell-escape オプションが ‘pdflatex’ に引き継がれている (org-latex-to-pdf-process を見てください) ことを確認します。

² Org-mode の例で説明するのに便利なリンクに行番号を使う間、-k を -n -r に追加することでソースコードのラベルを維持します。

³ 終了時、Org-mode によってアウトラインの見出しや特別なコメントと間違えられないようにするために、‘*’ か ‘#’ から始まる行はカンマが銭湯に追加されます。

ジョンを置き換えます。固定幅のリージョンは簡単に ASCII でイラストを書くための `artist-mode`⁴ を使うことで編集されます。空行でこのコマンドを使うことで、新しい固定幅のリージョンを作成します。

C-c l `C-c` 'によって作成した一時バッファでのソースコード例の編集中に `org-store-link` の呼び出しはラベルを指示します。現在のバッファがユニークであることを確認し、現在の行の最後に `(ref:label)` のように適切にフォーマットされたものが挿入されます。ラベルは `(label)` のようなリンクを記憶し、`C-c C-l` 検索する。

11.4 Include files

エクスポート中、あなたは別のファイルの内容をインクルードすることができます。例えば、`.emacs` をインクルードするなら、あなたは次のようにします:

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

2 目目のオプションは (e.g. `'quote'` や `'example'`、`'src'`) のようなマークアップで、3 目目はマークアップが `'src'` ならコンテンツの言語を表します。マークアップはオプションです; もし与えられなければ、Org-mode フォーマットのテキストと仮定される。インクルードの行は最初の行とそれに続く行のプレフィックスの指定のための追加キーワードパラメータの `:prefix1` と `:prefix` を、Org-mode のコンテンツを指定したレベル下げるための `:minilevel` を、同様に選択したマークアップ固有のオプションを持ちます。例えば、ファイルをインクルードするには:

```
#+INCLUDE: "~/snippets/xx" :prefix1 " + " :prefix " "
```

`:line` パラメータを使うことで、ファイルの指定した範囲の行のみをインクルードすることができます。範囲外の行はインクルードされません。範囲の開始と、または終了は明らかにデフォルトを使いません。

```
#+INCLUDE: "~/emacs" :lines "5-10"    Include lines 5 to 10, 10 excluded
#+INCLUDE: "~/emacs" :lines "-10"     Include lines 1 to 10, 10 excluded
#+INCLUDE: "~/emacs" :lines "10-"     Include lines from 10 to EOF
```

C-c ' ポイント位置のインクルードされたファイルに移動します。

11.5 Index entries

あなたは公開した文書のインデックスに用いるエントリーを規定することができます。これは `#+INDEX` から始まる行により設定します。感嘆符を含むエントリーはサブアイテムを作るでしょう。さらなる情報を見るには Section 13.1.8 [Generating an index], page 148 を参照してください。

```
* Curriculum Vitae
#+INDEX: CV
#+INDEX: Application!CV
```

11.6 Macro replacement

あなたはこのようにして

```
#+MACRO: name replacement text $1, $2 are arguments
```

which can be referenced anywhere in the document (even in code examples) with `{{{name(arg1,arg2)}}}`. In addition to defined macros, `{{{title}}}`, `{{{author}}}`, etc., will reference information set by the `#+TITLE:`, `#+AUTHOR:`, and similar lines. Also,

⁴ `org-edit-fixed-width-region-mode` 変数により、異なるモードを選択することもできます。

`{{date(FORMAT)}}` and `{{modification-time(FORMAT)}}` refer to current date time and to the modification time of the file being exported, respectively. *FORMAT* should be a format string understood by `format-time-string`.

マクロ展開はエクスポート中に行われ、一部の人は複雑な HTML コードの構築に用いる。

11.7 Embedded L^AT_EX

プレーンな ASCII はほとんどの場合ノートをとるのに十分です。例外は数学の記号や時々出てくる数式を必要とする科学に関するノートのようなものです。L^AT_EX⁵ は科学に関する文書の組版に広く使われています。多くの academics は L^AT_EX のソースコードの読み書きに使われていて、すぐに多くのエクスポートバックエンドに対応できるため、Org-mode は L^AT_EX コードのファイルへの組込みをサポートしています。

11.7.1 Special symbols

あなたは L^AT_EX マクロをギリシャ文字を表す `'\alpha'` や矢印を表す `'\tpo'` のような特殊記号の挿入に使うことができます。これらのマクロは補完が可能で、`'\'` まで入力し、その後何文字か入力して *M-TAB* を押すことで補完が可能です。L^AT_EX のコードとは違い、Org-mode は数学の区切り文字を囲まないようなマクロも使うことができます。以下に例を挙げます:

Angles are written as Geek letters `\alpha`, `\beta` and `\gamma`.

エクスポート時、これらのシンボルはエクスポート先のネイティブフォーマットに変換されます。HTML では `\alpha` のような文字列は `&\alpha`; にエクスポートされ、L^AT_EX では `α` となります。同様に、`\nbsp` は HTML では ` `; に、L^AT_EX では `~` となります。もしあなたが記号を単語の中に含めたいのであれば、次のようにします: `'\Aacute{ }stor'`。

非常に多くのエンティティが提供されていて、HTML と L^AT_EX からその名前を引き継いでいます; 完全なリストは `org-entities` 変数を見てください。`'\'` はシャイなハイフンとして扱われていて、`'--'` や `'---`、`'...'` は異なる長さのハイフンかドットの集合を作成するための全て特殊コマンドに変換されます。

もしあなたが UTF-8 文字でエンティティを表示したいのなら、以下のコマンド⁶:

`C-c C-x \` エンティティの UTF-8 での表示をトグルします。これはバッファの内容を変更せず、UTF-8 の文字を表示するためにオーバーレイを用いています。

11.7.2 Subscripts and superscripts

L^AT_EX と同じように、`'^'` と `'_'` が下付き文字と上付き文字を示しています。さらに、それらは *math-mode* にの区切り文字に組込まずに使うことができます。ASCII テキストの可読性の向上のため、複数字の下付き文字と上付き文字を波括弧で囲む必要はありません (囲んでもかまいませんが)。例

The mass of the sun is `M_sun = 1.989 x 10^30 kg`. The radius of the sun is `R_{sun} = 6.96 x 10^8 m`.

上付きテキスト、下付きテキストの説明を避けるため、あなたはバックスラッシュをつけた `'^'` と `'_'` を引用できます: `'\^'` と `'_'` です。異なる文脈でしばしば使われるアンダーラインのテキストを書くなら、常にこれらの下付き文字として解釈する Org-mode の慣習はあなたのやり方で得ることが

⁵ L^AT_EX は Donald Knuth の T_EX システムを基としたマクロシステムです。“L^AT_EX” で説明される多くの機能は T_EX からのものですが、違いはそれほどありません

⁶ あなたは `org-pretty-entities` 変数または `#+STARTUP` オプション `entitiespretty` にデフォルトを設定することができます

できます。この慣習を変更するには `org-export-with-sb-superscripts` 変数を設定するか、ファイルに次のように書いてください。

```
#+OPTIONS: ^:{}_
```

With this setting, ‘`a_b`’ will not be interpreted as a subscript, but ‘`a_{b}`’ will.

`C-c C-x \` さらに UTF-8 のエンティティを見るため、このコマンドは下付き文字と上付き文字を WYSISYM で形成する。

11.7.3 L^AT_EX の断片的なコード

シンボルと上付き、下付き、完全な式を越えることが必要です。Org-mode は L^AT_EX の数式を含むことができ、各エクスポート先への変換もサポートしています。L^AT_EX にエクスポートするとき、コードは明らかに残っています。HTML へエクスポートするとき、Org-mode は数式⁷ の処理と描画のために MathJax library (<http://www.mathjax.org>) (see Section 12.5.6 [Math formatting in HTML export], page 130) を呼び出します。最後に、数式表現はブラウザか DocBook 文書で描画可能な画像⁸ へと処理されます。

L^AT_EX のコード片は、特別なマークは全く必要ありません。以下のコード片は L^AT_EX のソースコードとして知られています:

- あらゆる種類の環境⁹。唯一必要なことは `\begin` 文は空白のみがある行に表示されることです。
- 通常の L^AT_EX の数学の区切り文字内部のテキスト。流通仕様との衝突を避けるために、囲まれたテキストに最大 2 つの改行が含まれている場合、‘`$`’ 文字は数学区切り文字のみとして認識され、‘`$`’ 文字がの間に空白がない、そして

For example:

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

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

もしあなたが他の目的に ASCII の区切り文字が必要なら、L^AT_EX コンバータに邪魔されえることを望まない文字を除外するために `org-format-latex-options` オプションを設定することができます。

L^AT_EX の処理は `org-export-with-LATEX-fragments` 変数を設定することができます。デフォルトの設定は `t` で、HTML には ‘`MathJax`’ を用い、DocBook と ASCII、L^AT_EX では処理しません。あなたはこの変数をファイルの冒頭部分に書くことで設定することもできます:

```
#+OPTIONS: LaTeX:t          Do the right thing automatically (MathJax)
#+OPTIONS: LaTeX:dvipng     Force using dvipng images
#+OPTIONS: LaTeX:nil        Do not process LATEX fragments at all
#+OPTIONS: LaTeX:verbatim  Verbatim export, for jsMath or so
```

⁷

⁸ これを行うには、あなたのシステムに L^AT_EX をインストールする必要があります。そしてまた、<http://sourceforge.net/projects/dvipng/> で入手できる ‘`dvipng`’ プログラムも必要です。

⁹ ‘`MathJax`’ が使われている時、‘`MathJax`’ によって認識されている環境が処理されます。‘`dvipng`’ を画像の生成に用いる時、L^AT_EX 環境が扱われます。

11.7.4 Previewing LaTeX fragments

もしあなたが‘dvipng’をインストールしているのであれば、 \LaTeX のコード片は出力された組版において画像として処理されます:

`C-c C-x C-l`

ポイント位置の \LaTeX コード片の画像プレビューの提供とソースコード上のオーバーレイ。もしポイント位置にコード片がないのであれば、現在のエントリ (2つの見出しの間) の全てのコード片を処理します。前置引数を付けて呼ばれた時は、サブツリー全体を処理します。前置引数を2つ付けて呼ばれた時、またはカーソルが最初の見出しの前にある時は、バッファ全体を処理します。

`C-c C-c` オーバーレイされたプレビュー画像を消去します。

プレビューの外観を変更するために、あなたは `org-format-latex-options` 変数をカスタマイズすることができます。とりわけ、`:scale` (そして HTML へのエクスポートでは `:html-scale`) プロパティは画像のプレビューサイズの調整に使われます。

11.7.5 CDLaTeX を数学の入力に使う

CDLaTeX モードは環境や数学テンプレートの挿入をスピードアップするために AUCTeX に似たメジャーモードである \LaTeX モードと併用して通常使われるマイナーモードです。Org-mode では、あなたは CDLaTeX モードのいくつかの機能を使用できます。あなたは <http://www.astro.uva.nl/~dominik/Tools/cdlatex> から ‘`cdlatex.el`’ と ‘`texmathp.el`’ (最近 AUCTeX に追加されました) をインストールする必要があります。Org-mode 中では CDLaTeX モード自身は使わないでください、代わりに Org-mode に一部である、より軽量のバージョンの `org-cdlatex-mode` を使ってください。M-x `org-cdlatex-mode` をカレントバッファで実行して有効にするか、全ての Org-mode ファイルで有効するために次の設定を行います:

```
(add-hook 'org-mode-hook 'turn-on-org-cdlatex)
```

このモードが有効である時、以下の機能が提供されます (詳細は CDLaTeX モードのドキュメントを参照してください)::

- `C-c {` による環境テンプレートの挿入。
- カーソルが \LaTeX のコード片¹⁰ の中にある場合、TAB キーはテンプレートの展開を行います。例えば、TAB は `fr` を `\frac{}{}` に展開しカーソルを最初の括弧に移動します。もう一度 TAB を押すと2つ目の括弧にカーソルが移動します。コード片の外だと、TAB は行の先頭にある環境の略語を展開します。例えば、もしあなたが行頭に ‘`equ`’ と書いていて TAB を押すと、この略語は `equation` 環境に展開されます。全ての略語を見るには、M-x `cdlatex-command-help` をタイプしてください。
- \LaTeX コード片の中で `_` と `^` を押すと、それらの文字が括弧のペアと一緒に挿入されます。もしあなたが TAB を括弧から抜け出すために使うなら、また括弧が1文字の文字かマクロのみを囲っているなら、それらは再び消去されます (`cdlatex-simplify-sub-super-script` 変数に依存します)。
- Pressing the backquote ‘ followed by a character inserts math macros, also outside \LaTeX fragments. If you wait more than 1.5 seconds after the backquote, a help window will pop up.

¹⁰ カーソルがコード片の中にあるときに Org-mode はテストを行うためのメソッドを持ちます。詳細は `org-inside-LaTeX-fragment-p` 関数のドキュメントを参照してください。

- 別の文字に続いてシングルクォート'を押すと、強調やフォントでポイント前のシンボルが変更されます。もしシングルクォートを入力した後 1.5 秒以上待つと、ヘルプウィンドウがポップアップします。文字の変更は L^AT_EX コード片の中でのみ働きます; それ以外ではクォートは通常通りの働きをします。

12 Exporting

org-mode のドキュメントは様々なフォーマットにエクスポートすることができます。ノートを共有し印刷するには ASCII 形式でエクスポートすることで Org ファイルの読みやすく、シンプルなものが得られます。HTML のエクスポートではノートをウェブに公開できるようになりますし、XOXO フォーマットは他の様々なアプリケーションでやりとりするうえで確かな基礎となります。L^AT_EX のエクスポートでは、org-mode とその構造化された編集機能を使って、容易に L^AT_EX のファイルを出力することができます。DocBook のエクスポートでは、Org ファイルを DocBook のツールを使った様々なフォーマットに変換することが可能です。プロジェクトの管理では、TaskJuggler 形式のエクスポートを使って、ガントリソースチャートを作成することができます。デッドラインや予約のような時間と関連のあるエントリーを iCal のようなデスクトップカレンダーに取り込むために org-mode は iCalendar 形式で抽出することもできます。現在、Org-mode はエクスポートのみをサポートしており、他の異なるフォーマットからインポートすることはできません。

org-mode は、`transient-mark-mode` がオンの時 (Emacs 23 ではデフォルト)、は選択したリージョンをエクスポートをすることができます。

12.1 Selective export

エクスポートしたいドキュメントのある部分を選択、または除外する時にタグを使うことができます。その挙動は、`org-export-select-tags` と `org-export-exclude-tags` の二つの変数により決まります。

org-mode はまず最初に *select* タグがバッファにないかチェックします。あった場合は、タグがない全てのツリーは除外されます。もし選択したツリーがサブツリーだった場合、それより上の階層はエクスポートされるものとして選択されますが、それより下の階層は選択されません。

もし、選択されたタグがなかった場合、バッファにある全ての内容がエクスポートされるものとして選択されるでしょう。

最後に、*exclude* タグでマークされていない全てのサブツリーはエクスポートバッファから除かれるでしょう。

12.2 Export options

エクスポートする際にはバッファにある特別な行が読みこまれます。その行には追加的な情報が含まれており、ファイルの中でどこにでも書くことができます。`C-c C-e t` と入力することで、バッファにそのような行をセットで挿入することができます。それぞれの行で `#+` と入力した後に `M-TAB` による補完を行ない、(see Section 15.1 [Completion], page 174) キーワードが正しいか、確認してみると良いでしょう。エクスポートと関連のない、バッファ内の設定の概要については Section 15.6 [In-buffer settings], page 176 を参照してください。特に、`#+SETUPFILE` を使うことによって含めることができる別のファイルの中でよく使われる (エクスポートの) オプションを指定できることに注意してください

`C-c C-e t` `org-insert-export-options-template`
エクスポートオプションのテンプレートを挿入します。下の例を見てください。

```
#+TITLE:          表示されるタイトル (デフォルトはバッファ名)
#+AUTHOR:         著者 (デフォルトは user-full-name の値)
#+DATE:           format-time-string で解釈される固定された日付の文字列
#+EMAIL:          彼/彼女のメールアドレス (デフォルトは user-mail-address の値)
```

```

#+DESCRIPTION: ページの説明, e.g. XHTML のメタタグで使われる。
#+KEYWORDS:    ページのキーワード, e.g. XHTML のメタタグで使われる。
#+LANGUAGE:    HTML で指定される言語
                e.g. 'en' (org-export-default-language)
#+TEXT:        冒頭に挿入される説明的な文章
#+TEXT:        複数の行に書くことができます。
#+OPTIONS:     H:2 num:t toc:t \n:nil @:t ::t |:t ^:t f:t TeX:t ...
#+BIND:        lisp-var lisp-val, e.g.: org-export-latex-low-levels itemize
                これらを確認するか, org-export-allow-BIND を設定すること
#+LINK_UP:     出力したページにおける ``up'' のリンク先
#+LINK_HOME:   出力したページにおける ``home'' のリンク先
#+LATEX_HEADER: LaTeX のヘッダーで使われる \usepackage{xyz} のような余分な
                行
#+EXPORT_SELECT_TAGS: エクスポートするツリーを示すタグ
#+EXPORT_EXCLUDE_TAGS: エクスポートから除外するツリーを示すタグ
#+XSLT:        FO ファイルを生成するのに DocBook のエクスポート機能が使う XSLT の
                スタイルシート

```

OPTIONS 行は 以下のようなエクスポートの設定を示すコンパクトな式です。¹

```

H:            エクスポートする見出しの階層数
num:          セクション番号の有無
toc:          目次の有無, または階層数の上限 (整数)
\n:          改行を維持するかどうか (うまく動作しない)
@:            HTML の引用タグの有無
::           固定幅の段落の有無
|:           表の有無
^:           上付き、下付き文字を示す TEX のようなシンタックスの有無
              "^^:{" は a_{b} 解釈されるが、
              簡潔な a_b はそのままとなるでしょう。
-:           特別な文字列を変換するかどうか
f:           this[1] のような脚注を用いるかどうか
todo:        TODO キーワードを出力した文字列に含めるかどうか
pri:         クッキーを優先するかどうか
tags:        タグの有無, not-in-toc となるかもしれません。
<:          DEADLINES のような時間/日付の有無
*:           強調テキストの有無 (太字, イタリック, アンダーライン)
TeX:         テキスト中のシンプルな TEX マクロの有無
LaTeX:       LATEX 出力の設定 デフォルトは auto
skip:        最初見出しの前にある文章をスキップするかどうか
author:      著者の名前/e-mail を出力するかどうか
email:       著者の e-mail を出力するかどうか
creator:     作者を出力するかどうか
timestamp:   作成した日付を出力するかどうか
d:           drawer を出力するかどうか

```

¹ もし、このように多くのオプションを設定したい時は、それぞれオプション行を作りことができます。

これらのオプションは HTML、 \LaTeX の両方のエクスポートに影響します。TeX と LaTeX のオプションを除き \LaTeX のエクスポートをするのに、それぞれ `t`、または `nil` となります。

`org-export-html-pre/postamble` を `t` とすると HTML にエクスポートする時に `author`、`email` 及び `creator` の値は上書きされるでしょう。代わりに `org-export-html-pre/postamble-format` が用いられます。

このようなオプションの初期値は変数のセットで与えられます。そのような変数は、`OPTIONS` のキーと公開するキーにも対応しています。(see Section 13.1.1 [Project alist], page 144), `org-export-plist-vars` の定数を見てください。

エクスポートのコマンドを呼び出す前に、`C-c @` で選択した単一のサブツリーをエクスポートする時、そのサブツリーは、`EXPORT_FILE_NAME`、`EXPORT_TITLE`、`EXPORT_TEXT`、`EXPORT_AUTHOR`、`EXPORT_DATE`、そして `EXPORT_OPTIONS` プロパティでエクスポートの設定を無視することができます。

12.3 The export dispatcher

全てのエクスポートコマンドはエクスポートコマンド選択画面から選ぶことができます。コマンド選択画面では、コマンドを特定するための追加的なキーの入力を促されます。通常、ファイルの全ての内容がエクスポートされますが、もしアクティブなリージョンに一つのアウトラインツリーが含まれていた場合、まず、見出しがドキュメントのタイトルとして扱われ、サブツリーがエクスポートされます。

C-c C-e **org-export**
エクスポート、または公開のコマンド選択画面です。エクスポート、または公開のコマンドを起動するのに必要なキーがヘルプウィンドウに表示されます。前置引数として、入力すると、直接エクスポート機能となります。二重の前置引数 `C-u C-u` を入力することで、コマンドは別の Emacs プロセスにおいてバックグラウンドで実行されます。²

C-c C-e v **org-export-visible**
`C-c C-e` のように動作しますが、今見えている文章だけがエクスポートされます。(i.e. アウトライン表示により、隠されていない文章)。

C-u C-u C-c C-e **org-export**
エクスポート機能が呼ばれますが、`org-export-run-in-background` の設定と逆の挙動となります。i.e. 動いていないバックグラウンドプロセスを呼びだしたり、現在の Emacs のプロセスで強制的に実行したりします。

12.4 ASCII/Latin-1/UTF-8 export

ASCII 形式へのエクスポートは、`org-mode` のファイルを ASCII のみが含まれる、シンプルで読みやすい形に書き出します。Latin-1 及び UTF-8 でのエクスポートでは特殊な文字やシンボルをそれらのエンコードで出力します。

C-c C-e a **org-export-as-ascii**
ASCII 形式のファイルをエクスポートします。Org ファイルを `'myfile.org'` だとすると、ASCII 形式のファイルは `'myfile.txt'` となるでしょう。そのファイルは警告なしに上書きされます。もしアクティブなリージョン³ があった場合、そのリージョンのみがエクスポートされます。選択したリージョンが一つのツリー⁴ を含んでいた場合、

² このような挙動をデフォルトにするには、`org-export-run-in-background` 変数を設定してください。

³ `transient-mark-mode` が有効である必要があります。

⁴ 現在のサブツリーの選択するには、`C-c @` と入力してください。

そのツリーの見出しがドキュメントのタイトルとなるでしょう。見出しがあるか、または `EXPORT_FILE_NAME` プロパティを継承していた場合、エクスポートするにはその名前が使われるでしょう。

`C-c C-e A` `org-export-as-ascii-to-buffer`
一時的なバッファに出力し、ファイルを作成しません。

`C-c C-e n` `org-export-as-latin1`
`C-c C-e N` `org-export-as-latin1-to-buffer`
上に示したコマンドのような動作をしますが、Latin-1 でエンコーディングされたものが出力されます。

`C-c C-e u` `org-export-as-utf8`
`C-c C-e U` `org-export-as-utf8-to-buffer`
上に示したコマンドのような動作をしますが、UTF-8 でエンコーディングされたものが出力されます。

`C-c C-e v a/n/u`
文書の中で、バッファで表示されている部分だけを出力する。

エクスポートされたものでは、最初の3つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層はアイテムのリストとしてエクスポートされます。この違いを異なる階層に変えたい場合は、前置引数で、その階層を指定します。例えば、

`C-1 C-c C-e a`

は最初の階層のみを見出しとし、それ以外はアイテムとなります。見出しがアイテムに変更された時、見出し後の文章のインデントは、アイテムの下にうまく調和するように変更されます。この変更は、最初の本文が全体のインデントを示しているという仮定のもとで実行されます。これよりも大きなインデントは、最初の文章との相対的なレイアウトを維持するように調整されます。最初の行より少ないインデントであれば、左寄せします。

次の見出しの前にあるリンクは脚注のような形でエクスポートされます。その脚注は、次の見出しの前に項目名とリンクがエクスポートされます。詳しい内容と他のオプションについては、変数 `org-export-ascii-links-to-notes` をご覧ください。

12.5 HTML export

`org-mode` には多くの HTML のフォーマットに対応した HTML (XHTML 1.0 準拠) エクスポート機能があります。それは、John Gruber が開発した *markdown* 言語に似ていますが、`org-mode` ではさらにテーブルもサポートしています。

12.5.1 HTML エクスポートのコマンド

`C-c C-e h` `org-export-as-html`
HTML ファイル `'myfile.html'` をエクスポートします。Org ファイル `'myfile.org'` をエクスポートすると、ASCII 形式のファイルは `'myfile.html'` となるでしょう。そのファイルは警告なしに上書きされます。もしアクティブなリージョン⁵ があった場合、そのリージョンのみがエクスポートされます。選択したリージョンが一つのツリー⁶ を含んでいた場合、そのツリーの見出しがドキュメントのタイトルとなるでしょう。見出

⁵ `transient-mark-mode` が有効である必要があります。

⁶ 現在のサブツリーの選択するには、`C-c @` と入力してください。

しがあるか、または `EXPORT_FILE_NAME` プロパティを継承していた場合、エクスポートする際にはその名前が使われるでしょう。

C-c C-e b **org-export-as-html-and-open**
HTML ファイルをエクスポートし、そのファイルをブラウザで開きます。

C-c C-e H **org-export-as-html-to-buffer**
一時的なバッファに出力し、ファイルを作成しません。

C-c C-e R **org-export-region-as-html**
アクティブなリージョンを一時的なバッファに出力します。前置引数があるとヘッダーとフッターを出力せずに、リージョンの HTML のみを出力します。これはカットアンドペーストで編集する際に便利です

C-c C-e v h/b/H/R
文書の中で、バッファで表示されている部分だけを出力する。

M-x org-export-region-as-html
`org-mode` の記法が使われているという前提でリージョンを HTML に変換します。これはどのバッファでも起動するグローバルなコマンドです。

M-x org-replace-region-by-HTML
`org-mode` の記法が使われているという前提でアクティブなリージョンを HTML に変換します。

エクスポートされたものでは、最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層はアイテムのリストとしてエクスポートされます。この違いを異なる階層に変えたい場合は、前置引数で、その階層を指定します。例えば、

C-2 C-c C-e b

この場合 2 番目のレベルまでを見出しとして取り扱い、それ以外は項目として取り扱います。

12.5.2 Quoting HTML tags

HTML にエクスポートする際、プレインな '`<`' and '`>`' は常に '`<`' と '`>`' に変換されます。もし単純な HTML タグをそのまま含めたい時は、'`@bold text@`' のように `ma@` でマークします。これは単純な HTML タグでしか動作しませんので注意してください。エクスポートするファイルにさらに広範囲な HTML をそのままコピーするには次のようなブロックが使えます。

#+HTML: エクスポートする HTML コード

or

マーカー間の全ての行は文字どおり出力されます。

12.5.3 Links in HTML export

内部リンク (see Section 4.2 [Internal links], page 34) エクスポートされ HTML でも同様に動作します。これには、ラジオターゲット (see Section 4.2.1 [Radio targets], page 35) により生成された自動リンクも含まれます。もしターゲットとなるファイルが公開される Org ファイルを示す同じ相対パス上にあっても、リンクは外部リンクとして動作するでしょう。他の '`.org`' ファイルへのリンクは、HTML にエクスポートされたものにも同じ相対パスでリンクされたファイルがある、という前提で、リンクに変換されます。'`id:`' リンクはファイル間で特定のエントリーにジャンプするのに使われます。リンクするファイル、公開ディレクトリでの公開に関する情報については、Section 13.1.6 [Publishing links], page 147 参照してください。

リンクの属性を記述したい時は、特別な#+ATTR_HTML行を用いることができます。この行は、<a>タグやタグを追加する属性を定義するために使われます。以下の例では、リンクに `title` と `style` の属性を設定しています。

```
#+ATTR_HTML: title="The Org-mode homepage" style="color:red;"
[[http://orgmode.org]]
```

12.5.4 Tables

org-mode の表は、`org-export-html-table-tag` で定義されているテーブルのタグを使って HTML にエクスポートされます。デフォルトの設定では、セルの罫線とフレームがない状態でテーブルが出力されます。個々のテーブルでその設定を変えたい場合は、次のような行をテーブルの前に記述してください。

```
#+CAPTION: これはセルの周囲に線が引かれた表です。
#+ATTR_HTML: border="2" rules="all" frame="all"
```

12.5.5 Images in HTML export

HTML のエクスポートでは Org ファイルにリンクがある画像をインライン表示することができます。その画像はリンクされているクリック可能な部分として扱われます。デフォルトでは、⁷、リンクに `description` がなければ、画像はインライン表示されます。つまり、`'[[file:myimg.jpg]]'` はインライン表示されますが、`'[[file:myimg.jpg][the image]]'` はが画像にリンクされる `'the image'` というテキストリンクが作られます。`description` の部分が `file:` リンクか画像を示す `http:` の URL の場合、画像はインラインに表示され、画像がクリックされると活性化されます。例えば、リンク先に高解像度の画像があるサムネイルを追加したい場合、次のように書くといいでしょう。

```
[[file:highres.jpg][file:thumb.jpg]]
```

インライン画像に属性を追加したい場合は、#+ATTR_HTML を使います。次の例では、テキストでの見やすさとアクセスのしやすさを考慮して `alt` 属性と `title` 属性を指定して、`align` を右にしています。

```
#+CAPTION: A black cat stalking a spider
#+ATTR_HTML: alt="cat/spider image" title="Action!" align="right"
[[./img/a.jpg]]
```

`http` のアドレスも使うことができます。

12.5.6 Math formatting in HTML export

L^AT_EX の数学用スニペット (see Section 11.7.3 [LaTeX fragments], page 122) は二つの異なる方法で HTML に表示される。デフォルトでは org-mode をインストールすると、すぐに MathJax system (<http://www.mathjax.org>) が使えるようになっています。<http://orgmode.org> は 'MathJax' が Org-mode ユーザ、小さなアプリケーション、そしてテストにとって便利だと考えているからです。もし特定のページで、あるいは常に 'MathJax' を使うのであれば、私達のサーバでの読みこみを減らすために **MathJax** をあなたのサーバにインストール⁸ してください。'MathJax' について設定するには、`org-export-html-mathjax-options` を使うか、バッファに次のような行を挿入してください。

```
#+MATHJAX: align:"left" mathml:t path:"/MathJax/MathJax.js"
```

⁷ ただし、`org-export-html-inline-images` を確認してください。

⁸ インストール方法については、MathJax のウェブサイトにあります。<http://www.mathjax.org/resources/docs/?install> を参照してください。

See the docstring of the variable この行の各パラメータの意味の知るための `org-export-html-mathjax-options`

望むのであれば、 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ を小さな画像に変換してブラウザ上のページに挿入することもできます。MathJax が有用である前には、これが `org-mode` でのデフォルトの方法でした。この方法を用いるには、あなたのシステムで `'dvipng'` プログラムが利用できる状態である必要があります。この方法は以下のような行を追加することでも有効になります。

```
#+OPTIONS: LaTeX:dvipng
```

12.5.7 Text areas in HTML export

コードサンプルを HTML にして公開する方法として、テキストエリアを使う方法があります。何かのアプリケーションに貼りつける前であれば、そのコードサンプルは編集することができます。example ブロックか `src` ブロックに `-t` スイッチが付加されることでテキストエリアに変換されます。このスイッチを使うことで、シンタックス、ラベルのハイライト、行番号に関するオプションが無効になります。`-h` と `-w` を使うことがあるかもしれません。それらのスイッチはテキストエリアの高さと幅を特定するもので、デフォルトでは高さが example ブロックの行数で幅は 80 となります。設定は、例えば以下のようになります。

```
#+BEGIN_EXAMPLE -t -w 40
(defun org-xor (a b)
  "Exclusive or."
  (if a (not b) b))
#+END_EXAMPLE
```

12.5.8 CSS support

エクスポートするファイルには、スタイルに関する情報を含めることができます。HTML エクスポート機能には、文章のパーツを適切に表示するために次に示す特別な CSS クラス⁹ があります。見出しやテーブルなどの標準的なクラスに加えて、それら特別な CSS クラスも変更することができます。

<code>p.author</code>	著者の情報、email 含む
<code>p.date</code>	公開日
<code>p.creator</code>	作成情報, <code>org-mode</code> のバージョン
<code>.title</code>	文章のタイトル
<code>.todo</code>	DONE となっていない TODO キーワード
<code>.done</code>	DONE キーワード、DONE と扱われる全てのキーワードが対象
<code>.WAITING</code>	各 TODO キーワードはその名前のクラス名も用いることができる
<code>.timestamp</code>	タイムスタンプ
<code>.timestamp-kwd</code>	SCHEDULED 等のタイムスタンプに関連するキーワード
<code>.timestamp-wrapper</code>	SCHEDULED 等のキーワードとタイムスタンプ全体
<code>.tag</code>	見出し中のタグ
<code>._HOME</code>	各タグはその名前のクラス名も用いることができる (" <code>\"_\"</code> に置き換えられる)
<code>.target</code>	リンクのターゲット
<code>.linenr</code>	コード中の行番号
<code>.code-highlighted</code>	参照されコード行のハイライト

⁹ TODO キーワードやタグに CSS が適用されるとコンフリクトを起こします。`org-export-html-todo-kwd-class-prefix` と `org-export-html-tag-class-prefix` を使って、それらをユニークにしてください。

<code>div.outline-N</code>	深さレベル N の div 要素 (見出しとテキスト)
<code>div.outline-text-N</code>	深さレベル N のテキスト部分の div 要素
<code>.section-number-N</code>	深さレベル N の見出しの番号。各レベルで異なる
<code>div.figure</code>	インライン画像のフォーマット方法
<code>pre.src</code>	ソースコードブロックのフォーマット方法
<code>pre.example</code>	例示ブロック
<code>p.verse</code>	verse ブロック
<code>div.footnotes</code>	脚注の見出し
<code>p.footnote</code>	脚注定義の文章、脚注を含む
<code>.footref</code>	脚注の参照番号 (常に<sup>となる)
<code>.footnum</code>	脚注定義中の番号 (常に<sup>となる)

エクスポートされたファイルは、基礎的な方法で定義されたコンパクトなスタイル¹⁰ が含まれています。この設定は上書きされるかもしれませんが、`org-export-html-style` (Org-wide の設定に使われます) や `org-export-html-style-extra` (ファイルごとの設定のような詳細な設定に使われます。) を使って追加されるかもしれません。後者の変数をファイルごとに設定するには、次のように行ないます。

```
#+STYLE: <link rel="stylesheet" type="text/css" href="stylesheet.css" />
```

長いスタイルの定義には複数行で記述することもできます。外部ファイルを参照せずに<style></style>セクションに直接記述してください。

サブツリーにスタイルを追加するには、ツリーにクラスを適用する `:HTML_CONTAINER_CLASS:` プロパティを使います。個々の見出しに CSS スタイルを適用するには、`:CUSTOM_ID:` プロパティで指定される ID を使うことができます。

12.5.9 ウェブページの表示に関する JavaScript のサポート

Sebastian Rose は、org-mode が生成した HTML ファイルに関するウェブエクスペリエンスを拡張するためにデザインされた Javascript プログラムを書きました。このプログラムを使うことで、異なる二つの方法で大きなファイルを見ることができます。一つめは *Info* のようなモードで、それぞれの章は別々に表示され、`n` キーと `p` キーで操作できます。(他のキーでも操作できます。利用できるキーの概要を知るには、`?` を入力してください。)。二つめは、org-mode が Emacs で提供するような折りたたまれたスタイルです。このスクリプトは、<http://orgmode.org/org-info.js> で利用できます。ドキュメントについては、<http://orgmode.org/worg/code/org-info-js/> にあります。このスクリプトは私達のサイトでホスティングしていますが、何度も使う場合は、orgmode.org にあるものを使わずにあなたのサーバにコピーしたものを使う方を選択するかもしれません。

このスクリプトを使うには、`'org-jsinfo.el'` がロードされているか、確認する必要があります。デフォルトでは、ロードされるようになっていますが、`M-x customize-variable RET org-modules RET` と入力して、確かにロードされている確認してください。このプログラムを使えるようにするには、次のような行を Org ファイルに追加するだけです。

```
#+INFOJS_OPT: view:info toc:nil
```

ファイル中にこの行が見つかり、HTML のヘッダーは自動的にこのスクリプトを起動させるのに必要なコードを自動的に追加します。以上のような行を使うと、次のようなオプションを設定できます。

`path:` スクリプトのパス。デフォルトでは、<http://orgmode.org/org-info.js> を使うようになっていますが、ローカルにコピーしたものを使いたい場合は

¹⁰ このスタイルは `org-export-html-style-default` で定義されており、変更できません。この初期設定を無効にするには `org-export-html-style-include-default` を修正してください。

‘`../scripts/org-info.js`’のようなパスを使ってください。

view: ウェブサイトを最初に開いた時の表示。可能な値は次のとおり:

- info** 一つのページに一つのセクションが表示される Info のようなインターフェイス
- overview** 最初はトップレベルのみが表示される折りたたみインターフェイス
- content** 全ての見出しが見える状態の折りたたみインターフェイス
- showall** 全ての見出しと文章が見える状態の折りたたみインターフェイス

sdepth: info や折りたたみモードで独立して表示されるセクションの最大の見出しレベル。デフォルトでは `org-export-headline-levels` (= `#+OPTIONS`の中の `H`スイッチ) の値が使われる。もし、`org-export-headline-levels`の値より小さかった場合、info/折りたたみ のセクションは小見出しまで含まれます。

toc: 目次表示の有無
`nil`としても、`i`を入力することで目次は表示されます。

tdepth: 目次の深さ。デフォルトでは、`org-export-headline-levels` `org-export-with-toc`の値が用いられます。

ftoc: CSS によって、目次の場所を指定するかどうか。
 「yes」の場合は、セクションとして表示されなくなります。

ltoc: それぞれのセクションにショートコンテンツを設置するかどうか。
 セクションの冒頭にショートコンテンツを設置する場合は値を `above` とします。

mouse: マウスを見出しの上に移動させた時にハイライトさせます。
 ‘underline’ (default) か、‘#cccccc’のように背景色が指定できます。

buttons: ビューの変更をトグルさせるボタンを様々なところに設置するかどうか。
`nil`の場合は、(デフォルト)、ボタンが一つだけ表示されます。

`org-infojs-options`を変更することで、これらのオプションの初期値を変更することができます。このスクリプトを常にページに適用させたい場合は、`org-export-html-use-infojs`を変更してください。

12.6 L^AT_EX と PDF のエクスポート

`org-mode` には、Bastien Guerry によって書かれた L^AT_EX のエクスポート機能があります。追加的な処理と合わせて、¹¹、このバックエンドは PDF の出力にも使われています。L^AT_EX の出力は、リンクと相互参照の実装に ‘hyperref’ を使っているので、出力された PDF ファイルは完全にリンクされているでしょう。セクションの階層に合わせて正しく出力されるためには、org ファイルは適切に構造化されていないといけないので注意してください。

12.6.1 L^AT_EX エクスポートのコマンド

`C-c C-e l` `org-export-as-latex`
 L^AT_EX ファイル ‘myfile.tex’を出力します。Org ファイルに対して ‘myfile.org’、ASCII ファイルは ‘myfile.tex’となるでしょう。そのファイルは警告なしに上書きさ

¹¹ デフォルトの LaTeX 出力は、pdf_{tex} または latex により出力されるよう設計されています。それには、xetex や恐らく luatex と互換性のないパッケージが含まれています。`org-export-latex-default-packages-alist`や `org-export-latex-packages-alist`を参照してください。

れます。アクティブなリージョン¹²があれば、そのリージョンのみが出力されるでしょう。選択したリージョンが一つのツリー¹³であった場合、ツリーの見出しがタイトルになります。ツリーの見出しのエントリーが `EXPORT_FILE_NAME` プロパティを継承、または持っている場合、エクスポートされる際には、その名前が使われるでしょう。

`C-c C-e L` `org-export-as-latex-to-buffer`
一時バッファに出力します。ファイルを作りません。

`C-c C-e v l/L`
文書の中で、バッファで表示されている部分だけを出力する。

`M-x org-export-region-as-latex`
Org-mode の記法が使われているという前提でリージョンを `LATEX` に変換します。これはどのバッファでも起動するグローバルなコマンドです。

`M-x org-replace-region-by-latex`
アクティブなリージョンを (Org-mode の記法が使われている前提で) `LATEX` コードに置き変えます。

`C-c C-e p` `org-export-as-pdf`
`LATEX` に出力し、PDF にも変換します。

`C-c C-e d` `org-export-as-pdf-and-open`
`LATEX` に出力し、PDF にも変換します。その際出力された PDF ファイルを開きます。

エクスポートされたものでは、最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層は概要のリストとしてエクスポートされます。エクスポート機能では、`org-latex-low-levels` を変更することで、この設定を無視、または変更することができます。

この違いを異なる階層で変えたい場合は、前置引数で、その階層を指定します。例えば、

`C-2 C-c C-e l`

この場合 2 番目のレベルまでを見出しとして取り扱い、それ以外は項目として取り扱います。

12.6.2 見出しと構造の分割

デフォルトでは、`LATEX` の出力には `article` クラスが使われます。

クラスは `org-export-latex-default-class` の値を変更することで、全体的に変更することもできますし、ファイル中に `org-export-latex-default-class` のようなオプションを追加することで、局所的に変更することもできます。`:LaTeX_CLASS:` プロパティを使えば、エクスポートするリージョンにそのツリー (サブツリー) のみが含まれていた場合にクラスを指定できます。クラスは、`org-export-latex-classes` にリストアップされています。この変数は、各クラス¹⁴ の見出しテンプレートを定義し、各クラスの構造の分割について定義します。クラス自体についても定義されます。`#+LaTeX_CLASS_OPTIONS`、または `LaTeX_CLASS_OPTIONS` プロパティは `\documentclass` マクロのオプションを指定します。見出しに `#+LATEX_HEADER: \usepackage{xyz}` を追加して同様のことをすることもできます。詳しい情報については、`org-export-latex-classes` のドキュメント文字列を参照してください。

¹² `transient-mark-mode` が有効である必要があります。

¹³ 現在のサブツリーを選択するには、`C-c @` を入力してください。

¹⁴ `org-export-latex-default-packages-alist` と `org-export-latex-packages-alist` が接合されたものです。

12.6.3 L^AT_EX コードの引用

Section 11.7 [Embedded L^AT_EX], page 121 で記述された埋め込まれた L^AT_EX は、L^AT_EX に正しく挿入されます。図の相互参照を生成するために、`\ref{LABEL}` のようなシンプルなマクロが含まれます。さらに、次のような行を追加することで、L^AT_EX エクスポートの際に表示だけしてほしい特別なコードを追加することができます。

```
#+LaTeX: エクスポートする際に文字のまま、出力される LaTeX code
```

or

```
#+BEGIN_LaTeX
```

```
マーカの間にある全ての行は文字がそのまま出力されます。
```

```
#+END_LaTeX
```

12.6.4 L^AT_EX エクスポートにおける表

L^AT_EX で表を出力する際に、ラベルと表題を指定することができます (see Section 11.2 [Images and tables], page 118)、ATTR_LaTeX行を使うことで、表に関する `longtable` 環境を呼び出すこともできます。複数のページにまたがる表や、デフォルトの表の環境を `table` から `table*` にするため、またはデフォルトの内部 `tabular` 環境を `tabularx` や `tabulary` にしたい時にも ATTR_LaTeX行は使われます。つまり、文字の配置や (`tabularx` や `tabulary` を使って) 幅を次のようにして設定できます。:

```
#+CAPTION: A long table
```

```
#+LABEL: tbl:long
```

```
#+ATTR_LaTeX: longtable align=1|lp{3cm}r|l
```

```
| ..... | ..... |
```

```
| ..... | ..... |
```

`tabulary` を使って、複数のセルにまたがる表を指定することもできます。

```
#+CAPTION: A wide table with tabulary
```

```
#+LABEL: tbl:wide
```

```
#+ATTR_LaTeX: table* tabulary width=\textwidth
```

```
| ..... | ..... |
```

```
| ..... | ..... |
```

12.6.5 L^AT_EX エクスポートにおける画像

Images that are linked to without a description part in the link, like `[[file:img.jpg]]` or `[[./img.jpg]]` will be inserted into the PDF output file resulting from L^AT_EX processing. Org will use an `\includegraphics` macro to insert the image. If you have specified a caption and/or a label as described in Section 11.2 [Images and tables], page 118, the figure will be wrapped into a `figure` environment and thus become a floating element. You can use an `#+ATTR_LaTeX:` line to specify the various options that can be used in the optional argument of the `\includegraphics` macro. To modify the placement option of the `figure` environment, add something like `placement=[h!]` to the Attributes.

If you would like to let text flow around the image, add the word `'wrap'` to the `#+ATTR_LaTeX:` line, which will make the figure occupy the left half of the page. To fine-tune, the `placement` field will be the set of additional arguments needed by the `wrapfigure` environment. Note that if you change the size of the image, you need to use compatible settings for `\includegraphics` and `wrapfigure`.

```
#+CAPTION: The black-body emission of the disk around HR 4049
```



```

#+LABEL:      fig:SED-HR4049
#+ATTR_LaTeX: width=5cm,angle=90
[./img/sed-hr4049.pdf]]

#+ATTR_LaTeX: width=0.38\textwidth wrap placement={r}{0.4\textwidth}
[./img/hst.png]]

```

If you need references to a label created in this way, write ‘`\ref{fig:SED-HR4049}`’ just like in \LaTeX .

12.6.6 Beamer クラスのエクスポート

The \LaTeX class ‘**beamer**’ allows production of high quality presentations using \LaTeX and pdf processing. Org-mode has special support for turning an Org-mode file or tree into a ‘**beamer**’ presentation.

When the \LaTeX class for the current buffer (as set with `#+LaTeX_CLASS: beamer`) or subtree (set with a `LaTeX_CLASS` property) is **beamer**, a special export mode will turn the file or tree into a beamer presentation. Any tree with not-too-deep level nesting should in principle be exportable as a beamer presentation. By default, the top-level entries (or the first level below the selected subtree heading) will be turned into frames, and the outline structure below this level will become itemize lists. You can also configure the variable `org-beamer-frame-level` to a different level—then the hierarchy above frames will produce the sectioning structure of the presentation.

A template for useful in-buffer settings or properties can be inserted into the buffer with `M-x org-insert-beamer-options-template`. Among other things, this will install a column view format which is very handy for editing special properties used by beamer.

You can influence the structure of the presentation using the following properties:

BEAMER_env

The environment that should be used to format this entry. Valid environments are defined in the constant `org-beamer-environments-default`, and you can define more in `org-beamer-environments-extra`. If this property is set, the entry will also get a `:B_environment:` tag to make this visible. This tag has no semantic meaning, it is only a visual aid.

BEAMER_envargs

The beamer-special arguments that should be used for the environment, like `[t]` or `[<+>]` of `<2-3>`. If the `BEAMER_col` property is also set, something like `C[t]` can be added here as well to set an options argument for the implied `columns` environment. `c[t]` or `c<2->` will set an options for the implied `column` environment.

BEAMER_col

The width of a column that should start with this entry. If this property is set, the entry will also get a `:BMCOL:` property to make this visible. Also this tag is only a visual aid. When this is a plain number, it will be interpreted as a fraction of `\textwidth`. Otherwise it will be assumed that you have specified the units, like ‘`3cm`’. The first such property in a frame will start a `columns` environment to surround the columns. This environment is closed when an

entry has a `BEAMER_col` property with value 0 or 1, or automatically at the end of the frame.

`BEAMER_extra`

Additional commands that should be inserted after the environment has been opened. For example, when creating a frame, this can be used to specify transitions.

Frames will automatically receive a `fragile` option if they contain source code that uses the verbatim environment. Special ‘beamer’ specific code can be inserted using `#+BEAMER:` and `#+BEGIN_beamer...#+end_beamer` constructs, similar to other export backends, but with the difference that `#+LaTeX:` stuff will be included in the presentation as well.

Outline nodes with `BEAMER_env` property value ‘note’ or ‘noteNH’ will be formatted as beamer notes, i.e. they will be wrapped into `\note{...}`. The former will include the heading as part of the note text, the latter will ignore the heading of that node. To simplify note generation, it is actually enough to mark the note with a *tag* (either `:B_note:` or `:B_noteNH:`) instead of creating the `BEAMER_env` property.

You can turn on a special minor mode `org-beamer-mode` for editing support with

```
#+STARTUP: beamer
```

`C-c C-b`

`org-beamer-select-environment`

In `org-beamer-mode`, this key offers fast selection of a beamer environment or the `BEAMER_col` property.

Column view provides a great way to set the environment of a node and other important parameters. Make sure you are using a COLUMN format that is geared toward this special purpose. The command `M-x org-insert-beamer-options-template` defines such a format.

Here is a simple example Org document that is intended for beamer export.

```
#+LaTeX_CLASS: beamer
#+TITLE: Example Presentation
#+AUTHOR: Carsten Dominik
#+LaTeX_CLASS_OPTIONS: [presentation]
#+BEAMER_FRAME_LEVEL: 2
#+BEAMER_HEADER_EXTRA: \usetheme{Madrid}\usecolortheme{default}
#+COLUMNS: %35ITEM %10BEAMER_env(Env) %10BEAMER_envargs(Args) %4BEAMER_col(Col) %8BEAMER_extra(Ex)

* This is the first structural section

** Frame 1 \\ with a subtitle
*** Thanks to Eric Fraga                                     :BMCOL:B_block:
    :PROPERTIES:
    :BEAMER_env: block
    :BEAMER_envargs: C[t]
    :BEAMER_col: 0.5
    :END:
    for the first viable beamer setup in Org
*** Thanks to everyone else                                   :BMCOL:B_block:
    :PROPERTIES:
    :BEAMER_col: 0.5
    :BEAMER_env: block
    :BEAMER_envargs: <2->
    :END:
```

```

    for contributing to the discussion
**** This will be formatted as a beamer note           :B_note:
** Frame 2 \\ where we will not use columns
*** Request                                           :B_block:
    Please test this stuff!
    :PROPERTIES:
    :BEAMER_env: block
    :END:

```

For more information, see the documentation on Worg.

12.7 DocBook export

Org contains a DocBook exporter written by Baoqiu Cui. Once an Org file is exported to DocBook format, it can be further processed to produce other formats, including PDF, HTML, man pages, etc., using many available DocBook tools and stylesheets.

Currently DocBook exporter only supports DocBook V5.0.

12.7.1 DocBook export commands

C-c C-e D **org-export-as-docbook**
 Export as DocBook file. For an Org file, ‘myfile.org’, the DocBook XML file will be ‘myfile.xml’. The file will be overwritten without warning. If there is an active region¹⁵, only the region will be exported. If the selected region is a single tree¹⁶, the tree head will become the document title. If the tree head entry has, or inherits, an `EXPORT_FILE_NAME` property, that name will be used for the export.

C-c C-e V **org-export-as-docbook-pdf-and-open**
 Export as DocBook file, process to PDF, then open the resulting PDF file.
 Note that, in order to produce PDF output based on exported DocBook file, you need to have XSLT processor and XSL-FO processor software installed on your system. Check variables `org-export-docbook-xslt-proc-command` and `org-export-docbook-xsl-fo-proc-command`.
 The stylesheet argument `%s` in variable `org-export-docbook-xslt-proc-command` is replaced by the value of variable `org-export-docbook-xslt-stylesheet`, which needs to be set by the user. You can also overrule this global setting on a per-file basis by adding an in-buffer setting `#+XSLT:` to the Org file.

C-c C-e v D
 Export only the visible part of the document.

12.7.2 Quoting DocBook code

You can quote DocBook code in Org files and copy it verbatim into exported DocBook file with the following constructs:

#+DOCBOOK: Literal DocBook code for export

or

¹⁵ This requires `transient-mark-mode` to be turned on

¹⁶ To select the current subtree, use `C-c @`.

```
#+BEGIN_DOCBOOK
All lines between these markers are exported by DocBook exporter
literally.
#+END_DOCBOOK
```

For example, you can use the following lines to include a DocBook warning admonition. As to what this warning says, you should pay attention to the document context when quoting DocBook code in Org files. You may make exported DocBook XML files invalid by not quoting DocBook code correctly.

```
#+BEGIN_DOCBOOK
<warning>
  <para>You should know what you are doing when quoting DocBook XML code
  in your Org file. Invalid DocBook XML may be generated by
  DocBook exporter if you are not careful!</para>
</warning>
#+END_DOCBOOK
```

12.7.3 Recursive sections

DocBook exporter exports Org files as articles using the `article` element in DocBook. Recursive sections, i.e. `section` elements, are used in exported articles. Top level headlines in Org files are exported as top level sections, and lower level headlines are exported as nested sections. The entire structure of Org files will be exported completely, no matter how many nested levels of headlines there are.

Using recursive sections makes it easy to port and reuse exported DocBook code in other DocBook document types like `book` or `set`.

12.7.4 Tables in DocBook export

Tables in Org files are exported as HTML tables, which have been supported since DocBook V4.3.

If a table does not have a caption, an informal table is generated using the `informaltable` element; otherwise, a formal table will be generated using the `table` element.

12.7.5 Images in DocBook export

Images that are linked to without a description part in the link, like ‘`[[file:img.jpg]]`’ or ‘`[[./img.jpg]]`’, will be exported to DocBook using `mediaobject` elements. Each `mediaobject` element contains an `imageobject` that wraps an `imagedata` element. If you have specified a caption for an image as described in Section 11.2 [Images and tables], page 118, a `caption` element will be added in `mediaobject`. If a label is also specified, it will be exported as an `xml:id` attribute of the `mediaobject` element.

Image attributes supported by the `imagedata` element, like `align` or `width`, can be specified in two ways: you can either customize variable `org-export-docbook-default-image-attributes` or use the `#+ATTR_DOCBOOK:` line. Attributes specified in variable `org-export-docbook-default-image-attributes` are applied to all inline images in the Org file to be exported (unless they are overridden by image attributes specified in `#+ATTR_DOCBOOK:` lines).

The `#+ATTR_DOCBOOK:` line can be used to specify additional image attributes or override default image attributes for individual images. If the same attribute appears in both the `#+ATTR_DOCBOOK:` line and variable `org-export-docbook-default-image-attributes`, the former takes precedence. Here is an example about how image attributes can be set:

```
#+CAPTION:      The logo of Org-mode
#+LABEL:         unicorn-svg
#+ATTR_DOCBOOK: scalefit="1" width="100%" depth="100%"
[[./img/org-mode-unicorn.svg]]
```

By default, DocBook exporter recognizes the following image file types: ‘jpeg’, ‘jpg’, ‘png’, ‘gif’, and ‘svg’. You can customize variable `org-export-docbook-inline-image-extensions` to add more types to this list as long as DocBook supports them.

12.7.6 DocBook 出力における特殊文字

Special characters that are written in TeX-like syntax, such as `\alpha`, `\Gamma`, and `\Zeta`, are supported by DocBook exporter. These characters are rewritten to XML entities, like `α`, `Γ`, and `Ζ`, based on the list saved in variable `org-entities`. As long as the generated DocBook file includes the corresponding entities, these special characters are recognized.

You can customize variable `org-export-docbook-doctype` to include the entities you need. For example, you can set variable `org-export-docbook-doctype` to the following value to recognize all special characters included in XHTML entities:

```
"<!DOCTYPE article [
<!ENTITY % xhtml1-symbol PUBLIC
\"-//W3C//ENTITIES Symbol for HTML//EN//XML\"
\"http://www.w3.org/2003/entities/2007/xhtml1-symbol.ent\"
>
%xhtml1-symbol;
]>
"
```

12.8 TaskJuggler export

TaskJuggler (<http://www.taskjuggler.org/>) is a project management tool. It provides an optimizing scheduler that computes your project time lines and resource assignments based on the project outline and the constraints that you have provided.

The TaskJuggler exporter is a bit different from other exporters, such as the HTML and LaTeX exporters for example, in that it does not export all the nodes of a document or strictly follow the order of the nodes in the document.

Instead the TaskJuggler exporter looks for a tree that defines the tasks and a optionally tree that defines the resources for this project. It then creates a TaskJuggler file based on these trees and the attributes defined in all the nodes.

12.8.1 TaskJuggler export commands

<code>C-c C-e j</code>	<code>org-export-as-taskjuggler</code>
Export as TaskJuggler file.	

*C-c C-e J***org-export-as-taskjuggler-and-open**

Export as TaskJuggler file and then open the file with TaskJugglerUI.

12.8.2 Tasks

Create your tasks as you usually do with Org-mode. Assign efforts to each task using properties (it's easiest to do this in the column view). You should end up with something similar to the example by Peter Jones in <http://www.contextualdevelopment.com/static/artifacts/articles/2008/project-planning/project-planning.org>. Now mark the top node of your tasks with a tag named `:taskjuggler_project:` (or whatever you customized `org-export-taskjuggler-project-tag` to). You are now ready to export the project plan with *C-c C-e J* which will export the project plan and open a gantt chart in TaskJugglerUI.

12.8.3 Resources

Next you can define resources and assign those to work on specific tasks. You can group your resources hierarchically. Tag the top node of the resources with `:taskjuggler_resource:` (or whatever you customized `org-export-taskjuggler-resource-tag` to). You can optionally assign an identifier (named `'resource_id'`) to the resources (using the standard Org properties commands, see Section 7.1 [Property syntax], page 58) or you can let the exporter generate identifiers automatically (the exporter picks the first word of the headline as the identifier as long as it is unique—see the documentation of `org-taskjuggler-get-unique-id`). Using that identifier you can then allocate resources to tasks. This is again done with the `'allocate'` property on the tasks. Do this in column view or when on the task type *C-c C-x p allocate RET <resource_id> RET*.

Once the allocations are done you can again export to TaskJuggler and check in the Resource Allocation Graph which person is working on what task at what time.

12.8.4 Export of properties

The exporter also takes TODO state information into consideration, i.e. if a task is marked as done it will have the corresponding attribute in TaskJuggler (`'complete 100'`). Also it will export any property on a task resource or resource node which is known to TaskJuggler, such as `'limits'`, `'vacation'`, `'shift'`, `'booking'`, `'efficiency'`, `'journalentry'`, `'rate'` for resources or `'account'`, `'start'`, `'note'`, `'duration'`, `'end'`, `'journalentry'`, `'milestone'`, `'reference'`, `'responsible'`, `'scheduling'`, etc for tasks.

12.8.5 Dependencies

The exporter will handle dependencies that are defined in the tasks either with the `'ORDERED'` attribute (see Section 5.2.7 [TODO dependencies], page 46), with the `'BLOCKER'` attribute (see `'org-depend.el'`) or alternatively with a `'depends'` attribute. Both the `'BLOCKER'` and the `'depends'` attribute can be either `'previous-sibling'` or a reference to an identifier (named `'task_id'`) which is defined for another task in the project. `'BLOCKER'` and the `'depends'` attribute can define multiple dependencies separated by either space or comma. You can also specify optional attributes on the dependency by simply appending it. The following examples should illustrate this:

```
* Preparation
:PROPERTIES:
```

```

      :task_id:  preparation
      :ORDERED:  t
      :END:
* Training material
  :PROPERTIES:
  :task_id:  training_material
  :ORDERED:  t
  :END:
** Markup Guidelines
  :PROPERTIES:
  :Effort:    2.0
  :END:
** Workflow Guidelines
  :PROPERTIES:
  :Effort:    2.0
  :END:
* Presentation
  :PROPERTIES:
  :Effort:    2.0
  :BLOCKER:  training_material { gapduration 1d } preparation
  :END:

```

12.8.6 Reports

TaskJuggler can produce many kinds of reports (e.g. gantt chart, resource allocation, etc). The user defines what kind of reports should be generated for a project in the TaskJuggler file. The exporter will automatically insert some default reports in the file. These defaults are defined in `org-export-taskjuggler-default-reports`. They can be modified using `customize` along with a number of other options. For a more complete list, see *M-x customize-group RET org-export-taskjuggler RET*.

For more information and examples see the Org-taskjuggler tutorial at <http://orgmode.org/worg/org-tutorials/org-taskjuggler.html>.

12.9 Freemind export

The Freemind exporter was written by Lennart Borgman.

`C-c C-e m` `org-export-as-freemind`
 Export as Freemind mind map ‘myfile.mm’.

12.10 XOXO export

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

`C-c C-e x` `org-export-as-xoxo`
 Export as XOXO file ‘myfile.html’.

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

12.11 iCalendar エクスポート

Some people use Org-mode for keeping track of projects, but still prefer a standard calendar application for anniversaries and appointments. In this case it can be useful to show deadlines and other time-stamped items in Org files in the calendar application. Org-mode can export calendar information in the standard iCalendar format. If you also want to have TODO entries included in the export, configure the variable `org-icalendar-include-todo`. Plain timestamps are exported as VEVENT, and TODO items as VTOD. It will also create events from deadlines that are in non-TODO items. Deadlines and scheduling dates in TODO items will be used to set the start and due dates for the TODO entry¹⁷. As categories, it will use the tags locally defined in the heading, and the file/tree category¹⁸. See the variable `org-icalendar-alarm-time` for a way to assign alarms to entries with a time.

The iCalendar standard requires each entry to have a globally unique identifier (UID). Org creates these identifiers during export. If you set the variable `org-icalendar-store-uid`, the UID will be stored in the `:ID:` property of the entry and re-used next time you report this entry. Since a single entry can give rise to multiple iCalendar entries (as a timestamp, a deadline, a scheduled item, and as a TODO item), Org adds prefixes to the UID, depending on what triggered the inclusion of the entry. In this way the UID remains unique, but a synchronization program can still figure out from which entry all the different instances originate.

C-c C-e i **org-export-icalendar-this-file**
Create iCalendar entries for the current file and store them in the same directory, using a file extension `‘.ics’`.

C-c C-e I **org-export-icalendar-all-agenda-files**
Like **C-c C-e i**, but do this for all files in `org-agenda-files`. For each of these files, a separate iCalendar file will be written.

C-c C-e c **org-export-icalendar-combine-agenda-files**
Create a single large iCalendar file from all files in `org-agenda-files` and write it to the file given by `org-combined-agenda-icalendar-file`.

The export will honor SUMMARY, DESCRIPTION and LOCATION¹⁹ properties if the selected entries have them. If not, the summary will be derived from the headline, and the description from the body (limited to `org-icalendar-include-body` characters).

How this calendar is best read and updated, depends on the application you are using. The FAQ covers this issue.

¹⁷ See the variables `org-icalendar-use-deadline` and `org-icalendar-use-scheduled`.

¹⁸ To add inherited tags or the TODO state, configure the variable `org-icalendar-categories`.

¹⁹ The LOCATION property can be inherited from higher in the hierarchy if you configure `org-use-property-inheritance` accordingly.

13 Publishing

Org includes a publishing management system that allows you to configure automatic HTML conversion of *projects* composed of interlinked org files. You can also configure Org to automatically upload your exported HTML pages and related attachments, such as images and source code files, to a web server.

You can also use Org to convert files into PDF, or even combine HTML and PDF conversion so that files are available in both formats on the server.

Publishing has been contributed to Org by David O’Toole.

13.1 Configuration

Publishing needs significant configuration to specify files, destination and many other properties of a project.

13.1.1 The variable `org-publish-project-alist`

Publishing is configured almost entirely through setting the value of one variable, called `org-publish-project-alist`. Each element of the list configures one project, and may be in one of the two following forms:

```
("project-name" :property value :property value ...)
    i.e. a well-formed property list with alternating keys and values
or
("project-name" :components ("project-name" "project-name" ...))
```

In both cases, projects are configured by specifying property values. A project defines the set of files that will be published, as well as the publishing configuration to use when publishing those files. When a project takes the second form listed above, the individual members of the `:components` property are taken to be sub-projects, which group together files requiring different publishing options. When you publish such a “meta-project”, all the components will also be published, in the sequence given.

13.1.2 Sources and destinations for files

Most properties are optional, but some should always be set. In particular, Org needs to know where to look for source files, and where to put published files.

<code>:base-directory</code>	Directory containing publishing source files
<code>:publishing-directory</code>	Directory where output files will be published. You can directly publish to a webserver using a file name syntax appropriate for the Emacs ‘ <code>tramp</code> ’ package. Or you can publish to a local directory and use external tools to upload your website (see Section 13.2 [Uploading files], page 148).
<code>:preparation-function</code>	Function or list of functions to be called before starting the publishing process, for example, to run <code>make</code> for updating files to be published. The project property list is scoped into this call as the variable <code>project-plist</code> .

:completion-function Function or list of functions called after finishing the publishing process, for example, to change permissions of the resulting files. The project property list is scoped into this call as the variable **project-plist**.

13.1.3 Selecting files

By default, all files with extension **‘.org’** in the base directory are considered part of the project. This can be modified by setting the properties

:base-extension Extension (without the dot!) of source files. This actually is a regular expression. Set this to the symbol **any** if you want to get all files in **:base-directory**, even without extension.

:exclude Regular expression to match file names that should not be published, even though they have been selected on the basis of their extension.

:include List of files to be included regardless of **:base-extension** and **:exclude**.

:recursive Non-nil means, check base-directory recursively for files to publish.

13.1.4 Publishing action

Publishing means that a file is copied to the destination directory and possibly transformed in the process. The default transformation is to export Org files as HTML files, and this is done by the function **org-publish-org-to-html** which calls the HTML exporter (see Section 12.5 [HTML export], page 128). But you also can publish your content as PDF files using **org-publish-org-to-pdf**, or as **ascii**, **latin1** or **utf8** encoded files using the corresponding functions. If you want to publish the Org file itself, but with *archived*, *commented*, and *tag-excluded* trees removed, use **org-publish-org-to-org** and set the parameters **:plain-source** and/or **:htmlized-source**. This will produce **‘file.org’** and **‘file.org.html’** in the publishing directory¹. Other files like images only need to be copied to the publishing destination; for this you may use **org-publish-attachment**. For non-Org files, you always need to specify the publishing function:

:publishing-function Function executing the publication of a file. This may also be a list of functions, which will all be called in turn.

:plain-source Non-nil means, publish plain source.

:htmlized-source Non-nil means, publish htmlized source.

The function must accept three arguments: a property list containing at least a **:publishing-directory** property, the name of the file to be published, and the path to the publishing directory of the output file. It should take the specified file, make the necessary transformation (if any) and place the result into the destination folder.

¹ **‘file-source.org’** and **‘file-source.org.html’** if source and publishing directories are equal. Note that with this kind of setup, you need to add **:exclude "-source\\.org"** to the project definition in **org-publish-project-alist** to prevent the published source files from being considered as new org files the next time the project is published.

13.1.5 Options for the HTML/L^AT_EX exporters

The property list can be used to set many export options for the HTML and L^AT_EX exporters. In most cases, these properties correspond to user variables in Org. The table below lists these properties along with the variable they belong to. See the documentation string for the respective variable for details.

:link-up	org-export-html-link-up
:link-home	org-export-html-link-home
:language	org-export-default-language
:customtime	org-display-custom-times
:headline-levels	org-export-headline-levels
:section-numbers	org-export-with-section-numbers
:section-number-format	org-export-section-number-format
:table-of-contents	org-export-with-toc
:preserve-breaks	org-export-preserve-breaks
:archived-trees	org-export-with-archived-trees
:emphasize	org-export-with-emphasize
:sub-superscript	org-export-with-sub-superscripts
:special-strings	org-export-with-special-strings
:footnotes	org-export-with-footnotes
:drawers	org-export-with-drawers
:tags	org-export-with-tags
:todo-keywords	org-export-with-todo-keywords
:priority	org-export-with-priority
:TeX-macros	org-export-with-TeX-macros
:LaTeX-fragments	org-export-with-LaTeX-fragments
:latex-listings	org-export-latex-listings
:skip-before-1st-heading	org-export-skip-text-before-1st-heading
:fixed-width	org-export-with-fixed-width
:timestamps	org-export-with-timestamps
:author	user-full-name
:email	user-mail-address : addr;addr;..
:author-info	org-export-author-info
:email-info	org-export-email-info
:creator-info	org-export-creator-info
:tables	org-export-with-tables
:table-auto-headline	org-export-highlight-first-table-line
:style-include-default	org-export-html-style-include-default
:style-include-scripts	org-export-html-style-include-scripts
:style	org-export-html-style
:style-extra	org-export-html-style-extra
:convert-org-links	org-export-html-link-org-files-as-html
:inline-images	org-export-html-inline-images
:html-extension	org-export-html-extension
:html-preamble	org-export-html-preamble
:html-postamble	org-export-html-postamble
:xml-declaration	org-export-html-xml-declaration

<code>:html-table-tag</code>	<code>org-export-html-table-tag</code>
<code>:expand-quoted-html</code>	<code>org-export-html-expand</code>
<code>:timestamp</code>	<code>org-export-html-with-timestamp</code>
<code>:publishing-directory</code>	<code>org-export-publishing-directory</code>
<code>:select-tags</code>	<code>org-export-select-tags</code>
<code>:exclude-tags</code>	<code>org-export-exclude-tags</code>
<code>:latex-image-options</code>	<code>org-export-latex-image-default-option</code>

Most of the `org-export-with-*` variables have the same effect in both HTML and \LaTeX exporters, except for `:TeX-macros` and `:LaTeX-fragments` options, respectively `nil` and `t` in the \LaTeX export. See `org-export-plist-vars` to check this list of options.

When a property is given a value in `org-publish-project-alist`, its setting overrides the value of the corresponding user variable (if any) during publishing. Options set within a file (see Section 12.2 [Export options], page 125), however, override everything.

13.1.6 Links between published files

To create a link from one Org file to another, you would use something like `'[[file:foo.org][The foo]]'` or simply `'file:foo.org.'` (see Chapter 4 [Hyperlinks], page 34). When published, this link becomes a link to `'foo.html'`. In this way, you can interlink the pages of your "org web" project and the links will work as expected when you publish them to HTML. If you also publish the Org source file and want to link to that, use an `http:` link instead of a `file:` link, because `file:` links are converted to link to the corresponding `'html'` file.

You may also link to related files, such as images. Provided you are careful with relative file names, and provided you have also configured Org to upload the related files, these links will work too. See Section 13.3.2 [Complex example], page 149, for an example of this usage.

Sometimes an Org file to be published may contain links that are only valid in your production environment, but not in the publishing location. In this case, use the property

`:link-validation-function` Function to validate links

to define a function for checking link validity. This function must accept two arguments, the file name and a directory relative to which the file name is interpreted in the production environment. If this function returns `nil`, then the HTML generator will only insert a description into the HTML file, but no link. One option for this function is `org-publish-validate-link` which checks if the given file is part of any project in `org-publish-project-alist`.

13.1.7 Generating a sitemap

The following properties may be used to control publishing of a map of files for a given project.

<code>:auto-sitemap</code>	When non- <code>nil</code> , publish a sitemap during <code>org-publish-current-project</code> or <code>org-publish-all</code> .
<code>:sitemap-filename</code>	Filename for output of sitemap. Defaults to <code>'sitemap.org'</code> (which becomes <code>'sitemap.html'</code>).

<code>:sitemap-title</code>	Title of sitemap page. Defaults to name of file.
<code>:sitemap-function</code>	Plug-in function to use for generation of the sitemap. Defaults to <code>org-publish-org-sitemap</code> , which generates a plain list of links to all files in the project.
<code>:sitemap-sort-folders</code>	Where folders should appear in the sitemap. Set this to <code>first</code> (default) or <code>last</code> to display folders first or last, respectively. Any other value will mix files and folders.
<code>:sitemap-sort-files</code>	How the files are sorted in the site map. Set this to <code>alphabetically</code> (default), <code>chronologically</code> or <code>anti-chronologically</code> . <code>chronologically</code> sorts the files with older date first while <code>anti-chronologically</code> sorts the files with newer date first. <code>alphabetically</code> sorts the files alphabetically. The date of a file is retrieved with <code>org-publish-find-date</code> .
<code>:sitemap-ignore-case</code>	Should sorting be case-sensitive? Default <code>nil</code> .
<code>:sitemap-file-entry-format</code>	With this option one can tell how a sitemap's entry is formatted in the sitemap. This is a format string with some escape sequences: <code>%t</code> stands for the title of the file, <code>%a</code> stands for the author of the file and <code>%d</code> stands for the date of the file. The date is retrieved with the <code>org-publish-find-date</code> function and formatted with <code>org-publish-sitemap-date-format</code> . Default <code>%t</code> .
<code>:sitemap-date-format</code>	Format string for the <code>format-time-string</code> function that tells how a sitemap entry's date is to be formatted. This property bypasses <code>org-publish-sitemap-date-format</code> which defaults to <code>%Y-%m-%d</code> .

13.1.8 Generating an index

Org-mode can generate an index across the files of a publishing project.

`:makeindex` When non-nil, generate an index in the file `'theindex.org'` and publish it as `'theindex.html'`.

The file will be created when first publishing a project with the `:makeindex` set. The file only contains a statement `#+include: "theindex.inc"`. You can then build around this include statement by adding a title, style information, etc.

13.2 Uploading files

For those people already utilizing third party sync tools such as `rsync` or `unison`, it might be preferable not to use the built in *remote* publishing facilities of Org-mode which rely

heavily on Tramp. Tramp, while very useful and powerful, tends not to be so efficient for multiple file transfer and has been known to cause problems under heavy usage.

Specialized synchronization utilities offer several advantages. In addition to timestamp comparison, they also do content and permissions/attribute checks. For this reason you might prefer to publish your web to a local directory (possibly even *in place* with your Org files) and then use ‘unison’ or ‘rsync’ to do the synchronization with the remote host.

Since Unison (for example) can be configured as to which files to transfer to a certain remote destination, it can greatly simplify the project publishing definition. Simply keep all files in the correct location, process your Org files with `org-publish` and let the synchronization tool do the rest. You do not need, in this scenario, to include attachments such as ‘jpg’, ‘css’ or ‘gif’ files in the project definition since the 3rd party tool syncs them.

Publishing to a local directory is also much faster than to a remote one, so that you can afford more easily to republish entire projects. If you set `org-publish-use-timestamps-flag` to `nil`, you gain the main benefit of re-including any changed external files such as source example files you might include with `#+INCLUDE`. The timestamp mechanism in Org is not smart enough to detect if included files have been modified.

13.3 Sample configuration

Below we provide two example configurations. The first one is a simple project publishing only a set of Org files. The second example is more complex, with a multi-component project.

13.3.1 Example: simple publishing configuration

This example publishes a set of Org files to the ‘public_html’ directory on the local machine.

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

13.3.2 Example: complex publishing configuration

This more complicated example publishes an entire website, including Org files converted to HTML, image files, Emacs Lisp source code, and style sheets. The publishing directory is remote and private files are excluded.

To ensure that links are preserved, care should be taken to replicate your directory structure on the web server, and to use relative file paths. For example, if your Org files are kept in ‘~/org’ and your publishable images in ‘~/images’, you would link to an image with

```
file:../images/myimage.png
```

On the web server, the relative path to the image should be the same. You can accomplish this by setting up an "images" folder in the right place on the web server, and publishing images to it.

```
(setq org-publish-project-alist
  '(("orgfiles"
    :base-directory "~/org/"
    :base-extension "org"
    :publishing-directory "/ssh:user@host:~/html/notebook/"
    :publishing-function org-publish-org-to-html
    :exclude "PrivatePage.org" ;; regexp
    :headline-levels 3
    :section-numbers nil
    :table-of-contents nil
    :style "<link rel=\"stylesheet\"
          href=\"../other/mystyle.css\" type=\"text/css\"/>"
    :html-preamble t)

    ("images"
     :base-directory "~/images/"
     :base-extension "jpg\\|gif\\|png"
     :publishing-directory "/ssh:user@host:~/html/images/"
     :publishing-function org-publish-attachment)

    ("other"
     :base-directory "~/other/"
     :base-extension "css\\|el"
     :publishing-directory "/ssh:user@host:~/html/other/"
     :publishing-function org-publish-attachment)

    ("website" :components ("orgfiles" "images" "other"))))
```

13.4 公開の開始

Once properly configured, Org can publish with the following commands:

<code>C-c C-e X</code>	<code>org-publish</code>
Prompt for a specific project and publish all files that belong to it.	
<code>C-c C-e P</code>	<code>org-publish-current-project</code>
Publish the project containing the current file.	
<code>C-c C-e F</code>	<code>org-publish-current-file</code>
Publish only the current file.	
<code>C-c C-e E</code>	<code>org-publish-all</code>
Publish every project.	

Org uses timestamps to track when a file has changed. The above functions normally only publish changed files. You can override this and force publishing of all files by giving a prefix argument to any of the commands above, or by customizing the variable `org-`

`publish-use-timestamps-flag`. This may be necessary in particular if files include other files via `#+SETUPFILE:` or `#+INCLUDE:.`

14 ソースコードとの連携

Source code can be included in Org-mode documents using a ‘src’ block, e.g.

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

Org-mode provides a number of features for working with live source code, including editing of code blocks in their native major-mode, evaluation of code blocks, converting code blocks into source files (known as *tangling* in literate programming), and exporting code blocks and their results in several formats. This functionality was contributed by Eric Schulte and Dan Davison, and was originally named Org-babel.

The following sections describe Org-mode’s code block handling facilities.

14.1 Structure of code blocks

The structure of code blocks is as follows:

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

Switches and header arguments are optional. Code can also be embedded in text inline using

```
src_<language>{<body>}
```

or

```
src_<language>[<header arguments>]{<body>}
```

<name> This name is associated with the code block. This is similar to the ‘#+tblname’ lines that can be used to name tables in Org-mode files. Referencing the name of a code block makes it possible to evaluate the block from other places in the file, other files, or from Org-mode table formulas (see Section 3.5 [The spreadsheet], page 24).

<language> The language of the code in the block.

<switches> Optional switches controlling exportation of the code block (see switches discussion in Section 11.3 [Literal examples], page 118)

<header arguments> Optional header arguments control many aspects of evaluation, export and tangling of code blocks. See the Section 14.8 [Header arguments], page 156 section. Header arguments can also be set on a per-buffer or per-subtree basis using properties.

<body> The source code.

14.2 Editing source code

Use `C-c` ' to edit the current code block. This brings up a language major-mode edit buffer containing the body of the code block. Saving this buffer will write the new contents back to the Org buffer. Use `C-c` ' again to exit.

The `org-src-mode` minor mode will be active in the edit buffer. The following variables can be used to configure the behavior of the edit buffer. See also the customization group `org-edit-structure` for further configuration options.

`org-src-lang-modes`

If an Emacs major-mode named `<lang>-mode` exists, where `<lang>` is the language named in the header line of the code block, then the edit buffer will be placed in that major-mode. This variable can be used to map arbitrary language names to existing major modes.

`org-src-window-setup`

Controls the way Emacs windows are rearranged when the edit buffer is created.

`org-src-preserve-indentation`

This variable is especially useful for tangling languages such as Python, in which whitespace indentation in the output is critical.

`org-src-ask-before-returning-to-edit-buffer`

By default, Org will ask before returning to an open edit buffer. Set this variable to nil to switch without asking.

To turn on native code fontification in the *Org* buffer, configure the variable `org-src-fontify-natively`.

14.3 Exporting code blocks

It is possible to export the *contents* of code blocks, the *results* of code block evaluation, *neither*, or *both*. For most languages, the default exports the contents of code blocks. However, for some languages (e.g. `ditaa`) the default exports the results of code block evaluation. For information on exporting code block bodies, see Section 11.3 [Literal examples], page 118.

The `:exports` header argument can be used to specify export behavior:

Header arguments:

`:exports code`

The default in most languages. The body of the code block is exported, as described in Section 11.3 [Literal examples], page 118.

`:exports results`

The code block will be evaluated and the results will be placed in the Org-mode buffer for export, either updating previous results of the code block located anywhere in the buffer or, if no previous results exist, placing the results immediately after the code block. The body of the code block will not be exported.

`:exports both`

Both the code block and its results will be exported.

`:exports none`

Neither the code block nor its results will be exported.

It is possible to inhibit the evaluation of code blocks during export. Setting the `org-export-babel-evaluate` variable to `nil` will ensure that no code blocks are evaluated as part of the export process. This can be useful in situations where potentially untrusted Org-mode files are exported in an automated fashion, for example when Org-mode is used as the markup language for a wiki.

14.4 Extracting source code

Creating pure source code files by extracting code from source blocks is referred to as “tangling”—a term adopted from the literate programming community. During “tangling” of code blocks their bodies are expanded using `org-babel-expand-src-block` which can expand both variable and “noweb” style references (see Section 14.10 [Noweb reference syntax], page 171).

Header arguments

`:tangle no`

The default. The code block is not included in the tangled output.

`:tangle yes`

Include the code block in the tangled output. The output file name is the name of the org file with the extension `‘.org’` replaced by the extension for the block language.

`:tangle filename`

Include the code block in the tangled output to file `‘filename’`.

Functions

`org-babel-tangle`

Tangle the current file. Bound to `C-c C-v t`.

`org-babel-tangle-file`

Choose a file to tangle. Bound to `C-c C-v f`.

Hooks

`org-babel-post-tangle-hook`

This hook is run from within code files tangled by `org-babel-tangle`. Example applications could include post-processing, compilation or evaluation of tangled code files.

14.5 Evaluating code blocks

Code blocks can be evaluated¹ and the results placed in the Org-mode buffer. By default, evaluation is only turned on for `emacs-lisp` code blocks, however support exists for evaluating blocks in many languages. See Section 14.7 [Languages], page 156 for a list of supported

¹ Whenever code is evaluated there is a potential for that code to do harm. Org-mode provides a number of safeguards to ensure that it only evaluates code with explicit confirmation from the user. For information on these safeguards (and on how to disable them) see Section 15.4 [Code evaluation security], page 175.

languages. See Section 14.1 [Structure of code blocks], page 152 for information on the syntax used to define a code block.

There are a number of ways to evaluate code blocks. The simplest is to press `C-c C-c` or `C-c C-v e` with the point on a code block². This will call the `org-babel-execute-src-block` function to evaluate the block and insert its results into the Org-mode buffer.

It is also possible to evaluate named code blocks from anywhere in an Org-mode buffer or an Org-mode table. `#+call` (or synonymously `#+function` or `#+lob`) lines can be used to remotely execute code blocks located in the current Org-mode buffer or in the “Library of Babel” (see Section 14.6 [Library of Babel], page 155). These lines use the following syntax.

```
#+call: <name>(<arguments>) <header arguments>
#+function: <name>(<arguments>) <header arguments>
#+lob: <name>(<arguments>) <header arguments>
```

<name> The name of the code block to be evaluated.

<arguments>

Arguments specified in this section will be passed to the code block. These arguments should relate to `:var` header arguments in the called code block expressed using standard function call syntax. For example if the original code block named `double` has the header argument `:var n=2`, then the call line passing the number four to that block would be written as `#+call: double(n=2)`.

<header arguments>

Header arguments can be placed after the function invocation. See Section 14.8 [Header arguments], page 156 for more information on header arguments.

All header arguments placed in the **<header arguments>** section described above will be applied to the evaluation of the `#+call:` line, however it is sometimes desirable to specify header arguments to be passed to the code block being evaluated.

This is possible through the use of the following optional extended syntax.

```
#+call: <name>[<block header arguments>](<arguments>) <header arguments>
```

Any header argument placed between the square brackets in the **<block header arguments>** section will be applied to the evaluation of the named code block. For more examples of passing header arguments to `#+call:` lines see [Header arguments in function calls], page 158.

14.6 Library of Babel

The “Library of Babel” is a library of code blocks that can be called from any Org-mode file. The library is housed in an Org-mode file located in the ‘`contrib`’ directory of Org-mode. Org-mode users can deposit functions they believe to be generally useful in the library.

Code blocks defined in the “Library of Babel” can be called remotely as if they were in the current Org-mode buffer (see Section 14.5 [Evaluating code blocks], page 154 for information on the syntax of remote code block evaluation).

² The `org-babel-no-eval-on-ctrl-c-ctrl-c` variable can be used to remove code evaluation from the `C-c C-c` key binding.

Code blocks located in any Org-mode file can be loaded into the “Library of Babel” with the `org-babel-load-ingest` function, bound to `C-c C-v i`.

14.7 Languages

Code blocks in the following languages are supported.

Language	Identifier	Language	Identifier
Asymptote	asymptote	Emacs Calc	calc
C	C	C++	C++
Clojure	clojure	CSS	css
ditaa	ditaa	Graphviz	dot
Emacs Lisp	emacs-lisp	gnuplot	gnuplot
Haskell	haskell	Javascript	js
LaTeX	latex	Ledger	ledger
Lisp	lisp	MATLAB	matlab
Mscgen	mscgen	Objective Caml	ocaml
Octave	octave	Org-mode	org
Oz	oz	Perl	perl
Plantuml	plantuml	Python	python
R	R	Ruby	ruby
Sass	sass	Scheme	scheme
GNU Screen	screen	shell	sh
SQL	sql	SQLite	sqlite

Language-specific documentation is available for some languages. If available, it can be found at <http://orgmode.org/worg/org-contrib/babel/languages>.

The `org-babel-load-languages` controls which languages are enabled for evaluation (by default only `emacs-lisp` is enabled). This variable can be set using the customization interface or by adding code like the following to your emacs configuration.

The following disables `emacs-lisp` evaluation and enables evaluation of R code blocks.

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((emacs-lisp . nil)
  (R . t)))
```

It is also possible to enable support for a language by loading the related elisp file with `require`.

The following adds support for evaluating `clojure` code blocks.

```
(require 'ob-clojure)
```

14.8 Header arguments

Code block functionality can be configured with header arguments. This section provides an overview of the use of header arguments, and then describes each header argument in detail.

14.8.1 Using header arguments

The values of header arguments can be set in six different ways, each more specific (and having higher priority) than the last.

System-wide header arguments

System-wide values of header arguments can be specified by customizing the `org-babel-default-header-args` variable:

```
:session    => "none"
:results    => "replace"
:exports    => "code"
:cache      => "no"
:noweb      => "no"
```

For example, the following example could be used to set the default value of `:noweb` header arguments to `yes`. This would have the effect of expanding `:noweb` references by default when evaluating source code blocks.

```
(setq org-babel-default-header-args
  (cons '(:noweb . "yes")
    (assq-delete-all :noweb org-babel-default-header-args)))
```

Language-specific header arguments

Each language can define its own set of default header arguments. See the language-specific documentation available online at <http://orgmode.org/worg/org-contrib/babel>.

Buffer-wide header arguments

Buffer-wide header arguments may be specified through the use of a special line placed anywhere in an Org-mode file. The line consists of the `#+BABEL:` keyword followed by a series of header arguments which may be specified using the standard header argument syntax.

For example the following would set `session` to `*R*`, and `results` to `silent` for every code block in the buffer, ensuring that all execution took place in the same session, and no results would be inserted into the buffer.

```
#+BABEL: :session *R* :results silent
```

Header arguments in Org-mode properties

Header arguments are also read from Org-mode properties (see Section 7.1 [Property syntax], page 58), which can be set on a buffer-wide or per-heading basis. An example of setting a header argument for all code blocks in a buffer is

```
#+property: tangle yes
```

When properties are used to set default header arguments, they are looked up with inheritance, so the value of the `:cache` header argument will default to `yes` in all code blocks in the subtree rooted at the following heading:

```
* outline header
:PROPERTIES:
:cache:      yes
```

:END:

Properties defined in this way override the properties set in `org-babel-default-header-args`. It is convenient to use the `org-set-property` function bound to `C-c C-x p` to set properties in Org-mode documents.

Code block specific header arguments

The most common way to assign values to header arguments is at the code block level. This can be done by listing a sequence of header arguments and their values as part of the `#+begin_src` line. Properties set in this way override both the values of `org-babel-default-header-args` and header arguments specified as properties. In the following example, the `:results` header argument is set to `silent`, meaning the results of execution will not be inserted in the buffer, and the `:exports` header argument is set to `code`, meaning only the body of the code block will be preserved on export to HTML or LaTeX.

```
#+source: factorial
#+begin_src haskell :results silent :exports code :var n=0
fac 0 = 1
fac n = n * fac (n-1)
#+end_src
```

Similarly, it is possible to set header arguments for inline code blocks:

```
src_haskell[:exports both]{fac 5}
```

Code block header arguments can span multiple lines using `==#+header:=` or `==#+headers:=` lines preceding a code block or nested in between the name and body of a named code block.

Multi-line header arguments on an un-named code block:

```
#+headers: :var data1=1
#+begin_src emacs-lisp :var data2=2
  (message "data1:%S, data2:%S" data1 data2)
#+end_src

#+results:
: data1:1, data2:2
```

Multi-line header arguments on a named code block:

```
#+source: named-block
#+header: :var data=2
#+begin_src emacs-lisp
  (message "data:%S" data)
#+end_src

#+results: named-block
: data:2
```

Header arguments in function calls

At the most specific level, header arguments for “Library of Babel” or function call lines can be set as shown in the two examples below. For more information on the structure of `#+call:` lines see Section 14.5 [Evaluating code blocks], page 154.

The following will apply the `:exports results` header argument to the evaluation of the `#+call:` line.

```
#+call: factorial(n=5) :exports results
```

The following will apply the `:session special` header argument to the evaluation of the `factorial` code block.

```
#+call: factorial[:session special](n=5)
```

14.8.2 Specific header arguments

The following header arguments are defined:

14.8.2.1 `:var`

The `:var` header argument is used to pass arguments to code blocks. The specifics of how arguments are included in a code block vary by language; these are addressed in the language-specific documentation. However, the syntax used to specify arguments is the same across all languages. The values passed to arguments can be literal values, values from org-mode tables and literal example blocks, the results of other code blocks, or Emacs Lisp code—see the “Emacs Lisp evaluation of variables” heading below.

These values can be indexed in a manner similar to arrays—see the “indexable variable values” heading below.

The following syntax is used to pass arguments to code blocks using the `:var` header argument.

```
:var name=assign
```

where `assign` can take one of the following forms

- literal value either a string `"string"` or a number 9.
- reference a table name:

```
#+tblname: example-table
| 1 |
| 2 |
| 3 |
| 4 |
```

```
#+source: table-length
#+begin_src emacs-lisp :var table=example-table
(length table)
#+end_src
```

```
#+results: table-length
: 4
```

a code block name, as assigned by `#+srcname:`, followed by parentheses:

```
#+begin_src emacs-lisp :var length=table-length()
(* 2 length)
#+end_src

#+results:
```



```
: 8
```

In addition, an argument can be passed to the code block referenced by `:var`. The argument is passed within the parentheses following the code block name:

```
#+source: double
#+begin_src emacs-lisp :var input=8
(* 2 input)
#+end_src

#+results: double
: 16

#+source: squared
#+begin_src emacs-lisp :var input=double(input=1)
(* input input)
#+end_src

#+results: squared
: 4
```

Alternate argument syntax

It is also possible to specify arguments in a potentially more natural way using the `#+source:` line of a code block. As in the following example arguments can be packed inside of parenthesis, separated by commas, following the source name.

```
#+source: double(input=0, x=2)
#+begin_src emacs-lisp
(* 2 (+ input x))
#+end_src
```

Indexable variable values

It is possible to reference portions of variable values by “indexing” into the variables. Indexes are 0 based with negative values counting back from the end. If an index is separated by `,s` then each subsequent section will index into the next deepest nesting or dimension of the value. Note that this indexing occurs *before* other table related header arguments like `:hlines`, `:colnames` and `:rownames` are applied. The following example assigns the last cell of the first row the table `example-table` to the variable `data`:

```
#+results: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |

#+begin_src emacs-lisp :var data=example-table[0,-1]
data
#+end_src

#+results:
```

```
: a
```

Ranges of variable values can be referenced using two integers separated by a `:`, in which case the entire inclusive range is referenced. For example the following assigns the middle three rows of `example-table` to `data`.

```
#+results: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |
| 5 | 3 |

#+begin_src emacs-lisp :var data=example-table[1:3]
data
#+end_src

#+results:
| 2 | b |
| 3 | c |
| 4 | d |
```

Additionally, an empty index, or the single character `*`, are both interpreted to mean the entire range and as such are equivalent to `0:-1`, as shown in the following example in which the entire first column is referenced.

```
#+results: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |

#+begin_src emacs-lisp :var data=example-table[,0]
data
#+end_src

#+results:
| 1 | 2 | 3 | 4 |
```

It is possible to index into the results of code blocks as well as tables. Any number of dimensions can be indexed. Dimensions are separated from one another by commas, as shown in the following example.

```
#+source: 3D
#+begin_src emacs-lisp
'(((1 2 3) (4 5 6) (7 8 9))
  ((10 11 12) (13 14 15) (16 17 18))
  ((19 20 21) (22 23 24) (25 26 27)))
#+end_src

#+begin_src emacs-lisp :var data=3D[1,,1]
```

```

    data
#+end_src

#+results:
| 11 | 14 | 17 |

```

Emacs Lisp evaluation of variables

Emacs lisp code can be used to initialize variable values. When a variable value starts with `(`, `[`, `'` or ``` it will be evaluated as Emacs Lisp and the result of the evaluation will be assigned as the variable value. The following example demonstrates use of this evaluation to reliably pass the file-name of the org-mode buffer to a code block—note that evaluation of header arguments is guaranteed to take place in the original org-mode file, while there is no such guarantee for evaluation of the code block body.

```

#+begin_src sh :var file-name=(buffer-file-name) :exports both
  wc -w $file
#+end_src

```

Note that values read from tables and lists will not be evaluated as Emacs Lisp, as shown in the following example.

```

#+results: table
| (a b c) |

#+headers: :var data=table[0,0]
#+begin_src perl
  $data
#+end_src

#+results:
: (a b c)

```

14.8.2.2 :results

There are three classes of `:results` header argument. Only one option per class may be supplied per code block.

- **collection** header arguments specify how the results should be collected from the code block
- **type** header arguments specify what type of result the code block will return—which has implications for how they will be inserted into the Org-mode buffer
- **handling** header arguments specify how the results of evaluating the code block should be handled.

Collection

The following options are mutually exclusive, and specify how the results should be collected from the code block.

- **value** This is the default. The result is the value of the last statement in the code block. This header argument places the evaluation in functional mode. Note that in

some languages, e.g., Python, use of this result type requires that a `return` statement be included in the body of the source code block. E.g., `:results value`.

- **output** The result is the collection of everything printed to STDOUT during the execution of the code block. This header argument places the evaluation in scripting mode. E.g., `:results output`.

Type

The following options are mutually exclusive and specify what type of results the code block will return. By default, results are inserted as either a table or scalar depending on their value.

- **table, vector** The results should be interpreted as an Org-mode table. If a single value is returned, it will be converted into a table with one row and one column. E.g., `:results value table`.
- **list** The results should be interpreted as an Org-mode list. If a single scalar value is returned it will be converted into a list with only one element.
- **scalar, verbatim** The results should be interpreted literally—they will not be converted into a table. The results will be inserted into the Org-mode buffer as quoted text. E.g., `:results value verbatim`.
- **file** The results will be interpreted as the path to a file, and will be inserted into the Org-mode buffer as a file link. E.g., `:results value file`.
- **raw, org** The results are interpreted as raw Org-mode code and are inserted directly into the buffer. If the results look like a table they will be aligned as such by Org-mode. E.g., `:results value raw`.
- **html** Results are assumed to be HTML and will be enclosed in a `begin_html` block. E.g., `:results value html`.
- **latex** Results assumed to be LaTeX and are enclosed in a `begin_latex` block. E.g., `:results value latex`.
- **code** Result are assumed to be parseable code and are enclosed in a code block. E.g., `:results value code`.
- **pp** The result is converted to pretty-printed code and is enclosed in a code block. This option currently supports Emacs Lisp, Python, and Ruby. E.g., `:results value pp`.
- **wrap** The result is wrapped in a `begin_result` block. This can be useful for inserting `raw` or `org` syntax results in such a way that their extent is known and they can be automatically removed or replaced.

Handling

The following results options indicate what happens with the results once they are collected.

- **silent** The results will be echoed in the minibuffer but will not be inserted into the Org-mode buffer. E.g., `:results output silent`.
- **replace** The default value. Any existing results will be removed, and the new results will be inserted into the Org-mode buffer in their place. E.g., `:results output replace`.

- **append** If there are pre-existing results of the code block then the new results will be appended to the existing results. Otherwise the new results will be inserted as with **replace**.
- **prepend** If there are pre-existing results of the code block then the new results will be prepended to the existing results. Otherwise the new results will be inserted as with **replace**.

14.8.2.3 :file

The header argument **:file** is used to specify an external file in which to save code block results. After code block evaluation an Org-mode style `[[file:]]` link (see Section 4.1 [Link format], page 34) to the file will be inserted into the Org-mode buffer. Some languages including R, gnuplot, dot, and ditaa provide special handling of the **:file** header argument automatically wrapping the code block body in the boilerplate code required to save output to the specified file. This is often useful for saving graphical output of a code block to the specified file.

The argument to **:file** should be either a string specifying the path to a file, or a list of two strings in which case the first element of the list should be the path to a file and the second a description for the link.

14.8.2.4 :dir and remote execution

While the **:file** header argument can be used to specify the path to the output file, **:dir** specifies the default directory during code block execution. If it is absent, then the directory associated with the current buffer is used. In other words, supplying **:dir path** temporarily has the same effect as changing the current directory with *M-x cd path*, and then not supplying **:dir**. Under the surface, **:dir** simply sets the value of the Emacs variable `default-directory`.

When using **:dir**, you should supply a relative path for file output (e.g. **:file myfile.jpg** or **:file results/myfile.jpg**) in which case that path will be interpreted relative to the default directory.

In other words, if you want your plot to go into a folder called ‘Work’ in your home directory, you could use

```
#+begin_src R :file myplot.png :dir ~/Work
matplot(matrix(rnorm(100), 10), type="l")
#+end_src
```

Remote execution

A directory on a remote machine can be specified using tramp file syntax, in which case the code will be evaluated on the remote machine. An example is

```
#+begin_src R :file plot.png :dir /dand@yakuba.princeton.edu:
plot(1:10, main=system("hostname", intern=TRUE))
#+end_src
```

Text results will be returned to the local Org-mode buffer as usual, and file output will be created on the remote machine with relative paths interpreted relative to the remote directory. An Org-mode link to the remote file will be created.

So, in the above example a plot will be created on the remote machine, and a link of the following form will be inserted in the org buffer:

```
[[file:/scp:dand@yakuba.princeton.edu:/home/dand/plot.png] [plot.png]]
```

Most of this functionality follows immediately from the fact that `:dir` sets the value of the Emacs variable `default-directory`, thanks to tramp. Those using XEmacs, or GNU Emacs prior to version 23 may need to install tramp separately in order for these features to work correctly.

Further points

- If `:dir` is used in conjunction with `:session`, although it will determine the starting directory for a new session as expected, no attempt is currently made to alter the directory associated with an existing session.
- `:dir` should typically not be used to create files during export with `:exports results` or `:exports both`. The reason is that, in order to retain portability of exported material between machines, during export links inserted into the buffer will **not** be expanded against `default-directory`. Therefore, if `default-directory` is altered using `:dir`, it is probable that the file will be created in a location to which the link does not point.

14.8.2.5 :exports

The `:exports` header argument specifies what should be included in HTML or LaTeX exports of the Org-mode file.

- `code` The default. The body of code is included into the exported file. E.g., `:exports code`.
- `results` The result of evaluating the code is included in the exported file. E.g., `:exports results`.
- `both` Both the code and results are included in the exported file. E.g., `:exports both`.
- `none` Nothing is included in the exported file. E.g., `:exports none`.

14.8.2.6 :tangle

The `:tangle` header argument specifies whether or not the code block should be included in tangled extraction of source code files.

- `抽出` The code block is exported to a source code file named after the basename (name w/o extension) of the Org-mode file. E.g., `:tangle yes`.
- `no` The default. The code block is not exported to a source code file. E.g., `:tangle no`.
- `other` Any other string passed to the `:tangle` header argument is interpreted as a file basename to which the block will be exported. E.g., `:tangle basename`.

14.8.2.7 :mkdirp

The `:mkdirp` header argument can be used to create parent directories of tangled files when missing. This can be set to `yes` to enable directory creation or to `no` to inhibit directory creation.

14.8.2.8 :comments

By default code blocks are tangled to source-code files without any insertion of comments beyond those which may already exist in the body of the code block. The `:comments` header argument can be set as follows to control the insertion of extra comments into the tangled code file.

- **no** The default. No extra comments are inserted during tangling.
- **link** The code block is wrapped in comments which contain pointers back to the original Org file from which the code was tangled.
- **yes** A synonym for “link” to maintain backwards compatibility.
- **org** Include text from the org-mode file as a comment.
The text is picked from the leading context of the tangled code and is limited by the nearest headline or source block as the case may be.
- **both** Turns on both the “link” and “org” comment options.
- **noweb** Turns on the “link” comment option, and additionally wraps expanded noweb references in the code block body in link comments.

14.8.2.9 :no-expand

By default, code blocks are expanded with `org-babel-expand-src-block` during tangling. This has the effect of assigning values to variables specified with `:var` (see Section 14.8.2.1 [var], page 159), and of replacing “noweb” references (see Section 14.10 [Noweb reference syntax], page 171) with their targets. The `:no-expand` header argument can be used to turn off this behavior.

14.8.2.10 :session

The `:session` header argument starts a session for an interpreted language where state is preserved.

By default, a session is not started.

A string passed to the `:session` header argument will give the session a name. This makes it possible to run concurrent sessions for each interpreted language.

14.8.2.11 :noweb

The `:noweb` header argument controls expansion of “noweb” style (see Section 14.10 [Noweb reference syntax], page 171) references in a code block. This header argument can have one of three values: **yes** **no** or **tangle**.

- **yes** All “noweb” syntax references in the body of the code block will be expanded before the block is evaluated, tangled or exported.
- **no** The default. No “noweb” syntax specific action is taken on evaluating code blocks, However, noweb references will still be expanded during tangling.
- 抽出 All “noweb” syntax references in the body of the code block will be expanded before the block is tangled, however “noweb” references will not be expanded when the block is evaluated or exported.

Noweb prefix lines

Noweb insertions are now placed behind the line prefix of the `<<reference>>`. This behavior is illustrated in the following example. Because the `<<example>>` noweb reference appears behind the SQL comment syntax, each line of the expanded noweb reference will be commented.

This code block:

```
-- <<example>>
```

expands to:

```
-- this is the
-- multi-line body of example
```

Note that noweb replacement text that does not contain any newlines will not be affected by this change, so it is still possible to use inline noweb references.

14.8.2.12 :cache

The `:cache` header argument controls the use of in-buffer caching of the results of evaluating code blocks. It can be used to avoid re-evaluating unchanged code blocks. This header argument can have one of two values: `yes` or `no`.

- `no` The default. No caching takes place, and the code block will be evaluated every time it is called.
- `yes` Every time the code block is run a SHA1 hash of the code and arguments passed to the block will be generated. This hash is packed into the `#+results:` line and will be checked on subsequent executions of the code block. If the code block has not changed since the last time it was evaluated, it will not be re-evaluated.

Code block caches notice if the value of a variable argument to the code block has changed. If this is the case, the cache is invalidated and the code block is re-run. In the following example, `caller` will not be re-run unless the results of `random` have changed since it was last run.

```
#+srcname: random
#+begin_src R :cache yes
runif(1)
#+end_src

#+results[a2a72cd647ad44515fab62e144796432793d68e1]: random
0.4659510825295

#+srcname: caller
#+begin_src emacs-lisp :var x=random :cache yes
x
#+end_src

#+results[bec9c8724e397d5df3b696502df3ed7892fc4f5f]: caller
0.254227238707244
```


14.8.2.13 :sep

The `:sep` header argument can be used to control the delimiter used when writing tabular results out to files external to Org-mode. This is used either when opening tabular results of a code block by calling the `org-open-at-point` function bound to `C-c C-o` on the code block, or when writing code block results to an external file (see Section 14.8.2.3 [file], page 164) header argument.

By default, when `:sep` is not specified output tables are tab delimited.

14.8.2.14 :hlines

Tables are frequently represented with one or more horizontal lines, or hlines. The `:hlines` argument to a code block accepts the values `yes` or `no`, with a default value of `no`.

- `no` Strips horizontal lines from the input table. In most languages this is the desired effect because an `hline` symbol is interpreted as an unbound variable and raises an error. Setting `:hlines no` or relying on the default value yields the following results.

```
#+tblname: many-cols
| a | b | c |
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |

#+source: echo-table
#+begin_src python :var tab=many-cols
    return tab
#+end_src

#+results: echo-table
| a | b | c |
| d | e | f |
| g | h | i |
```

- `yes` Leaves hlines in the table. Setting `:hlines yes` has this effect.

```
#+tblname: many-cols
| a | b | c |
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |

#+source: echo-table
#+begin_src python :var tab=many-cols :hlines yes
    return tab
#+end_src

#+results: echo-table
| a | b | c |
```

```
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |
```

14.8.2.15 :colnames

The `:colnames` header argument accepts the values `yes`, `no`, or `nil` for unassigned. The default value is `nil`.

- `nil` If an input table looks like it has column names (because its second row is an hline), then the column names will be removed from the table before processing, then reapplied to the results.

```
#+tblname: less-cols
| a |
|---|
| b |
| c |

#+srcname: echo-table-again
#+begin_src python :var tab=less-cols
    return [[val + '*' for val in row] for row in tab]
#+end_src

#+results: echo-table-again
| a |
|---|
| b* |
| c* |
```

Please note that column names are not removed before the table is indexed using variable indexing See Section 14.8.2.1 [var], page 159.

- `no` No column name pre-processing takes place
- `yes` Column names are removed and reapplied as with `nil` even if the table does not “look like” it has column names (i.e. the second row is not an hline)

14.8.2.16 :rownames

The `:rownames` header argument can take on the values `yes` or `no`, with a default value of `no`.

- `no` No row name pre-processing will take place.
- `yes` The first column of the table is removed from the table before processing, and is then reapplied to the results.

```
#+tblname: with-rownames
| one | 1 | 2 | 3 | 4 | 5 |
| two | 6 | 7 | 8 | 9 | 10 |

#+srcname: echo-table-once-again
#+begin_src python :var tab=with-rownames :rownames yes
```

```

    return [[val + 10 for val in row] for row in tab]
#+end_src

#+results: echo-table-once-again
| one | 11 | 12 | 13 | 14 | 15 |
| two | 16 | 17 | 18 | 19 | 20 |

```

Please note that row names are not removed before the table is indexed using variable indexing See Section 14.8.2.1 [var], page 159.

14.8.2.17 :shebang

Setting the `:shebang` header argument to a string value (e.g. `:shebang "#!/bin/bash"`) causes the string to be inserted as the first line of any tangled file holding the code block, and the file permissions of the tangled file are set to make it executable.

14.8.2.18 :eval

The `:eval` header argument can be used to limit the evaluation of specific code blocks. `:eval` accepts two arguments “never” and “query”. `:eval never` will ensure that a code block is never evaluated, this can be useful for protecting against the evaluation of dangerous code blocks. `:eval query` will require a query for every execution of a code block regardless of the value of the `org-confirm-babel-evaluate` variable.

14.9 Results of evaluation

The way in which results are handled depends on whether a session is invoked, as well as on whether `:results value` or `:results output` is used. The following table shows the table possibilities. For a full listing of the possible results header arguments see Section 14.8.2.2 [results], page 162.

	Non-session	Session
<code>:results value</code>	value of last expression	value of last expression
<code>:results output</code>	contents of STDOUT	concatenation of interpreter output

Note: With `:results value`, the result in both `:session` and non-session is returned to Org-mode as a table (a one- or two-dimensional vector of strings or numbers) when appropriate.

14.9.1 Non-session

14.9.1.1 :results value

This is the default. Internally, the value is obtained by wrapping the code in a function definition in the external language, and evaluating that function. Therefore, code should be written as if it were the body of such a function. In particular, note that Python does not automatically return a value from a function unless a `return` statement is present, and so a `return` statement will usually be required in Python.

This is the only one of the four evaluation contexts in which the code is automatically wrapped in a function definition.

14.9.1.2 :results output

The code is passed to the interpreter as an external process, and the contents of the standard output stream are returned as text. (In certain languages this also contains the error output stream; this is an area for future work.)

14.9.2 Session

14.9.2.1 :results value

The code is passed to the interpreter running as an interactive Emacs inferior process. The result returned is the result of the last evaluation performed by the interpreter. (This is obtained in a language-specific manner: the value of the variable `_` in Python and Ruby, and the value of `.Last.value` in R).

14.9.2.2 :results output

The code is passed to the interpreter running as an interactive Emacs inferior process. The result returned is the concatenation of the sequence of (text) output from the interactive interpreter. Notice that this is not necessarily the same as what would be sent to `STDOUT` if the same code were passed to a non-interactive interpreter running as an external process. For example, compare the following two blocks:

```
#+begin_src python :results output
print "hello"
2
print "bye"
#+end_src

#+resname:
: hello
: bye
```

In non-session mode, the ‘2’ is not printed and does not appear.

```
#+begin_src python :results output :session
print "hello"
2
print "bye"
#+end_src

#+resname:
: hello
: 2
: bye
```

But in `:session` mode, the interactive interpreter receives input ‘2’ and prints out its value, ‘2’. (Indeed, the other print statements are unnecessary here).

14.10 Noweb reference syntax

The “noweb” (see <http://www.cs.tufts.edu/~nr/noweb/>) Literate Programming system allows named blocks of code to be referenced by using the familiar Noweb syntax:

<<code-block-name>>

When a code block is tangled or evaluated, whether or not “noweb” references are expanded depends upon the value of the `:noweb` header argument. If `:noweb yes`, then a Noweb reference is expanded before evaluation. If `:noweb no`, the default, then the reference is not expanded before evaluation.

Note: the default value, `:noweb no`, was chosen to ensure that correct code is not broken in a language, such as Ruby, where <<arg>> is a syntactically valid construct. If <<arg>> is not syntactically valid in languages that you use, then please consider setting the default value.

14.11 Key bindings and useful functions

Many common Org-mode key sequences are re-bound depending on the context.

Within a code block, the following key bindings are active:

<code>C-c C-c</code>	<code>org-babel-execute-src-block</code>
<code>C-c C-o</code>	<code>org-babel-open-src-block-result</code>
<code>C-up</code>	<code>org-babel-load-in-session</code>
<code>M-down</code>	<code>org-babel-pop-to-session</code>

In an Org-mode buffer, the following key bindings are active:

<code>C-c C-v a</code> or <code>C-c C-v C-a</code>	<code>org-babel-sha1-hash</code>
<code>C-c C-v b</code> or <code>C-c C-v C-b</code>	<code>org-babel-execute-buffer</code>
<code>C-c C-v f</code> or <code>C-c C-v C-f</code>	<code>org-babel-tangle-file</code>
<code>C-c C-v g</code>	<code>org-babel-goto-named-source-block</code>
<code>C-c C-v h</code>	<code>org-babel-describe-bindings</code>
<code>C-c C-v l</code> or <code>C-c C-v C-l</code>	<code>org-babel-lob-ingest</code>
<code>C-c C-v p</code> or <code>C-c C-v C-p</code>	<code>org-babel-expand-src-block</code>
<code>C-c C-v s</code> or <code>C-c C-v C-s</code>	<code>org-babel-execute-subtree</code>
<code>C-c C-v t</code> or <code>C-c C-v C-t</code>	<code>org-babel-tangle</code>
<code>C-c C-v z</code> or <code>C-c C-v C-z</code>	<code>org-babel-switch-to-session</code>

14.12 バッチ処理

It is possible to call functions from the command line. This shell script calls `org-babel-tangle` on every one of its arguments.

Be sure to adjust the paths to fit your system.

```
#!/bin/sh
# -*- mode: shell-script -*-
#
# tangle files with org-mode
#
DIR=`pwd`
FILES=""
ORGINSTALL=~/.src/org/lisp/org-install.el"
```

```
# wrap each argument in the code required to call tangle on it
for i in $@; do
    FILES="$FILES \"$i\""
done

emacs -Q --batch -l $ORGINSTALL \
--eval "(progn
(add-to-list 'load-path (expand-file-name \"~/src/org/lisp/\"))
(add-to-list 'load-path (expand-file-name \"~/src/org/contrib/lisp/\"))
(require 'org)(require 'org-exp)(require 'ob)(require 'ob-tangle)
(mapc (lambda (file)
      (find-file (expand-file-name file \"$DIR\"))
      (org-babel-tangle)
      (kill-buffer)) '($FILES)))" 2>&1 |grep tangled
```

15 Miscellaneous

15.1 Completion

Emacs は補完無しでは Emacs とはいえませんが、そして org-mode はそれが意味をなすたびに使用します。もしあなたが *iswitchb* か *ido* のようなインタフェースを補完のプロンプトとして好むのであれば、あなたは `org-completion-use-iswitchb` や `org-completion-use-ido` 変数のいずれかを設定することで指定することができます。

Org-mode はバッファ中の補完をサポートします。この種類の補完はミニバッファを活用します。あなたは簡単に数文字をバッファに入力し、補完キーを補完するテキストの右側で押します。

M-TAB ポイント位置での補完

- 見出しの先頭では、TODO キーワードを補完します。
- ‘\’の後では、エクスポート機能によりサポートされる $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のシンボルを補完します。
- ‘*’の後では、‘`[*find this headline]`’のようにリンクを検索できるように、カレントバッファで見出しを補完します。
- 見出し中の ‘:’の後では、タグを補完します。タグのリストは `org-tag-alist` 変数 (see Section 6.2 [Setting tags], page 54 より、もしかすると、バッファ中の ‘`#+TAGS`’ オプションでも設定されているかもしれません。) から与えられるか、カレントバッファで使われている全てのタグから動的に生成されます。
- 見出しの外にある ‘:’の後では、プロパティキーを補完します。キーのリストは現在のバッファで使われている全てのキーから動的に構築されます。
- ‘[’の後では、リンクの省略記法を補完します (see Section 4.6 [Link abbreviations], page 39)。
- ‘+’の後では、Org-mode 向けのファイル固有の設定としてセットする ‘`TYP_TODO`’ や ‘`OPTIONS`’ のようなスペシャルキーワードを補完します。オプションキーワードが既に補完されているなら、**M-TAB** を再び押すことでこのキーワードの設定の例を挿入します。
- ‘`#+STARTUP:`’ の後の行の中では、`STARTUP` キーワードを補完します、すなわち、はこの行では正しいキーです。
- 他の場所では、IsPELL を用いた辞書補完が行われます。

15.2 Easy Templates

Org-mode は僅かなキーストロークのみによる空の構造の (`#+BEGIN_SRC` や `#+END_SRC` のような) 要素の挿入をサポートします。これはネイティブなテンプレート拡張機構を通じて得られるものです。ここで留意すべきこととして、Emacs は例えば ‘`yasnipet`’ のような同じように使うことができるいくつかの他のテンプレート機構を持ちます。

構造要素の挿入には、‘<’をタイプし、続いてテンプレートセレクトと **TAB** をタイプします。補完は上記のキーストロークが単独で行に入力されている場合のみ働きます。

以下のテンプレートセレクトが現在サポートされています。

```
s      #+begin_src ... #+end_src
e      #+begin_example ... #+end_example
```

```

q      +#+begin_quote ... +#+end_quote
v      +#+begin_verse ... +#+end_verse
c      +#+begin_center ... +#+end_center
l      +#+begin_latex ... +#+end_latex
L      +#+latex:
h      +#+begin_html ... +#+end_html
H      +#+html:
a      +#+begin_ascii ... +#+end_ascii
A      +#+ascii:
i      +#+include: line

```

例えば、空の行で `\<e\` と入力し、その後 TAB を入力すると、EXAMPLE テンプレートが補完されます。

あなたは `org-structure-template-list` 変数をカスタマイズすることで追加のテンプレートをインストールすることができます。詳細は変数の docstring を参照してください。

15.3 Speed keys

最初のアスタリスクの前のように、カーソルが見出しの先頭にある時、シングルキーはコマンドを実行できるようになっています。`org-use-speed-commands` 変数を設定することでこの機能を有効にします。あらかじめ定義されているコマンドのリストを挙げます。そして、`org-speed-commands-user` 変数にコマンドを追加することもできます。Speed keys は操作や他のコマンドを使うスピードを上げるだけではなく、TTY やキーボードに限界があるモバイル端末上で実行できない、または簡単に実行できないキーに割り当てられたコマンドを実行するための別の可能性を提供します。

コマンドが実行可能かどうかを見るには、機能を有効にして見出しの先頭にカーソルを置いて ? を押します。

15.4 コードの評価とセキュリティの問題

Org-mode は評価を含むコードのスニペットを使って作業をするためのツールを提供します。

あなたのマシン上でコードが動くことは常にセキュリティのリスクをもたらします。目的のため、またはアクシデントによって良くないコードや悪意のあるコードは実行されます。Org-mode はあなたが明確に実行の許可を与える場合のみそのようなコードを評価するデフォルトの設定を持っていて、そしてカジュアルなユーザに対してはこれらの機能は予防措置として保つべきです。

そのようなコードを通常用いる人々のために、確認用のプロンプトが表示され、そしてあなたはそれをオフにするかもしれません。これを行うことは可能ですが、あなたはリスクとかかわることを承知しなくてはなりません。

コードの評価は以下に挙げる状況を引き起こします:

ソースコードブロック

ソースコードブロックはエクスポート中かブロック中で `C-c C-c` を押した時に評価されます。ここで最も重要な事はコードスニペットを含む Org-mode のファイルがある意味で、実行可能なファイルに似ているということです。それで、あなたはそれらに対応し正しいソースのみを Emacs にロードすべきです— あなたがコンピュータ上にインストールしたプログラムのように。

デフォルトのセキュリティ装置を切る変数をカスタマイズする前にあなたがしていることを確かめてください。

org-confirm-babel-evaluate [User Option]

t(これがデフォルトです)の時、ユーザはコードブロックを評価する前に毎回確認されます。nilの時、ユーザは確認されません。関数をセットすると、それは2つの引数(言語とコードブロックの本体)を伴って呼ばれ、tを返せば尋ね、nilならば尋ねません。

例えば、これは\"ditaa\"コード(安全性は考慮されています)を確認無しで実行する方法です:

```
(defun my-org-confirm-babel-evaluate (lang body)
  (not (string= lang "ditaa"))) ; don't ask for ditaa
  (setq org-confirm-babel-evaluate 'my-org-confirm-babel-evaluate))
```

以下の shell と elisp はリンクしています。

Org-mode はコードを直接評価できる 2 つのリンクタイプ (see Section 4.3 [External links], page 35) を持っています。実行されるコードが見えないため、これらのリンクは問題があります。

org-confirm-shell-link-function [User Option]

シェルへのリンクを実行するための問い合わせを行う関数。

org-confirm-elisp-link-function [User Option]

Emacs Lisp へのリンクを実行するための問い合わせを行う関数。

表中の式 表中の式 (see Section 3.5 [The spreadsheet], page 24) は *calc* インタプリタでも *Emacs Lisp* インタプリタでも実行できるコードです。

15.5 Customization

Org-mode をカスタマイズするために使われる変数は 180 以上あります。マニュアルの圧縮のため、私はここで変数の説明はしません。変数のカスタマイズの構造化された概要は *M-x org-customize* で見ることができます。もしくは、Org->Customization から Browse Org Group を選択してください。多くの設定はバッファに特別な行を書くこと (see Section 15.6 [In-buffer settings], page 176) でそのファイル中で有効にすることができます。

15.6 バッファ中での設定の要約

Org-mode はファイル単位で設定を定義するために、バッファ内の特別な行を使用します。これらの行は '#+' に続くキーワードとコロン、そして設定を定義する語句でできています。いくつかの設定用語句は同じ行に書くことも、分けて書くこともできます。これらの設定はマニュアルを通じて説明されており、ここには要約を載せています。バッファ中の行を編集した後は、カーソル位置の変更をすぐに反映するために *C-c C-c* を押しってください。そうでなければ、新しい Emacs のセッションでファイルを再び開いた時のみ反映されます。

#+ARCHIVE: %s_done::

この行はこの行はアジェンダファイルのアーカイブ場所を設定します。次の '#+ARCHIVE' 行まで、もしくはファイルの末尾までの全ての行で適用されます。最初の行はまた、それより前の全てのエントリにも適用されます。関係する変数は *org-archive-location* です。

#+CATEGORY:

この行はアジェンダファイルのカテゴリを設定します。カテゴリは次の‘**#+CATEGORY**’行か、ファイル末尾までの全ての行で適用されます。また、それより前の全ての行にも適用されます。

#+COLUMNS: %25ITEM

カラムビューのデフォルトフォーマットを設定します。**COLUMNS**プロパティが適用されていない箇所でカラムビューが呼ばれた場合、このフォーマットが適用されます。

#+CONSTANTS: name1=value1 ...

テーブル式の中で使われる定数として、ファイルローカルな値を設定します。この行はローカル変数 **org-table-formula-constants-local** を設定します。グローバルバージョンは **org-table-formula-constants** です。

#+FILETAGS: :tag1:tag2:tag3:

ファイル中のエントリトップレベルエントリを含めたエントリが引き継ぐタグを設定します。

#+DRAWERS: NAME1

ファイルローカルな引き出しのセットを設定します。関係するグローバル変数が **org-drawers** です。

#+LINK: linkword replace

これらの行初はリンクの省略記法を指定します。See Section 4.6 [Link abbreviations], page 39. 関係する変数は **org-link-abbrev-alist** です。

#+PRIORITIES: highest lowest default

これらの行は優先順位の上限と初期値を設定します。3つ全てが、A から Z までの文字か、0 から 9 までの数字のどれかでなくてははいけません。最も高い優先順位は最も低い優先順位より低い ASCII の数値を持つ必要があります。

#+PROPERTY: Property_Name Value

この行は現在のバッファ中のエントリがデフォルトで引き継ぐ値を設定します。与えられたプロパティの値を指定するのに、最も便利です。

#+SETUPFILE: file

この行はバッファ内の設定を持つファイルを定義します。通常は、これは完全に無視されます。バッファのオプションの設定行をパースされた時のみ、バッファにそれらが含まれていればこのファイルのコンテンツはパースされます。特に、ファイルは内部設定の別の Org-mode ファイルとすることができます。カーソルをこの行に置き、**C-c** を押すことであなたはこのファイルを開くことができます。

#+STARTUP:

この行は Org-mode のファイルが開かれた時にに使われるオプションを設定します。最初のオプションセットはアウトラインツリーの初期表示を設定します。グローバルなデフォルトの設定に関係する変数は **org-startup-folded** で、初期値は **t** で、それはすなわち **overview** です。

overview	トップレベルの見出しのみ
content	全ての見出し
showall	全てのエントリを畳み込まない
showeverything	コンテンツの引き出しも表示する

動的な仮想インデントは変数 `org-startup-indent`¹ によって制御されます。

```
indent      start with org-indent-mode turned on
noindent    start with org-indent-mode turned off
```

それから、開いたファイルの表を整列させつオプションがあります。これはファイルがナローされた表を含む時に役に立ちます。関係する変数は `org-startup-align-all-tables` で、デフォルトでは `nil` です。

```
align       align all tables
noalign     don't align tables on startup
```

ファイルを開いたとき、インライン画像は自動的に表示されます。関係する変数は `org-startup-with-inline-images` で、ファイルを開いた際の遅延を避けるためにデフォルトでは `nil` になっています。

```
inlineimages show inline images
noinlineimages don't show inline images on startup
```

TODO アイテムの完了と再開のログを取ることとその感覚はこれらのオプションによって設定することが可能です (変数 `org-log-done` と `org-log-note-clock-out`、`org-log-repeat` を参照してください)。

```
logdone      アイテムに DONE マークがついたとき、タイムスタンプを記録します
lognotedone  ノートに DONE マークがついたとき、タイムスタンプを記録します

nologdone    アイテムに DONE マークがついたとき、タイムスタンプを記録しません
logrepeat    アイテムが再開されたとき、時刻を記録します
lognoterepeat アイテムが再開されたとき、ノートを記録します
nologrepeat  アイテムが再開されたとき、記録を行いません
lognoteclock-out 時間測定が終了したとき、ノートを記録します
nolognoteclock-out 時間測定が終了したとき、ノートを記録しません
logreschedule スケジューリングが変わったとき、タイムスタンプを記録します

lognotereschedule スケジューリングが変わったとき、ノートを記録します
nologreschedule スケジューリングが変わったとき、記録しません
logredeadline デッドラインが変更されたとき、タイムスタンプを記録します
lognoteredecline デッドラインが変更されたとき、ノートを記録します
nologredeadline デッドラインが変更されたとき、記録しません
logrefile     リファイル時にタイムスタンプを記録します
lognoterefile リファイル時にノートを記録します
nologrefile   リファイル時に記録しません
```

¹ Emacs22 と Org-mode6.29 が必要です

ここは、インデントされたアウトラインの見出しを隠すオプションです。関係する変数は `org-hide-leading-stars` と `org-odd-levels-only` で、デフォルトの設定は両方とも `nil` (`showstars` と `oddeven`) を意味します。

<code>hidestars</code>	全てのヘッドラインの「*」を見えなくします
<code>showstars</code>	全てのヘッドラインの「*」を見えるようにします
<code>indent</code>	virtual indentation according to outline level
<code>noindent</code>	アウトラインのレベルに一致する仮想インデントを行いません
<code>odd</code>	奇数のアウトラインレベル (1, 3, …) のみ
<code>oddeven</code>	全てのアウトラインレベル

タイムスタンプ (`org-put-time-stamp-overlays` と `org-time-stamp-overlay-formats` 変数) のカスタムフォーマットオーバーレイを切り替えます、

`customtime` カスタマイズされたタイムフォーマットで覆います

以下のオプションは表計算 (`constants-unit-system` 変数) に影響を与えます。

<code>constcgs</code>	'constants.el' should use the c-g-s unit system
<code>constSI</code>	'constants.el' should use the SI unit system

脚注の設定を行うには、以下のキーワードを使います。関係する変数は `org-footnote-define-inline` と `org-footnote-auto-label`、`org-footnote-auto-adjust` です。

<code>fninline</code>	脚注のインラインを定義します
<code>fnnoinline</code>	分割されたセクションでの脚注を定義します
<code>fnlocal</code>	最初の言及元の近くの脚注を定義しますが、インラインではありません
<code>fnprompt</code>	脚注ラベルのプロンプト
<code>fnauto</code>	[fn:1] のようなラベルを自動的に作成します (これがデフォルトです)
<code>fnconfirm</code>	編集と確認用の自動ラベルを用意します
<code>fnplain</code>	[1] のようなラベルを自動的に作成します
<code>fnadjust</code>	自動的に脚注の番号を振り直し、ソートします
<code>nofnadjust</code>	自動的に番号の振り直しとソートを行いません

開始時にブロックを見えなくするには、これらのキーワードを使います。関連する変数は `org-hide-block-startup` です。

<code>hideblocks</code>	開始時に全ての開始/終了ブロックを見えなくします
<code>nohideblocks</code>	開始時にブロックを見えなくしません

UTF-8 の文字の表示は変数 `org-pretty-entities` と以下のキーワードにより制御されます。

<code>entitiespretty</code>	可能な時、UTF-8 の文字を表示します
<code>entitiesplain</code>	空白にします

#+TAGS: *TAG1(c1) TAG2(c2)*

これらの行はファイル中の正しいタグと関連する *fast tag selection* キーを指定します。関連する変数は `org-tag-alist` です。

#+TBLFM: この行には行上にある表の数式が含まれます。

```
#+TITLE:, #+AUTHOR:, #+EMAIL:, #+LANGUAGE:, #+TEXT:, #+DATE:,
#+OPTIONS:, #+BIND:, #+XSLT:,
#+DESCRIPTION:, #+KEYWORDS:,
#+LATEX_HEADER:, #+STYLE:, #+LINK_UP:, #+LINK_HOME:,
#+EXPORT_SELECT_TAGS:, #+EXPORT_EXCLUDE_TAGS:
```

これらの行はエクスポートするファイルの設定を提供します。詳細については Section 12.2 [Export options], page 125 を参照してください。

```
#+TODO: #+SEQ_TODO: #+TYP_TODO:
```

これらの行は現在のファイルでの TODO キーワードとそれらの説明をセットします。関連する変数は `org-todo-keywords` です。

15.7 The very busy C-c C-c key

Org-mode では `C-c C-c` キーは多くの目的を持っていて、それはこのマニュアルの中のあちこちに分散されて書かれています。このキーの具体的な機能として、見出しにタグを追加するものがあります (see Chapter 6 [Tags], page 54)。他の多くの状況では、“ここを見て、見たものにに応じて更新する” というようなものを意味します。これは、異なる文脈でそれが何を意味するかの概要です。

- ツリーの抽出か時間表示からバッファ中のハイライトがあるなら、それらのハイライトを消去します。
- カーソルが特別な行である `#+KEYWORD` 行上にあるなら、このトリガはバッファをこの行でスキャンし情報を更新します。
- カーソルが表の中にあるなら、表を再調整します。もし自動テーブルエディタがオフになっていても、このコマンドは同じように働きます。
- カーソルが `#+TBLFM` 行にあるなら、全ての表に式を再適用します。
- カレントバッファがキャプチャバッファの場合、ノートを閉じファイルします。接頭辞引数を付けると相互作用せずにデフォルトの場所にファイルします。
- カーソルが `<<<target>>>` 上にあるなら、ラジオターゲットとバッファ中の関係するリンクを更新します。
- カーソルがプロパティ行かプロパティ引き出しの開始か終了行にあるなら、プロパティコマンドを提供します。
- カーソルが脚注参照上にあるなら、関係する定義部に行きます。反対も同様です。
- カーソルが統計データクッキー上にあるなら、それを更新します。
- カーソルがチェックボックス付きのプレーンリスト上にあるなら、チェックボックスのステータスをトグルします。
- カーソルが数字付きリスト上にあるなら、順序を整理しなおします。
- カーソルは。動的なブロックの `#+BEGIN` 行上にあるなら、ブロックを更新します。

15.8 より見やすいアウトラインビュー

多くの人々は Org-mode の見出しについて「*」の数が増えたときや見出しの下テキストがインデントされていないことを不快に感じます。アウトラインの見出しが本当のセクションの見出しの場所で本のように文書を書く時、これは問題ではありませんが、さらに正しい位置のリストアウトライン中では、インデント構造はより見やすくなります。

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

もしあなたが少なくとも Emacs23.2² と Org-mode のバージョン 6.29 を使っているのなら、この主のビューは `org-indent-mode` を使って表示する時間を動的に実現できます。このマイナーモードでは、全ての行は必要なスペース³ が前について表示されます。見出しはまた追加の「*」が前に置かれていて、それで 1 レベルにつき 2⁴ スペースシフトしてインデントします。全ての見出しの「*」の最後の 1 個だけは `org-hide` フェイス⁵ を使うことで見えなくなります- これについてのさらなる情報は「2.」を見てください。あなたは `org-indent-mode` を有効にするか `org-startup-indented` 変数で全てのファイルについて設定するか、ファイル毎に独立して設定することができます。

`#+STARTUP: indent`

もしあなたが Emacs/Org-mode の古いバージョンでも同じような効果を得たいのであれば、またはもしあなたがプレーンテキストの見た目が Emacs での表示と同様になるようにスペース文字でインデントしたいのであれば、Org-mode は以下の方法であなたをサポートします:

1. 見出しの下テキストのインデント

以下のように、あなたは各見出しの下テキストを見出しと同じ位置にインデントできます。

```
*** 3rd level
    more text, now indented
```

Org-mode は段落詰め、行の折り返し、構造の編集、適切なインデントの保存と適合と同時にこれをサポートします。

2. *Hiding leading stars*

You can modify the display in such a way that all leading stars become invisible. To do this in a global way, configure the variable `org-hide-leading-stars` or change this on a per-file basis with

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

「*」を隠した状態だと、ツリーはこうなります:

```
* Top level headline
  * Second level
  * 3rd level
  ...
```

先頭の「*」は本当に空白スペースに置き換えられたわけではなく、それらは文字色を背景色にする `org-hide` フェイスによって見えているだけです。もしあなたが白か黒の背景色を使っていないのであれば、あなたはこのフェイスを必要な効果が得られるようにカスタマイズする必要があります

² Emacs23.1 は `org-indent-mode` がクラッシュします

³ `org-indent-mode` は `visual-line-mode` (または純粋に `word-wrap` をセットします) が (見出しを含めた) 長い行を正しいインデントでラップするように `wrap-prefix` プロパティをセットします

⁴ `org-indent-indentation-per-level` 変数を参照してください

⁵ `org-indent-mode` を有効にすると、`org-hide-leading-stars` に `t` が、`org-adapt-indentation-hige` に `nil` がセットされます

あるでしょう。別の方法として、このフォントは余分な「*」が色を用いて、例えば、白い背景色の上に gray90を使うことで目に見えなくするというものがあります。

3.

あなたが全ての偶数レベルをスキップし、1、3、5といった奇数レベルのみを使い、効果的にある見出しレベルから次⁶に行くために2つの「*」を追加するのであれば、物事はより見やすくなります。この方法で、私達はこのセクションの冒頭で見られるアウトラインビューを得ます。構造の編集とこの慣例を正しく操作するエクスポートコマンドを作成すうために、`org-odd-levels-only`変数を設定するか、各ファイルに以下のような行を追加します。

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

あなたは `M-x org-convert-to-odd-levels RET`により Org-mode のファイルを 1 レベル 1 スターから 1 レベル 2 スターに変換することができます。逆の操作は、`M-x org-convert-to-oddeven-levels`です。

15.9 Org-mode を tty 端末で使う

Org-mode はとても多くのコマンドを用意しているため、デフォルトでは Org-mode のコアコマンドの多くは、例えばカーソルキー (`left`、`right`、`up`、`down`) や `TAB`、`RET`、とりわけ `Meta` や `Shift` といったモディファイヤキーと一緒に使われるキーなど、通常 tty 端末では扱えないキーにバインドされています。特別なキーが利用できない時に tty 端末上でこれらのコマンドにアクセスするには、以下の別バインディングを用いることができます。下記の tty 端末バインディングはおそらく扱いにくいでしょう；カスタマイズしたバインディングの方が以下のいくつかのものよりよいことに気づくかもしれません。

デフォルト	代替 1	スピー ドキー	代替 2
<code>S-TAB</code>	<code>C-u TAB</code>	<code>C</code>	
<code>M-left</code>	<code>C-c C-x l</code>	<code>l</code>	<code>Esc left</code>
<code>M-S-left</code>	<code>C-c C-x L</code>	<code>L</code>	
<code>M-right</code>	<code>C-c C-x r</code>	<code>r</code>	<code>Esc right</code>
<code>M-S-right</code>	<code>C-c C-x R</code>	<code>R</code>	
<code>M-up</code>	<code>C-c C-x u</code>		<code>Esc up</code>
<code>M-S-up</code>	<code>C-c C-x U</code>	<code>U</code>	
<code>M-down</code>	<code>C-c C-x d</code>		<code>Esc down</code>
<code>M-S-down</code>	<code>C-c C-x D</code>	<code>D</code>	
<code>S-RET</code>	<code>C-c C-x c</code>		
<code>M-RET</code>	<code>C-c C-x m</code>		<code>Esc RET</code>
<code>M-S-RET</code>	<code>C-c C-x M</code>		
<code>S-left</code>	<code>C-c left</code>		
<code>S-right</code>	<code>C-c right</code>		
<code>S-up</code>	<code>C-c up</code>		
<code>S-down</code>	<code>C-c down</code>		
<code>C-S-left</code>	<code>C-c C-x left</code>		
<code>C-S-right</code>	<code>C-c C-x right</code>		

⁶ あなたがプロパティの検索やリファイルの対象のためにレベルを指定する必要がある時、`'LEVEL=2'` は3つの「*」などにも対応します

15.10 他のパッケージとの関係

Org-mode は GNU Emacs の世界に生きていて、他のコードと様々な方法で連携します。

15.10.1 Packages that Org cooperates with

‘`calc.el`’ by Dave Gillespie

Org uses the Calc package for implementing spreadsheet functionality in its tables (see Section 3.5 [The spreadsheet], page 24). Org checks for the availability of Calc by looking for the function `calc-eval` which will have been autoloaded during setup if Calc has been installed properly. As of Emacs 22, Calc is part of the Emacs distribution. Another possibility for interaction between the two packages is using Calc for embedded calculations. See Section “Embedded Mode” in *GNU Emacs Calc Manual*.

‘`constants.el`’ by Carsten Dominik

In a table formula (see Section 3.5 [The spreadsheet], page 24), it is possible to use names for natural constants or units. Instead of defining your own constants in the variable `org-table-formula-constants`, install the ‘`constants`’ package which defines a large number of constants and units, and lets you use unit prefixes like ‘M’ for ‘Mega’, etc. You will need version 2.0 of this package, available at <http://www.astro.uva.nl/~dominik/Tools>. Org checks for the function `constants-get`, which has to be autoloaded in your setup. See the installation instructions in the file ‘`constants.el`’.

‘`cdlatex.el`’ by Carsten Dominik

Org-mode can make use of the CDLaTeX package to efficiently enter L^AT_EX fragments into Org files. See Section 11.7.5 [CDLaTeX mode], page 123.

‘`imenu.el`’ by Ake Stenhoff and Lars Lindberg

Imenu allows menu access to an index of items in a file. Org-mode supports Imenu—all you need to do to get the index is the following:

```
(add-hook 'org-mode-hook
  (lambda () (imenu-add-to-menubar "Imenu")))
```

By default the index is two levels deep—you can modify the depth using the option `org-imenu-depth`.

‘`remember.el`’ by John Wiegley

Org used to use this package for capture, but no longer does.

‘`speedbar.el`’ by Eric M. Ludlam

Speedbar is a package that creates a special frame displaying files and index items in files. Org-mode supports Speedbar and allows you to drill into Org files directly from the Speedbar. It also allows you to restrict the scope of agenda commands to a file or a subtree by using the command `<` in the Speedbar frame.

‘`table.el`’ by Takaaki Ota

Complex ASCII tables with automatic line wrapping, column- and row-spanning, and alignment can be created using the Emacs table package by Takaaki Ota (<http://sourceforge.net/projects/table>, and also part of

Emacs 22). Org-mode will recognize these tables and export them properly. Because of interference with other Org-mode functionality, you unfortunately cannot edit these tables directly in the buffer. Instead, you need to use the command `C-c '` to edit them, similar to source code snippets.

`C-c '` `org-edit-special`
Edit a `'table.el` table. Works when the cursor is in a `table.el` table.

`C-c ~` `org-table-create-with-table.el`
Insert a `'table.el` table. If there is already a table at point, this command converts it between the `'table.el` format and the Org-mode format. See the documentation string of the command `org-convert-table` for the restrictions under which this is possible.

`'table.el` is part of Emacs since Emacs 22.

`'footnote.el` by Steven L. Baur

Org-mode recognizes numerical footnotes as provided by this package. However, Org-mode also has its own footnote support (see Section 2.10 [Footnotes], page 16), which makes using `'footnote.el` unnecessary.

15.10.2 Packages that lead to conflicts with Org-mode

In Emacs 23, `shift-selection-mode` is on by default, meaning that cursor motions combined with the shift key should start or enlarge regions. This conflicts with the use of `S-cursor` commands in Org to change timestamps, TODO keywords, priorities, and item bullet types if the cursor is at such a location. By default, `S-cursor` commands outside special contexts don't do anything, but you can customize the variable `org-support-shift-select`. Org-mode then tries to accommodate shift selection by (i) using it outside of the special contexts where special commands apply, and by (ii) extending an existing active region even if the cursor moves across a special context.

`'CUA.el` by Kim. F. Storm

Key bindings in Org conflict with the `S-<cursor>` keys used by CUA mode (as well as `pc-select-mode` and `s-region-mode`) to select and extend the region. In fact, Emacs 23 has this built-in in the form of `shift-selection-mode`, see previous paragraph. If you are using Emacs 23, you probably don't want to use another package for this purpose. However, if you prefer to leave these keys to a different package while working in Org-mode, configure the variable `org-replace-disputed-keys`. When set, Org will move the following key bindings in Org files, and in the agenda buffer (but not during date selection).

<code>S-UP</code>	\Rightarrow	<code>M-p</code>	<code>S-DOWN</code>	\Rightarrow	<code>M-n</code>
<code>S-LEFT</code>	\Rightarrow	<code>M--</code>	<code>S-RIGHT</code>	\Rightarrow	<code>M++</code>
<code>C-S-LEFT</code>	\Rightarrow	<code>M-S--</code>	<code>C-S-RIGHT</code>	\Rightarrow	<code>M-S++</code>

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

‘yasnippet.el’

The way Org mode binds the TAB key (binding to [tab] instead of "\t") overrules YASnippet’s access to this key. The following code fixed this problem:

```
(add-hook 'org-mode-hook
  (lambda ()
    (org-set-local 'yas/trigger-key [tab])
    (define-key yas/keymap [tab] 'yas/next-field-group)))
```

The latest version of yasnippet doesn’t play well with Org mode. If the above code does not fix the conflict, start by defining the following function:

```
(defun yas/org-very-safe-expand ()
  (let ((yas/fallback-behavior 'return-nil)) (yas/expand)))
```

Then, tell Org mode what to do with the new function:

```
(add-hook 'org-mode-hook
  (lambda ()
    (make-variable-buffer-local 'yas/trigger-key)
    (setq yas/trigger-key [tab])
    (add-to-list 'org-tab-first-hook 'yas/org-very-safe-expand)
    (define-key yas/keymap [tab] 'yas/next-field)))
```

‘windmove.el’ by Hovav Shacham

This package also uses the *S-cursor* keys, so everything written in the paragraph above about CUA mode also applies here. If you want make the windmove function active in locations where Org-mode does not have special functionality on *S-cursor*, add this to your configuration:

```
;; Make windmove work in org-mode:
(add-hook 'org-shiftup-final-hook 'windmove-up)
(add-hook 'org-shiftright-final-hook 'windmove-right)
(add-hook 'org-shiftleft-final-hook 'windmove-left)
(add-hook 'org-shiftdown-final-hook 'windmove-down)
```

‘viper.el’ by Michael Kifer

Viper uses *C-c /* and therefore makes this key not access the corresponding Org-mode command *org-sparse-tree*. You need to find another key for this command, or override the key in *viper-vi-global-user-map* with

```
(define-key viper-vi-global-user-map "C-c /" 'org-sparse-tree)
```

Appendix A Hacking

This appendix covers some aspects where users can extend the functionality of Org.

A.1 Hooks

Org has a large number of hook variables that can be used to add functionality. This appendix about hacking is going to illustrate the use of some of them. A complete list of all hooks with documentation is maintained by the Worg project and can be found at <http://orgmode.org/worg/org-configs/org-hooks.php>.

A.2 Add-on packages

A large number of add-on packages have been written by various authors. These packages are not part of Emacs, but they are distributed as contributed packages with the separate release available at the Org-mode home page at <http://orgmode.org>. The list of contributed packages, along with documentation about each package, is maintained by the Worg project at <http://orgmode.org/worg/org-contrib/>.

A.3 Adding hyperlink types

Org has a large number of hyperlink types built-in (see Chapter 4 [Hyperlinks], page 34). If you would like to add new link types, Org provides an interface for doing so. Let's look at an example file, 'org-man.el', that will add support for creating links like '[`man:printf`] [The printf manpage]]' to show Unix manual pages inside Emacs:

```
;;; org-man.el - Support for links to manpages in Org

(require 'org)

(org-add-link-type "man" 'org-man-open)
(add-hook 'org-store-link-functions 'org-man-store-link)

(defcustom org-man-command 'man
  "The Emacs command to be used to display a man page."
  :group 'org-link
  :type '(choice (const man) (const woman)))

(defun org-man-open (path)
  "Visit the manpage on PATH.
PATH should be a topic that can be thrown at the man command."
  (funcall org-man-command path))

(defun org-man-store-link ()
  "Store a link to a manpage."
  (when (memq major-mode '(Man-mode woman-mode))
    ;; This is a man page, we do make this link
    (let* ((page (org-man-get-page-name))
           (link (concat "man:" page)))
```

```

      (description (format "Manpage for %s" page)))
    (org-store-link-props
     :type "man"
     :link link
     :description description))))

(defun org-man-get-page-name ()
  "Extract the page name from the buffer name."
  ;; This works for both `Man-mode' and `woman-mode'.
  (if (string-match "\\(\\S+\\)\\*" (buffer-name))
      (match-string 1 (buffer-name))
      (error "Cannot create link to this man page")))

(provide 'org-man)

;;; org-man.el ends here

```

You would activate this new link type in `.emacs` with

```
(require 'org-man)
```

Let's go through the file and see what it does.

1. It does `(require 'org)` to make sure that `org.el` has been loaded.
2. The next line calls `org-add-link-type` to define a new link type with prefix `'man'`. The call also contains the name of a function that will be called to follow such a link.
3. The next line adds a function to `org-store-link-functions`, in order to allow the command `C-c l` to record a useful link in a buffer displaying a man page.

The rest of the file defines the necessary variables and functions. First there is a customization variable that determines which Emacs command should be used to display man pages. There are two options, `man` and `woman`. Then the function to follow a link is defined. It gets the link path as an argument—in this case the link path is just a topic for the manual command. The function calls the value of `org-man-command` to display the man page.

Finally the function `org-man-store-link` is defined. When you try to store a link with `C-c l`, this function will be called to try to make a link. The function must first decide if it is supposed to create the link for this buffer type; we do this by checking the value of the variable `major-mode`. If not, the function must exit and return the value `nil`. If yes, the link is created by getting the manual topic from the buffer name and prefixing it with the string `'man:'`. Then it must call the command `org-store-link-props` and set the `:type` and `:link` properties. Optionally you can also set the `:description` property to provide a default for the link description when the link is later inserted into an Org buffer with `C-c C-l`.

When it makes sense for your new link type, you may also define a function `org-PREFIX-complete-link` that implements special (e.g. completion) support for inserting such a link with `C-c C-l`. Such a function should not accept any arguments, and return the full link with prefix.

A.4 Context-sensitive commands

Org has several commands that act differently depending on context. The most important example is the `C-c C-c` (see Section 15.7 [The very busy C-c C-c key], page 180). Also the `M-cursor` and `M-S-cursor` keys have this property.

Add-ons can tap into this functionality by providing a function that detects special context for that add-on and executes functionality appropriate for the context. Here is an example from Dan Davison's `'org-R.el'` which allows you to evaluate commands based on the 'R' programming language¹. For this package, special contexts are lines that start with `#+R:` or `#+RR:`.

```
(defun org-R-apply-maybe ()
  "Detect if this is context for org-R and execute R commands."
  (if (save-excursion
        (beginning-of-line 1)
        (looking-at "#\\+RR?:"))
      (progn (call-interactively 'org-R-apply)
              t) ;; to signal that we took action
      nil)) ;; to signal that we did not

(add-hook 'org-ctrl-c-ctrl-c-hook 'org-R-apply-maybe)
```

The function first checks if the cursor is in such a line. If that is the case, `org-R-apply` is called and the function returns `t` to signal that action was taken, and `C-c C-c` will stop looking for other contexts. If the function finds it should do nothing locally, it returns `nil` so that other, similar functions can have a try.

A.5 任意のシンタックスによる表やリスト

Since Orgtbl mode can be used as a minor mode in arbitrary buffers, a frequent feature request has been to make it work with native tables in specific languages, for example \LaTeX . However, this is extremely hard to do in a general way, would lead to a customization nightmare, and would take away much of the simplicity of the Orgtbl-mode table editor.

This appendix describes a different approach. We keep the Orgtbl mode table in its native format (the *source table*), and use a custom function to *translate* the table to the correct syntax, and to *install* it in the right location (the *target table*). This puts the burden of writing conversion functions on the user, but it allows for a very flexible system.

Bastien added the ability to do the same with lists, in Orgstruct mode. You can use Org's facilities to edit and structure lists by turning `orgstruct-mode` on, then locally exporting such lists in another format (HTML, \LaTeX or Texinfo.)

A.5.1 Radio tables

To define the location of the target table, you first need to create two lines that are comments in the current mode, but contain magic words for Orgtbl mode to find. Orgtbl mode will insert the translated table between these lines, replacing whatever was there before. For example:

¹ `'org-R.el'` has been replaced by the org-mode functionality described in Chapter 14 [Working With Source Code], page 152 and is now obsolete.

```
/* BEGIN RECEIVE ORGTBL table_name */
/* END RECEIVE ORGTBL table_name */
```

Just above the source table, we put a special line that tells Orgtbl mode how to translate this table and where to install it. For example:

```
#+ORGTBL: SEND table_name translation_function arguments....
```

`table_name` is the reference name for the table that is also used in the receiver lines. `translation_function` is the Lisp function that does the translation. Furthermore, the line can contain a list of arguments (alternating key and value) at the end. The arguments will be passed as a property list to the translation function for interpretation. A few standard parameters are already recognized and acted upon before the translation function is called:

`:skip N` Skip the first `N` lines of the table. Hlines do count as separate lines for this parameter!

`:skipcols (n1 n2 ...)`

List of columns that should be skipped. If the table has a column with calculation marks, that column is automatically discarded as well. Please note that the translator function sees the table *after* the removal of these columns, the function never knows that there have been additional columns.

The one problem remaining is how to keep the source table in the buffer without disturbing the normal workings of the file, for example during compilation of a C file or processing of a \LaTeX file. There are a number of different solutions:

- The table could be placed in a block comment if that is supported by the language. For example, in C mode you could wrap the table between ‘/*’ and ‘*/’ lines.
- Sometimes it is possible to put the table after some kind of *END* statement, for example ‘\bye’ in \TeX and ‘\end{document}’ in \LaTeX .
- You can just comment the table line-by-line whenever you want to process the file, and uncomment it whenever you need to edit the table. This only sounds tedious—the command `M-x orgtbl-toggle-comment` makes this comment-toggling very easy, in particular if you bind it to a key.

A.5.2 A \LaTeX example of radio tables

The best way to wrap the source table in \LaTeX is to use the `comment` environment provided by ‘comment.sty’. It has to be activated by placing `\usepackage{comment}` into the document header. Orgtbl mode can insert a radio table skeleton² with the command `M-x orgtbl-insert-radio-table`. You will be prompted for a table name, let’s say we use ‘salesfigures’. You will then get the following template:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| | |
```

² By default this works only for \LaTeX , HTML, and Texinfo. Configure the variable `orgtbl-radio-tables` to install templates for other modes.

```
\end{comment}
```

The `#+ORGTBL: SEND` line tells Orgtbl mode to use the function `orgtbl-to-latex` to convert the table into \LaTeX and to put it into the receiver location with name `salesfigures`. You may now fill in the table—feel free to use the spreadsheet features³:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| Month | Days | Nr sold | per day |
|-----+-----+-----+-----|
| Jan   | 23   | 55      | 2.4   |
| Feb   | 21   | 16      | 0.8   |
| March | 22   | 278     | 12.6  |
#+TBLFM: $4=$3/$2;%.1f
% $ (optional extra dollar to keep font-lock happy, see footnote)
\end{comment}
```

When you are done, press `C-c C-c` in the table to get the converted table inserted between the two marker lines.

Now let's assume you want to make the table header by hand, because you want to control how columns are aligned, etc. In this case we make sure that the table translator skips the first 2 lines of the source table, and tell the command to work as a *splice*, i.e. to not produce header and footer commands of the target table:

```
\begin{tabular}{lrrrr}
Month & \multicolumn{1}{c}{Days} & & Nr. sold & per day \\
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\end{tabular}
%
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex :splice t :skip 2
| Month | Days | Nr sold | per day |
|-----+-----+-----+-----|
| Jan   | 23   | 55      | 2.4   |
| Feb   | 21   | 16      | 0.8   |
| March | 22   | 278     | 12.6  |
#+TBLFM: $4=$3/$2;%.1f
\end{comment}
```

The \LaTeX translator function `orgtbl-to-latex` is already part of Orgtbl mode. It uses a `tabular` environment to typeset the table and marks horizontal lines with `\hline`. Furthermore, it interprets the following parameters (see also see Section A.5.3 [Translator functions], page 191):

³ If the `'#+TBLFM'` line contains an odd number of dollar characters, this may cause problems with font-lock in \LaTeX mode. As shown in the example you can fix this by adding an extra line inside the `comment` environment that is used to balance the dollar expressions. If you are using \LaTeX with the `font-latex` library, a much better solution is to add the `comment` environment to the variable `LaTeX-verbatim-environments`.

`:splice nil/t`

When set to `t`, return only table body lines, don't wrap them into a `tabular` environment. Default is `nil`.

`:fmt fmt` A format to be used to wrap each field, it should contain `%s` for the original field value. For example, to wrap each field value in dollars, you could use `:fmt "s"`. This may also be a property list with column numbers and formats. For example `:fmt (2 "s" 4 "%s\\%")`. A function of one argument can be used in place of the strings; the function must return a formatted string.

`:efmt efmt`

Use this format to print numbers with exponentials. The format should have `%s` twice for inserting mantissa and exponent, for example `"%s\\times10^{%s}"`. The default is `"%s\\,(%s)"`. This may also be a property list with column numbers and formats, for example `:efmt (2 "$s\\times10^{%s}" 4 "$s\\cdot10^{%s}")`. After `efmt` has been applied to a value, `fmt` will also be applied. Similar to `fmt`, functions of two arguments can be supplied instead of strings.

A.5.3 Translator functions

`Orgtbl` mode has several translator functions built-in: `orgtbl-to-csv` (comma-separated values), `orgtbl-to-tsv` (TAB-separated values) `orgtbl-to-latex`, `orgtbl-to-html`, and `orgtbl-to-texinfo`. Except for `orgtbl-to-html`⁴, these all use a generic translator, `orgtbl-to-generic`. For example, `orgtbl-to-latex` itself is a very short function that computes the column definitions for the `tabular` environment, defines a few field and line separators and then hands processing over to the generic translator. Here is the entire code:

```
(defun orgtbl-to-latex (table params)
  "Convert the Orgtbl mode TABLE to LaTeX."
  (let* ((alignment (mapconcat (lambda (x) (if x "r" "l"))
                                org-table-last-alignment ""))
        (params2
         (list
          :tstart (concat "\\begin{tabular}" alignment "{")
          :tend "\\end{tabular}"
          :lstart "" :lend " \\\\" :sep " & "
          :efmt "%s\\,(%s)" :hline "\\hline")))
        (orgtbl-to-generic table (org-combine-plists params2 params))))
```

As you can see, the properties passed into the function (variable `PARAMS`) are combined with the ones newly defined in the function (variable `PARAMS2`). The ones passed into the function (i.e. the ones set by the ‘`ORGTBL SEND`’ line) take precedence. So if you would like to use the \LaTeX translator, but wanted the line endings to be ‘`\\[2mm]`’ instead of the default ‘`\\`’, you could just overrule the default with

```
#+ORGTBL: SEND test orgtbl-to-latex :lend " \\\\[2mm]"
```

For a new language, you can either write your own converter function in analogy with the \LaTeX translator, or you can use the generic function directly. For example, if you have

⁴ The HTML translator uses the same code that produces tables during HTML export.

a language where a table is started with ‘!BTBL!’, ended with ‘!ETBL!’, and where table lines are started with ‘!BL!’, ended with ‘!EL!’, and where the field separator is a TAB, you could call the generic translator like this (on a single line!):

```
#+ORGTBL: SEND test orgtbl-to-generic :tstart "!BTBL!" :tend "!ETBL!"
                        :lstart "!BL! " :lend " !EL!" :sep "\t"
```

Please check the documentation string of the function `orgtbl-to-generic` for a full list of parameters understood by that function, and remember that you can pass each of them into `orgtbl-to-latex`, `orgtbl-to-texinfo`, and any other function using the generic function.

Of course you can also write a completely new function doing complicated things the generic translator cannot do. A translator function takes two arguments. The first argument is the table, a list of lines, each line either the symbol `hline` or a list of fields. The second argument is the property list containing all parameters specified in the ‘#+ORGTBL: SEND’ line. The function must return a single string containing the formatted table. If you write a generally useful translator, please post it on emacs-orgmode@gnu.org so that others can benefit from your work.

A.5.4 ラジオリスト

Sending and receiving radio lists works exactly the same way as sending and receiving radio tables (see Section A.5.1 [Radio tables], page 188). As for radio tables, you can insert radio list templates in HTML, L^AT_EX and Texinfo modes by calling `org-list-insert-radio-list`.

Here are the differences with radio tables:

- Orgstruct mode must be active.
- Use the `ORGLST` keyword instead of `ORGTBL`.
- The available translation functions for radio lists don’t take parameters.
- `C-c C-c` will work when pressed on the first item of the list.

Here is a L^AT_EX example. Let’s say that you have this in your L^AT_EX file:

```
% BEGIN RECEIVE ORGLST to-buy
% END RECEIVE ORGLST to-buy
\begin{comment}
#+ORGLST: SEND to-buy org-list-to-latex
- a new house
- a new computer
  + a new keyboard
  + a new mouse
- a new life
\end{comment}
```

Pressing ‘C-c C-c’ on `a new house` and will insert the converted L^AT_EX list between the two marker lines.

A.6 Dynamic blocks

Org documents can contain *dynamic blocks*. These are specially marked regions that are updated by some user-written function. A good example for such a block is the clock table inserted by the command `C-c C-x C-r` (see Section 8.4 [Clocking work time], page 73).

Dynamic blocks are enclosed by a BEGIN-END structure that assigns a name to the block and can also specify parameters for the function producing the content of the block.

```
#+BEGIN: myblock :parameter1 value1 :parameter2 value2 ...
```

```
#+END:
```

Dynamic blocks are updated with the following commands

```
C-c C-x C-u                                     org-dblock-update
          Update dynamic block at point.
```

```
C-u C-c C-x C-u
          Update all dynamic blocks in the current file.
```

Updating a dynamic block means to remove all the text between BEGIN and END, parse the BEGIN line for parameters and then call the specific writer function for this block to insert the new content. If you want to use the original content in the writer function, you can use the extra parameter `:content`.

For a block with name `myblock`, the writer function is `org-dblock-write:myblock` with as only parameter a property list with the parameters given in the begin line. Here is a trivial example of a block that keeps track of when the block update function was last run:

```
#+BEGIN: block-update-time :format "on %m/%d/%Y at %H:%M"
```

```
#+END:
```

The corresponding block writer function could look like this:

```
(defun org-dblock-write:block-update-time (params)
  (let ((fmt (or (plist-get params :format) "%d. %m. %Y")))
    (insert "Last block update at: "
            (format-time-string fmt (current-time)))))
```

If you want to make sure that all dynamic blocks are always up-to-date, you could add the function `org-update-all-dblocks` to a hook, for example `before-save-hook`. `org-update-all-dblocks` is written in a way such that it does nothing in buffers that are not in `org-mode`.

You can narrow the current buffer to the current dynamic block (like any other block) with `org-narrow-to-block`.

A.7 Special agenda views

Org provides a special hook that can be used to narrow down the selection made by these agenda views: `todo`, `alltodo`, `tags`, `tags-todo`, `tags-tree`. You may specify a function that is used at each match to verify if the match should indeed be part of the agenda view, and if not, how much should be skipped. You can specify a global condition that will be applied to all agenda views, this condition would be stored in the variable `org-agenda-skip-function-global`. More commonly, such a definition is applied only to specific custom searches, using `org-agenda-skip-function`.

Let's say you want to produce a list of projects that contain a `WAITING` tag anywhere in the project tree. Let's further assume that you have marked all tree headings that define a project with the `TODO` keyword `PROJECT`. In this case you would run a `TODO` search

for the keyword `PROJECT`, but skip the match unless there is a `WAITING` tag anywhere in the subtree belonging to the project line.

To achieve this, you must write a function that searches the subtree for the tag. If the tag is found, the function must return `nil` to indicate that this match should not be skipped. If there is no such tag, return the location of the end of the subtree, to indicate that search should continue from there.

```
(defun my-skip-unless-waiting ()
  "Skip trees that are not waiting"
  (let ((subtree-end (save-excursion (org-end-of-subtree t))))
    (if (re-search-forward ":waiting:" subtree-end t)
        nil ; tag found, do not skip
        subtree-end))) ; tag not found, continue after end of subtree
```

Now you may use this function in an agenda custom command, for example like this:

```
(org-add-agenda-custom-command
  '("b" todo "PROJECT"
    ((org-agenda-skip-function 'my-skip-unless-waiting)
     (org-agenda-overriding-header "Projects waiting for something: "))))
```

Note that this also binds `org-agenda-overriding-header` to get a meaningful header in the agenda view.

A general way to create custom searches is to base them on a search for entries with a certain level limit. If you want to study all entries with your custom search function, simply do a search for `'LEVEL>0'`⁵, and then use `org-agenda-skip-function` to select the entries you really want to have.

You may also put a Lisp form into `org-agenda-skip-function`. In particular, you may use the functions `org-agenda-skip-entry-if` and `org-agenda-skip-subtree-if` in this form, for example:

```
'(org-agenda-skip-entry-if 'scheduled)
  Skip current entry if it has been scheduled.

'(org-agenda-skip-entry-if 'notscheduled)
  Skip current entry if it has not been scheduled.

'(org-agenda-skip-entry-if 'deadline)
  Skip current entry if it has a deadline.

'(org-agenda-skip-entry-if 'scheduled 'deadline)
  Skip current entry if it has a deadline, or if it is scheduled.

'(org-agenda-skip-entry-if 'todo '("TODO" "WAITING"))
  Skip current entry if the TODO keyword is TODO or WAITING.

'(org-agenda-skip-entry-if 'todo 'done)
  Skip current entry if the TODO keyword marks a DONE state.

'(org-agenda-skip-entry-if 'timestamp)
  Skip current entry if it has any timestamp, may also be deadline or scheduled.
```

⁵ Note that, when using `org-odd-levels-only`, a level number corresponds to order in the hierarchy, not to the number of stars.

```
'(org-agenda-skip-entry 'regexp "regular expression")
    Skip current entry if the regular expression matches in the entry.

'(org-agenda-skip-entry 'notregexp "regular expression")
    Skip current entry unless the regular expression matches.

'(org-agenda-skip-subtree-if 'regexp "regular expression")
    Same as above, but check and skip the entire subtree.
```

Therefore we could also have written the search for WAITING projects like this, even without defining a special function:

```
(org-add-agenda-custom-command
  '("b" todo "PROJECT"
    ((org-agenda-skip-function '(org-agenda-skip-subtree-if
                                'regexp ":waiting:"))
      (org-agenda-overriding-header "Projects waiting for something: "))))
```

A.8 Extracting agenda information

Org provides commands to access agenda information for the command line in Emacs batch mode. This extracted information can be sent directly to a printer, or it can be read by a program that does further processing of the data. The first of these commands is the function `org-batch-agenda`, that produces an agenda view and sends it as ASCII text to STDOUT. The command takes a single string as parameter. If the string has length 1, it is used as a key to one of the commands you have configured in `org-agenda-custom-commands`, basically any key you can use after `C-c a`. For example, to directly print the current TODO list, you could use

```
emacs -batch -l ~/.emacs -eval '(org-batch-agenda "t")' | lpr
```

If the parameter is a string with 2 or more characters, it is used as a tags/TODO match string. For example, to print your local shopping list (all items with the tag ‘shop’, but excluding the tag ‘NewYork’), you could use

```
emacs -batch -l ~/.emacs \
  -eval '(org-batch-agenda "+shop-NewYork")' | lpr
```

You may also modify parameters on the fly like this:

```
emacs -batch -l ~/.emacs \
  -eval '(org-batch-agenda "a" \
    org-agenda-span month \
    org-agenda-include-diary nil \
    org-agenda-files (quote ("~/org/project.org")))' \
  | lpr
```

which will produce a 30-day agenda, fully restricted to the Org file ‘~/org/projects.org’, not even including the diary.

If you want to process the agenda data in more sophisticated ways, you can use the command `org-batch-agenda-csv` to get a comma-separated list of values for each agenda item. Each line in the output will contain a number of fields separated by commas. The fields in a line are:

category	The category of the item																				
head	The headline, without TODO keyword, TAGS and PRIORITY																				
type	The type of the agenda entry, can be <table> <tr><td>todo</td><td>selected in TODO match</td></tr> <tr><td>tagsmatch</td><td>selected in tags match</td></tr> <tr><td>diary</td><td>imported from diary</td></tr> <tr><td>deadline</td><td>a deadline</td></tr> <tr><td>scheduled</td><td>scheduled</td></tr> <tr><td>timestamp</td><td>appointment, selected by timestamp</td></tr> <tr><td>closed</td><td>entry was closed on date</td></tr> <tr><td>upcoming-deadline</td><td>warning about nearing deadline</td></tr> <tr><td>past-scheduled</td><td>forwarded scheduled item</td></tr> <tr><td>block</td><td>entry has date block including date</td></tr> </table>	todo	selected in TODO match	tagsmatch	selected in tags match	diary	imported from diary	deadline	a deadline	scheduled	scheduled	timestamp	appointment, selected by timestamp	closed	entry was closed on date	upcoming-deadline	warning about nearing deadline	past-scheduled	forwarded scheduled item	block	entry has date block including date
todo	selected in TODO match																				
tagsmatch	selected in tags match																				
diary	imported from diary																				
deadline	a deadline																				
scheduled	scheduled																				
timestamp	appointment, selected by timestamp																				
closed	entry was closed on date																				
upcoming-deadline	warning about nearing deadline																				
past-scheduled	forwarded scheduled item																				
block	entry has date block including date																				
todo	The TODO keyword, if any																				
tags	All tags including inherited ones, separated by colons																				
date	The relevant date, like 2007-2-14																				
time	The time, like 15:00-16:50																				
extra	String with extra planning info																				
priority-l	The priority letter if any was given																				
priority-n	The computed numerical priority																				

Time and date will only be given if a timestamp (or deadline/scheduled) led to the selection of the item.

A CSV list like this is very easy to use in a post-processing script. For example, here is a Perl program that gets the TODO list from Emacs/Org and prints all the items, preceded by a checkbox:

```
#!/usr/bin/perl

# define the Emacs command to run
$cmd = "emacs -batch -l ~/.emacs -eval '(org-batch-agenda-csv \"t\")'";

# run it and capture the output
$agenda = qx{$cmd 2>/dev/null};

# loop over all lines
foreach $line (split(/\n/, $agenda)) {
  # get the individual values
  ($category, $head, $type, $todo, $tags, $date, $time, $extra,
   $priority_l, $priority_n) = split(/,/, $line);
  # process and print
  print "[ ] $head\n";
}
```

A.9 Using the property API

Here is a description of the functions that can be used to work with properties.

- org-entry-properties** *&optional pom which* [Function]
 Get all properties of the entry at point-or-marker POM.
 This includes the TODO keyword, the tags, time strings for deadline, scheduled, and clocking, and any additional properties defined in the entry. The return value is an alist. Keys may occur multiple times if the property key was used several times.
 POM may also be nil, in which case the current entry is used. If WHICH is nil or 'all', get all properties. If WHICH is 'special' or 'standard', only get that subclass.
- org-entry-get** *pom property &optional inherit* [Function]
 Get value of PROPERTY for entry at point-or-marker POM. By default, this only looks at properties defined locally in the entry. If INHERIT is non-nil and the entry does not have the property, then also check higher levels of the hierarchy. If INHERIT is the symbol **selective**, use inheritance if and only if the setting of **org-use-property-inheritance** selects PROPERTY for inheritance.
- org-entry-delete** *pom property* [Function]
 Delete the property PROPERTY from entry at point-or-marker POM.
- org-entry-put** *pom property value* [Function]
 Set PROPERTY to VALUE for entry at point-or-marker POM.
- org-buffer-property-keys** *&optional include-specials* [Function]
 Get all property keys in the current buffer.
- org-insert-property-drawer** [Function]
 Insert a property drawer at point.
- org-entry-put-multivalued-property** *pom property &rest values* [Function]
 Set PROPERTY at point-or-marker POM to VALUES. VALUES should be a list of strings. They will be concatenated, with spaces as separators.
- org-entry-get-multivalued-property** *pom property* [Function]
 Treat the value of the property PROPERTY as a whitespace-separated list of values and return the values as a list of strings.
- org-entry-add-to-multivalued-property** *pom property value* [Function]
 Treat the value of the property PROPERTY as a whitespace-separated list of values and make sure that VALUE is in this list.
- org-entry-remove-from-multivalued-property** *pom property value* [Function]
 Treat the value of the property PROPERTY as a whitespace-separated list of values and make sure that VALUE is *not* in this list.
- org-entry-member-in-multivalued-property** *pom property value* [Function]
 Treat the value of the property PROPERTY as a whitespace-separated list of values and check if VALUE is in this list.
- org-property-allowed-value-functions** [User Option]
 Hook for functions supplying allowed values for a specific property. The functions must take a single argument, the name of the property, and return a flat list of allowed values. If ':ETC' is one of the values, use the values as completion help, but allow also other values to be entered. The functions must return nil if they are not responsible for this property.

A.10 マッピング API を使う

Org has sophisticated mapping capabilities to find all entries satisfying certain criteria. Internally, this functionality is used to produce agenda views, but there is also an API that can be used to execute arbitrary functions for each or selected entries. The main entry point for this API is:

org-map-entries *func* **&optional** *match scope &rest skip* [Function]
 Call FUNC at each headline selected by MATCH in SCOPE.

FUNC is a function or a Lisp form. The function will be called without arguments, with the cursor positioned at the beginning of the headline. The return values of all calls to the function will be collected and returned as a list.

The call to FUNC will be wrapped into a save-excursion form, so FUNC does not need to preserve point. After evaluation, the cursor will be moved to the end of the line (presumably of the headline of the processed entry) and search continues from there. Under some circumstances, this may not produce the wanted results. For example, if you have removed (e.g. archived) the current (sub)tree it could mean that the next entry will be skipped entirely. In such cases, you can specify the position from where search should continue by making FUNC set the variable ‘org-map-continue-from’ to the desired buffer position.

MATCH is a tags/property/todo match as it is used in the agenda match view. Only headlines that are matched by this query will be considered during the iteration. When MATCH is nil or t, all headlines will be visited by the iteration.

SCOPE determines the scope of this command. It can be any of:

nil	the current buffer, respecting the restriction if any
tree	the subtree started with the entry at point
file	the current buffer, without restriction
file-with-archives	the current buffer, and any archives associated with it
agenda	all agenda files
agenda-with-archives	all agenda files with any archive files associated with them
(file1 file2 ...)	if this is a list, all files in the list will be scanned

The remaining args are treated as settings for the skipping facilities of the scanner. The following items can be given here:

archive	skip trees with the archive tag
comment	skip trees with the COMMENT keyword
function or Lisp form	will be used as value for org-agenda-skip-function , so whenever the function returns t, FUNC will not be called for that entry and search will continue from the point where the function leaves it

The function given to that mapping routine can really do anything you like. It can use the property API (see Section A.9 [Using the property API], page 196) to gather more

information about the entry, or in order to change metadata in the entry. Here are a couple of functions that might be handy:

org-todo &optional *arg* [Function]
 Change the TODO state of the entry. See the docstring of the functions for the many possible values for the argument ARG.

org-priority &optional *action* [Function]
 Change the priority of the entry. See the docstring of this function for the possible values for ACTION.

org-toggle-tag *tag* &optional *onoff* [Function]
 Toggle the tag TAG in the current entry. Setting ONOFF to either **on** or **off** will not toggle tag, but ensure that it is either on or off.

org-promote [Function]
 Promote the current entry.

org-demote [Function]
 Demote the current entry.

Here is a simple example that will turn all entries in the current file with a tag **TOMORROW** into **TODO** entries with the keyword **UPCOMING**. Entries in comment trees and in archive trees will be ignored.

```
(org-map-entries
 '(org-todo "UPCOMING")
 "+TOMORROW" 'file 'archive 'comment)
```

The following example counts the number of entries with **TODO** keyword **WAITING**, in all agenda files.

```
(length (org-map-entries t "/+WAITING" 'agenda))
```


Appendix B MobileOrg

MobileOrg (<http://mobileorg.ncogni.to/>) は Richard Moreland よって開発された *iPhone/iPod Touch* シリーズの携帯端末のためのアプリケーションです。*MobileOrg* は「リアル」のコンピュータ上にある Org-mode システムのために、オフラインのビューとキャプチャーによるサポートを提供します。その機能によって、実際のエントリーがどのように変化したかについて記録することができます。Android のユーザーは Matt Jones よって作成された MobileOrg Android (<http://wiki.github.com/matburt/mobileorg-android/>) のアプリをチェックしてください。

この付録では、*MobileOrg* で表示されるフォーマットの中でアジェンダビューを作成し、キャプチャーされたノートと *MobileOrg* で変更を、メインのシステムに統合していくために、Org-mode のサポートについて説明します。

MobileOrg の中でタグや TODO の状態を変更するためには、あなたは、例えば、ひとつひとつのファイルが、一部しか使っていないとしても、全ての重要なタグや TODO キーワードを網羅するように `org-todo-keywords` と `org-tags-alist` 変数のカスタマイズを設定しなければなりません。MobileOrg は、同様にインバッファの設定で状態やタグを提供しますが、これらの変数の中で設定されているものについてのみ、TODO の状態についての設定 (see Section 5.2.5 [Per-file keywords], page 45) や相互に排他的な タグ (see Section 6.2 [Setting tags], page 54) についての装備状況を理解してください。

B.1 Setting up the staging area

MobileOrg はサーバー上のディレクトリを通して、Emacs と相互に連携させる必要があります。もしも公開のサーバーを使用しているなら、そのサーバーにアップロードされるファイルを暗号化したいと考えるかもしれません。この機能は Org-mode 7.02 の *MobileOrg 1.5* (iPhone バージョン) で実現していますが、あなたのシステムに `openssl` をインストールしておく必要があるでしょう。暗号化するために、*MobileOrg* にパスワードを設定し、Emacs 上では、`org-mobile-use-encryption`¹ 変数を設定しておく必要があります。

無料の Dropbox.com (<http://dropbox.com>) のアカウント² を使い、ディレクトリを作成するのが最も簡単な方法です。MobileOrg で最初に Dropbox に接続したときに Dropbox の中に MobileOrg のディレクトリが作成されます。そのディレクトリが作成されたあと、次のように Emacs に書き込みます。

```
(setq org-mobile-directory "~/Dropbox/MobileOrg")
```

Org-mode はそのディレクトリの中に、*MobileOrg* 用のファイルを置いたり、そこからキャプチャーされたノートを読み込んだりするコマンドを持っています。

B.2 Pushing to MobileOrg

この操作では、`org-mobile-files` の中にリストアップされている全てのファイルを、`org-mobile-directory` で指定したディレクトリにコピーします。デフォルトではこのリストにはすべてのアジェン

¹ もしもあなたの Emacs の設定ファイルの中にパスワードを安全に保存したいならば、`org-mobile-encryption-password` 変数を設定すると良いでしょう。その変数の説明文を読んでください。暗号化は、`.org` ファイルの内容のみに適用されることに注意してください。ファイルの名称そのものは、そのまま表示されます。

² もしも Dropbox を利用できない場合、または MobileOrg のバージョンがそれをサポートしていない場合には、webdav サーバが利用できます。詳しい情報を得るには、MobileOrg の説明部と FAQ entry (<http://orgmode.org/worg/org-faq.html#mobileorg-webdav>) をチェックしてください。

ダファイル (org-agenda-filesに登録されている) を含んでいます。しかしながら、org-mobiles-filesをカスタマイズすることでファイルを追加できます。ファイル名は、org-directoryとの相対パスで登録されるので、すべてのファイルがこのディレクトリの中に入ることになります。プッシュする操作で、ユーザー³によって定義されたすべてのカスタマイズされたアジェンダビューを持った‘agendas.org’という特別な Org-mode ファイルを作成します。最後に、Org-mode は全ての他のファイルへのリンクを含んだ‘index.org’というファイルを書き込みます。MobileOrg は、最初サーバーからこのファイルを読み込み、それから、そこに置かれているすべてのアジェンダファイルと Org-mode ファイルをダウンロードします。ダウンロードのスピードを上げるために、MobileOrg は、どのファイルのチェック記号⁴ が変更されたかどうかを読み取るだけなのです。

B.3 MobileOrg から pull する

MobileOrg がサーバーと同期する際に、Org-mode のファイルを閲覧するために呼び出すだけではありません。それによってサーバー上の‘mobileorg.org’というファイルに対して、フラグがつけられたり、変更されたりしたエントリーに対して、キャプチャーされたエントリーやポインターを追加します。Org-mode では、この情報を InBox ファイルに統合し、フラグがつけられたエントリーにポインタを使って操作するという pull の操作機能をもっています。どのように動作するのでしょうか。

1. Org-mode は、‘mobileorg.org’⁵ の中で発見した全てのエントリーを移動し、org-mobile-inbox-for-pull変数によって、ポインターが付けられたファイルに追加します。記録されたエントリと編集されたイベントは、それぞれ InBox ファイル中でトップレベルのエントリーとして位置づけられるでしょう。
2. エントリーを移動したあと、Org-mode は、MobileOrg の中で作られた変更を実行することを試みます。いくつかの変更は直接、ユーザーの確認無しに適用されます。例では、タグ、TODO の状態、見出しそして本文に対するすべての変更がはっきりと適用されるというものです。将来の行動のために、フラグを付けられたエントリーは、:FLAGGED: というタグが付けられるでしょう。そのため、再び簡単に見つけることができるでしょう。あるエントリを探したり、変更を適用するさいに問題があれば、ポインターのついたエントリーは inboxに残され、エラーメッセージの印がつけられるでしょう。あなたはあとでこれらの案件を手動で解決する必要があります。
3. Org-mode では、その際にフラグがつけられたすべてのエントリーとともに、アジェンダビューを作成できます。そしてユーザーはそれらの項目をやり終えたり、必要な行動を実行するでしょう。MobileOrg のエントリーにフラグが付けられている間に、ノートが保存されていたら、そのノートは、カーソルがアジェンダの行の上に置かれた時に、エコーエリア上に表示されるでしょう。

? そういう特別なアジェンダの中で、?が入力されたときには、別のウインドウでフラグの付けられたノートの全てが表示され、キルリング上に内容がコピーされます。そして、? z C-y C-c C-cを使用することで、フラグのつけられたノートを、そのエントリーの通常のノートとして保存することができます。?を2度続けて入力すると、(プロパティの中に保存されていた) 記録されているフラグの付いたノートと一緒に、:FLAGGED: というタグを削除するよう指示したことになります。この方

³ アジェンダを作成する際に、Org-mode ではすべての参照されるエントリーに ID 属性を強制的に付加します。そのため、これらのエントリーは、将来の行動のために、それらのエントリーに MobileOrg によってフラグを付けたとしても、ユニークなものとして識別されます。もしも、こんなにも沢山のエントリーにそういう属性値をつけたくない場合は、org-mobile-force-id-on-agenda-items変数を nil と設定してください。Org-mode は、各エントリーが十分ユニークであることを期待したうえで、アウトラインの階層構造に依存することになるでしょう。

⁴ ‘checksums.dat’ というファイルの中に自動的に保存されます。

⁵ ‘mobileorg.org’はこの操作のあとで空になります。

法で、あなたはこのフラグの付けられたエントリーを意図したプロセスで完了させるという指示をすることになります。

もしも、すべてのフラグのついたエントリーを直接処理することができないならば、あなたは `C-ca ?` を入力して、アジェンダビュー⁶ にいつでも戻ることができます。

⁶ しかしながら、微妙な差があることに注意してください。 `M-x org-mobile-pull RET` によって、自動的に作成されたビューは、最後に pull されて配置されたすべてのファイルを検索することを保証されています。これは、あなたのアジェンダファイルのリストに、現在含まれていないファイルも含みます。もしもあなたが、ビューを再作成するために、`C-ca ?` を最後に使用したならば、カレントのアジェンダファイルのみが検索されます。

Appendix C History and acknowledgments

Org was born in 2003, out of frustration over the user interface of the Emacs Outline mode. I was trying to organize my notes and projects, and using Emacs seemed to be the natural way to go. However, having to remember eleven different commands with two or three keys per command, only to hide and show parts of the outline tree, that seemed entirely unacceptable to me. Also, when using outlines to take notes, I constantly wanted to restructure the tree, organizing it parallel to my thoughts and plans. *Visibility cycling* and *structure editing* were originally implemented in the package ‘`outline-magic.el`’, but quickly moved to the more general ‘`org.el`’. As this environment became comfortable for project planning, the next step was adding *TODO entries*, basic *timestamps*, and *table support*. These areas highlighted the two main goals that Org still has today: to be a new, outline-based, plain text mode with innovative and intuitive editing features, and to incorporate project planning functionality directly into a notes file.

Since the first release, literally thousands of emails to me or to `emacs-orgmode@gnu.org` have provided a constant stream of bug reports, feedback, new ideas, and sometimes patches and add-on code. Many thanks to everyone who has helped to improve this package. I am trying to keep here a list of the people who had significant influence in shaping one or more aspects of Org. The list may not be complete, if I have forgotten someone, please accept my apologies and let me know.

Before I get to this list, a few special mentions are in order:

Bastien Guerry

Bastien has written a large number of extensions to Org (most of them integrated into the core by now), including the LaTeX exporter and the plain list parser. His support during the early days, when he basically acted as co-maintainer, was central to the success of this project. Bastien also invented Worg, helped establishing the Web presence of Org, and sponsors hosting costs for the `orgmode.org` website.

Eric Schulte and Dan Davison

Eric and Dan are jointly responsible for the Org-babel system, which turns Org into a multi-language environment for evaluating code and doing literate programming and reproducible research.

John Wiegley

John has contributed a number of great ideas and patches directly to Org, including the attachment system (‘`org-attach.el`’), integration with Apple Mail (‘`org-mac-message.el`’), hierarchical dependencies of TODO items, habit tracking (‘`org-habits.el`’), and encryption (‘`org-crypt.el`’). Also, the capture system is really an extended copy of his great ‘`remember.el`’.

Sebastian Rose

Without Sebastian, the HTML/XHTML publishing of Org would be the pitiful work of an ignorant amateur. Sebastian has pushed this part of Org onto a much higher level. He also wrote ‘`org-info.js`’, a Java script for displaying webpages derived from Org using an Info-like or a folding interface with single-key navigation.

OK, now to the full list of contributions! Again, please let me know what I am missing here!

- *Russel Adams* came up with the idea for drawers.
- *Thomas Baumann* wrote ‘`org-bbdb.el`’ and ‘`org-mhe.el`’.
- *Christophe Bataillon* created the great unicorn logo that we use on the Org-mode website.
- *Alex Bochannek* provided a patch for rounding timestamps.
- *Jan B^{cc}b⁶cker* wrote ‘`org-docview.el`’.
- *Brad Bozarth* showed how to pull RSS feed data into Org-mode files.
- *Tom Breton* wrote ‘`org-choose.el`’.
- *Charles Cave*’s suggestion sparked the implementation of templates for Remember, which are now templates for capture.
- *Pavel Chalmoviansky* influenced the agenda treatment of items with specified time.
- *Gregory Chernov* patched support for Lisp forms into table calculations and improved XEmacs compatibility, in particular by porting ‘`noutline.el`’ to XEmacs.
- *Sacha Chua* suggested copying some linking code from Planner.
- *Baoqiu Cui* contributed the DocBook exporter.
- *Eddward DeVilla* proposed and tested checkbox statistics. He also came up with the idea of properties, and that there should be an API for them.
- *Nick Dokos* tracked down several nasty bugs.
- *Kees Dullemond* used to edit projects lists directly in HTML and so inspired some of the early development, including HTML export. He also asked for a way to narrow wide table columns.
- *Thomas S. Dye* contributed documentation on Worg and helped integrating the Org-Babel documentation into the manual.
- *Christian Egli* converted the documentation into Texinfo format, inspired the agenda, patched CSS formatting into the HTML exporter, and wrote ‘`org-taskjuggler.el`’.
- *David Emery* provided a patch for custom CSS support in exported HTML agendas.
- *Nic Ferrier* contributed mailcap and XOXO support.
- *Miguel A. Figueroa-Villanueva* implemented hierarchical checkboxes.
- *John Foerch* figured out how to make incremental search show context around a match in a hidden outline tree.
- *Raimar Finken* wrote ‘`org-git-line.el`’.
- *Mikael Fornius* works as a mailing list moderator.
- *Austin Frank* works as a mailing list moderator.
- *Eric Fraga* drove the development of BEAMER export with ideas and testing.
- *Barry Gidden* did proofreading the manual in preparation for the book publication through Network Theory Ltd.
- *Niels Giesen* had the idea to automatically archive DONE trees.
- *Nicolas Goaziou* rewrote much of the plain list code.
- *Kai Grossjohann* pointed out key-binding conflicts with other packages.

- *Brian Gough* of Network Theory Ltd publishes the Org mode manual as a book.
- *Bernt Hansen* has driven much of the support for auto-repeating tasks, task state change logging, and the clocktable. His clear explanations have been critical when we started to adopt the Git version control system.
- *Manuel Hermenegildo* has contributed various ideas, small fixes and patches.
- *Phil Jackson* wrote `'org-irc.el'`.
- *Scott Jaderholm* proposed footnotes, control over whitespace between folded entries, and column view for properties.
- *Matt Jones* wrote *MobileOrg Android*.
- *Tokuya Kameshima* wrote `'org-wl.el'` and `'org-mew.el'`.
- *Shidai Liu* ("Leo") asked for embedded L^AT_EX and tested it. He also provided frequent feedback and some patches.
- *Matt Lundin* has proposed last-row references for table formulas and named invisible anchors. He has also worked a lot on the FAQ.
- *David Maus* wrote `'org-atom.el'`, maintains the issues file for Org, and is a prolific contributor on the mailing list with competent replies, small fixes and patches.
- *Jason F. McBrayer* suggested agenda export to CSV format.
- *Max Mikhanosha* came up with the idea of refiling.
- *Dmitri Minaev* sent a patch to set priority limits on a per-file basis.
- *Stefan Monnier* provided a patch to keep the Emacs-Lisp compiler happy.
- *Richard Moreland* wrote *MobileOrg* for the iPhone.
- *Rick Moynihan* proposed allowing multiple TODO sequences in a file and being able to quickly restrict the agenda to a subtree.
- *Todd Neal* provided patches for links to Info files and Elisp forms.
- *Greg Newman* refreshed the unicorn logo into its current form.
- *Tim O'Callaghan* suggested in-file links, search options for general file links, and TAGS.
- *Osamu Okano* wrote `'orgcard2ref.pl'`, a Perl program to create a text version of the reference card.
- *Takeshi Okano* translated the manual and David O'Toole's tutorial into Japanese.
- *Oliver Oppitz* suggested multi-state TODO items.
- *Scott Otterson* sparked the introduction of descriptive text for links, among other things.
- *Pete Phillips* helped during the development of the TAGS feature, and provided frequent feedback.
- *Martin Pohlack* provided the code snippet to bundle character insertion into bundles of 20 for undo.
- *T.V. Raman* reported bugs and suggested improvements.
- *Matthias Rempé* (Oelde) provided ideas, Windows support, and quality control.
- *Paul Rivier* provided the basic implementation of named footnotes. He also acted as mailing list moderator for some time.
- *Kevin Rogers* contributed code to access VM files on remote hosts.

- *Frank Rueell* solved the mystery of the `keymapp nil` bug, a conflict with `'allout.el'`.
- *Jason Riedy* generalized the send-receive mechanism for Orgtbl tables with extensive patches.
- *Philip Rooke* created the Org reference card, provided lots of feedback, developed and applied standards to the Org documentation.
- *Christian Schlauer* proposed angular brackets around links, among other things.
- *Paul Sexton* wrote `'org-ctags.el'`.
- Linking to VM/BBDB/Gnus was first inspired by *Tom Shannon's* `'organizer-mode.el'`.
- *Ilya Shlyakhter* proposed the Archive Sibling, line numbering in literal examples, and remote highlighting for referenced code lines.
- *Stathis Sideris* wrote the `'ditaa.jar'` ASCII to PNG converter that is now packaged into Org's `'contrib'` directory.
- *Daniel Sinder* came up with the idea of internal archiving by locking subtrees.
- *Dale Smith* proposed link abbreviations.
- *James TD Smith* has contributed a large number of patches for useful tweaks and features.
- *Adam Spiers* asked for global linking commands, inspired the link extension system, added support for mairix, and proposed the mapping API.
- *Ulf Stegemann* created the table to translate special symbols to HTML, LaTeX, UTF-8, Latin-1 and ASCII.
- *Andy Stewart* contributed code to `'org-w3m.el'`, to copy HTML content with links transformation to Org syntax.
- *David O'Toole* wrote `'org-publish.el'` and drafted the manual chapter about publishing.
- *Sebastien Vauban* reported many issues with LaTeX and BEAMER export and enabled source code highlighting in Gnus.
- *Stefan Vollmar* organized a video-recorded talk at the Max-Planck-Institute for Neurology. He also inspired the creation of a concept index for HTML export.
- *Jürgen Vollmer* contributed code generating the table of contents in HTML output.
- *Samuel Wales* has provided important feedback and bug reports.
- *Chris Wallace* provided a patch implementing the `'QUOTE'` keyword.
- *David Wainberg* suggested archiving, and improvements to the linking system.
- *Carsten Wimmer* suggested some changes and helped fix a bug in linking to Gnus.
- *Roland Winkler* requested additional key bindings to make Org work on a tty.
- *Piotr Zielinski* wrote `'org-mouse.el'`, proposed agenda blocks and contributed various ideas and code snippets.

Concept index

名前 (列やフィールド)	26	固定幅の段落	126
名前を TODO キーワードとして	43		
		拡張された TODO キーワード	43
画像、HTML の中でインライン	130		
画像、LaTeX に中のインライン	135	動的なインデント	180
画像、インライン	39		
		時間順に並べたビュー	97
改行の維持	126	時間の情報、エクスポートの中で	126
見出しとセクション、マークアップのルール	116	先頭の「*」を隠す	180
見出しの階層	126	属性、並び順	46, 52, 53
見出し、LaTeX ファイルのための	134	属性、ログをとる	48, 61
		属性、COOKIE_DATA	51, 52
概要	1		
		目次	126
強調されたテキスト	126	目次、マークアップのルール	116
		章の番号	126
(外部出力に用いる) ヘッドラインレベル ..	128, 129, 134	構造の分割、LaTeX エクスポートのための ..	134
公開のためのインデックスのエントリ	120	著者	4
奇数レベルのみのアウトライン	180	著者の情報、エクスポートの中で	126
特別な文字列	126	取り消されたテキスト、マークアップのルール	117
特別なプロパティ (CLOCKSUM)	59, 115		
		補完, リンクの	37
日記の統合	93	補完, ファイル名の	38
		表、マークアップのルール	118
特殊記号	121	数学記号	121

- 脚注、マークアップのルール 117
- 文書のタイトル、マークアップのルール 116
 文字通りのテキスト、マークアップのルール 117
 斜体のテキスト、マークアップのルール.. 117
 暫定マークモード 11, 21
- 省略記法, リンクの 39
- 最初の見出しより前のテキスト、マークアップ
 のルール 116
- 太字のテキスト、マークアップのルール.. 117
 大括弧, リンクの周辺 36
 外部リンク 35
 外部リンク、出力する HTML の 129
- 検索文字列, カスタム 41
 検索ビュー 98
- 計算中の定数 26
- 種類を TODO キーワードとして 43
- 段落、マークアップのルール 117
- 完了、タグの 54, 174
 完了、TODO キーワードの 43, 174
- 水平線、マークアップのルール 118
- 下付き文字 121
 下線のあるテキスト、マークアップのルール
 117
- 上付き文字 121
 上付き、下付き文字を示す $\text{T}_{\text{E}}\text{X}$ のようなシン
 タックス 126
- 内部リンク 34
 内部リンク、出力する HTML の 129
- リンクの保存 36
 リンクの補完 37
 リンクの省略記法 39
 リンクの挿入 37
 リンクのフォーマット 34
 リンク、出力する HTML の 129
 リンクをたどる 38
 リンク、次/前を探す 39
 リンク、扱い 36
 リンク、戻る 39
 リンク、ラジオターゲット 35
 リテラルの例、マークアップのルール 118
 リスト、マークアップのルール 117
 バッファ中での設定 176
 バグレポート 4
 プレーンテキスト外部リンク 36
 プロパティ (API) 65, 196
 プロパティ (ARCHIVE) 61, 88
 プロパティ (CATEGORY) 61, 99
 プロパティ (COLUMNS) 60, 177
 プロパティのための API 65, 196
 プロパティ、EXPORT_FILE_NAME... 127, 128,
 133, 138
 プロパティ、EXPORT_TITLE 116
 はじめに 1
 ラジオターゲット 35
 メンテナ 4
 パッケージ、他のものとの連携 183
 フォーマット, リンクの 34
 フィードバック 4
 ファイル名の補完 38
 ファイル毎のキーワード 45
 ファイルリンク 35
 ファイルリンクにおける検索オプション 40
 ファイルリンク, 検索 40
 ファイルのインクルード、マークアップのルー
 ル 120
 ファイル、アジェンダリストに追加する 90
 ハイパーリンク 34
 マークリング 39
 テンプレートの挿入 174
 テーブルの中での計算 21, 24
 テーブル、HTML の 130
 テキストエリア、HTML の中の 131
 オプションのキーワードの補完 45, 125, 174
 アウトラインビューを見やすくする 180

アクティブなリージョン ...	11, 21, 127, 128, 133, 138
アジェンダ	92
アジェンダ用のファイル	90
アジェンダビュー	90
アジェンダビュー (出力)	109, 113
アジェンダビューの出力	109, 113
アジェンダファイル	90
アジェンダのコマンド選択画面	91
アジェンダのコマンドを選択する	91
コードラインのリファレンス、マークアップの ルール	118
コードのテキスト、マークアップのルール ..	117
ターゲット, リンクの	34
ターゲット, ラジオ	35
ソースコードのフォーマット、マークアップの ルール	119
コメント行	118
コマンド選択画面、エクスポートコマンドのた めの	127
タイムスタンプ	66
スペシャルキーワード	176
カレンダーの統合	93
キーバインド (グローバル)	4
キーワードオプション	45
キャプチャ	80
カウントダウンタイマ	79
カスタム検索文字列	41
カスタマイズ	176
カスタマイズの変数	176
カスタマイズのオプション	176
インライン画像	39
インライン画像、マークアップのルール ..	118
インストール	3
エラーのバクトレース	5
エクスポート	125
エクスポート中のマクロによる置き換え ..	120
エクスポートされない部分	118
エクスポートのオプション	125
エクスポート。タグによる選択	125
グローバルなキーバインド	4
#	
#+ARCHIVE	88
#+ATTR_DOCBOOK	140
#+ATTR_HTML	130
#+ATTR_LaTeX	135
#+AUTHOR	125
#+BEGIN, clocktable	75
#+BEGIN, columnview	64
#+BEGIN:dynamic block	193
#+BEGIN_CENTER	117
#+BEGIN_COMMENT	118
#+BEGIN_DOCBOOK	138
#+BEGIN_EXAMPLE	118
#+BEGIN_HTML	129
#+BEGIN_LaTeX	135
#+BEGIN_QUOTE	117
#+BEGIN_SRC	119
#+BEGIN_VERSE	117
#+BIND	125
#+CAPTION	118, 130, 135, 140
#+CATEGORY	99
#+COLUMNS	61
#+CONSTANTS	26
#+DATE	125
#+DESCRIPTION	125
#+DOCBOOK	138
#+DRAWERS	16
#+EMAIL	125
#+EXPORT_EXCLUDE_TAGS	125
#+EXPORT_SELECT_TAGS	125
#+FILETAGS	54
#+HTML	129
#+INCLUDE	120
#+INFOJS_OPT	132
#+KEYWORDS	125
#+LABEL	118, 135, 140
#+LANGUAGE	125
#+LaTeX	135
#+LATEX_CLASS	134
#+LATEX_CLASS_OPTIONS	134
#+LATEX_HEADER	125, 134
#+LINK	40
#+LINK_HOME	125
#+LINK_UP	125
#+MACRO	120
#+OPTIONS	116, 125
#+ORGLST	192
#+ORGTBL	189
#+ORGTBL, SEND	189
#+PLOT	32
#+PRIORITIES	50
#+PROPERTY	58
#+SEQ_TODO	45
#+SETUPFILE	177
#+STARTUP:	177
#+STYLE	132
#+TAGS	55
#+TBLFM	28
#+TBLNAME	26
#+TEXT	116, 125
#+TITLE	116, 125
#+TODO	45
#+TYP_TODO	45
#+XSLT	125
1	
1 日のアジェンダ	92
1 週間のアジェンダ	92
A	
acknowledgments	203

action, for publishing 145
activation 4
add-on packages 186
add-ons, context-sensitive commands 188
agenda files, removing buffers 110
agenda views, custom 110
agenda views, user-defined 193
agenda, column view 115
agenda, pipe 195
agenda, with block views 111
alignment in tables 22
anniversaries, from BBDB 94
API, for mapping 198
appointment reminders 94
‘`appt.el`’ 94
archive locations 88
archiving 88
ASCII 形式へのエクスポート 127
Atom feeds 86
attachments 85
autoload 4

B

babel, languages 156
babel, library of 155
Baur, Steven L. 184
BBDB リンク 35
BBDB, anniversaries 94
block agenda 111
blocking, of checkboxes 52
blocks, folding 16
Boolean logic, for tag/property searches 96

C

C-c C-c、概観 180
‘`calc`’ package 24
‘`calc.el`’ 183
calendar commands, from agenda 108
calendar, for selecting date 69
category 99
category, require for tags/property match 96
‘`cdlatex.el`’ 183
CDLaTeX 123
checkbox blocking 52
checkbox statistics 52
checkboxes 51
checkboxes and TODO dependencies 46
children, subtree visibility state 7
clocking time 73
clocktable, dynamic block 74
code block, batch execution 172
code block, editing 153
code block, evaluating 154
code block, exporting 153
code block, extracting source code 154
code block, header arguments 156

code block, key bindings 172
code block, languages 156
code block, library 155
code block, noweb reference 171
code block, results of evaluation 170
code block, structure 152
column formula 28
column view, for properties 61
column view, in agenda 115
column, of field coordinates 25
commands, in agenda buffer 101
completion, of dictionary words 174
completion, of link abbreviations 174
completion, of property keys 174
completion, of TeX symbols 174
‘`constants.el`’ 183
`content`, STARTUP キーワード 8, 177
contents, global visibility state 8
context-sensitive commands, hooks 188
coordinates, of field 25
copying, of subtrees 9
creating timestamps 67
CSS、HTML エクスポートに関する 131
‘`CUA.el`’ 184
Cui, Baoqiu 138
custom agenda views 110
custom date/time format 69
cutting, of subtrees 9
cycling, visibility 7

D

date format, custom 69
date range 66
date stamp 66
date stamps 66
date tree 80
date, reading in minibuffer 68
dates 66
Davison, Dan 152
DEADLINE keyword 70
deadlines 66
debugging, of table formulas 30
demotion, of subtrees 9
dependencies, of TODO states 46
diary entries, creating from agenda 108
dictionary word completion 174
directories, for publishing 144
display changing, in agenda 102
DocBook 出力における特殊文字 140
DocBook export 138
DocBook recursive sections 139
document structure 7
Dominik, Carsten 183
DONE は最終の TODO キーワード 45
drawer, for properties 58
drawer, for state change recording 47
drawers 16

dvipng 130
dynamic blocks 192

E

editing tables 19
editing, of table formulas 29
effort estimates 77
effort filtering, in agenda 104
ELisp リンク 35
emacsserver 87
entitiesplain, STARTUP キーワード 179
entitiespretty, STARTUP キーワード 179
evaluate time range 67
external archiving 88

F

FAQ 1
field coordinates 25
field formula 28
field references 24
files, selecting for publishing 145
filtering, by tag and effort, in agenda 104
fnadjust, STARTUP キーワード 179
fnauto, STARTUP キーワード 179
fnconfirm, STARTUP キーワード 179
fninline, STARTUP キーワード 179
fnlocal, STARTUP キーワード 179
fnplain, STARTUP キーワード 179
fnprompt, キーワード 179
folded, subtree visibility state 7
folding, sparse trees 12
'footnote.el' 117, 184
footnotes 16, 126
format specifier 26
formula debugging 30
formula editing 29
formula syntax, Calc 26
formula, for individual table field 28
formula, for range of fields 28
formula, for table column 28
formula, in tables 21
Freemind export 142

G

Gillespie, Dave 183
global cycling 8
global TODO list 94
global visibility states 8
Gnus リンク 35
graph, in tables 32
grouping columns in tables 23
Guerry, Bastien 133

H

habits 48
hacking 186
headline navigation 9
headline tagging 54
headline, promotion and demotion 9
headlines 7
hide text 7
hideblocks, STARTUP keyword 16, 179
history 203
hooks 186
HTML の引用タグ 126
HTML のインライン画像 130
HTML のエントリ 121
HTML エクスポート、CSS 131
HTML export 128
HTML, and Orgtbl mode 191
hyperlinks, adding new types 186

I

iCalendar エクスポート 143
idle, resolve, dangling 76
images, inline in DocBook 139
'imenu.el' 183
inactive timestamp 66
index, in a publishing project 148
Info リンク 35
inheritance, of properties 60
inheritance, of tags 54
inlineimages, STARTUP keyword 39, 178
inlining images in DocBook 139
iPhone 200
IRC リンク 35

J

jumping, to headlines 9

L

LaTeX のコード片、プレビュー 123
L^AT_EX の見出し 134
L^AT_EX の構造の分割 134
L^AT_EX の解釈 121
L^AT_EX の断片的なコード 122, 126
L^AT_EX の断片、マークアップのルール 121
L^AT_EX の中のインライン画像 135
L^AT_EX のエントリ 121
L^AT_EX のエクスポート 133
L^AT_EX クラス 134
L^AT_EX, and Orgtbl mode 189
Latin-1 でのエクスポート 127
level, require for tags/property match 96
link abbreviations, completion of 174
links, external 35
links, internal 34

links, publishing	147
Lisp forms, as table formulas	27
lists, in other modes	188
lists, ordered	13
lists, plain	13
logging, of progress	47
Ludlam, Eric M.	183

M

mapping entries, API	198
marking characters, tables	32
match view	95
matching, of properties	95
matching, of tags	95
matching, tags	54
MathJax	130
MH-E リンク	35
mind map	142
minor mode for structure editing	18
minor mode for tables	23
MobileOrg	200
mode, for ‘calc’	26
motion commands in agenda	101
motion, between headlines	9

N

named references	26
narrow columns in tables	22
nofnadjust, STARTUP キーワード	179
nofninline, STARTUP キーワード	179
nohideblocks, STARTUP keyword	16, 179
noinlineimages, STARTUP keyword	39, 178

O

occur, command	12
option keyword completion	174
options, for custom agenda views	111
options, for publishing	146
ordered lists	13
org-agenda, command	93
org-hide-block-startup	179
org-list-insert-radio-list	192
Org-mode (利用開始)	4
org-pretty-entities	179
org-publish-project-alist	144
Orgstruct mode	18
Orgtbl mode	23, 188
Ota, Takaaki	183
Outline mode	7
outline tree	7
outlines	7
overview, global visibility state	8
overview, STARTUP キーワード	8, 177

P

pastings, of subtrees	9
PDF 出力	133, 138
plain lists	13
plot tables using Gnuplot	32
presentation, of agenda items	99
print edition	1
printing sparse trees	12
priorities	50
priorities, of agenda items	100
progress logging	47
Project management	140
projects, for publishing	144
promotion, of subtrees	9
properties	58
properties, column view	61
properties, inheritance	60
properties, searching	60
properties, special	59
property syntax	58
property, _ALL	58
property, ATTACH_DIR	86
property, ATTACH_DIR_INHERIT	86
property, CUSTOM_ID	34, 36
property, DESCRIPTION	143
property, Effort	77
property, ID	36, 64, 143
property, LATEX_CLASS	134
property, LATEX_CLASS_OPTIONS	134
property, LOCATION	143
property, LOG_INTO_DRAWER	47
property, special, ALLTAGS	59
property, special, BLOCKED	59
property, special, CATEGORY	59
property, special, CLOSED	59
property, special, DEADLINE	59
property, special, FILE	59
property, special, ITEM	59
property, special, PRIORITY	59
property, special, SCHEDULED	59
property, special, TAGS	59
property, special, TIMESTAMP	59
property, special, TIMESTAMP_IA	59
property, special, TODO	59
property, SUMMARY	143
property, VISIBILITY	8
property: CLOCK_MODELINE_TOTAL	73
property: LAST_REPEAT	73
protocols, for external access	87
publishing	144

Q

query editing, in agenda	104
--------------------------------	-----

R

radio lists	192
-------------------	-----

radio tables 188
 range formula 28
 range references 25
 ranges, time 66
 recomputing table fields 30
 references 24
 references, named 26
 references, remote 26
 references, to a different table 26
 references, to fields 24
 references, to ranges 25
 refiling notes 87
 region, active 11, 21, 127, 128, 133, 138
 regular expressions, with tags search 96
 relative timer 78
 ‘remember.el’ 183
 remote editing, bulk, from agenda 107
 remote editing, from agenda 105
 remote editing, undo 105
 remote references 26
 repeated tasks 71
 report, of clocked time 74
 resolve idle time 76
 RMAIL リンク 35
 Rose, Sebastian 132
 row, of field coordinates 25
 RSS フィールド 86
 rsync 148

S

SCHEDULED keyword 70
 scheduling 66
 Schulte, Eric 152
 Scripts, for agenda processing 195
 searching for tags 56
 searching, for text 98
 searching, of properties 60
 setting tags 54
 SHELL リンク 35
 shift-selection-mode 15
 shift-selection-mode 184
 show all, command 8
 show all, global visibility state 8
 show hidden text 7
 showall, STARTUP キーワード 8, 177
 showeverything, STARTUP キーワード 8, 177
 sitemap, of published pages 147
 sorting, of agenda items 100
 sorting, of subtrees 9
 source code, batch execution 172
 source code, block header arguments 156
 source code, block structure 152
 source code, editing 153
 source code, evaluating 154
 source code, exporting 153
 source code, extracting 154
 source code, languages 156

source code, library 155
 source code, noweb reference 171
 source code, results of evaluation 170
 source code, working with 152
 sparse tree, for deadlines 71
 sparse tree, tag based 54
 sparse trees 12
 speed keys 175
 ‘speedbar.el’ 183
 spreadsheet capabilities 24
 STARTUP キーワード、<nologrefile 178
 STARTUP キーワード、align 178
 STARTUP キーワード、indent 177
 STARTUP キーワード、logdone 178
 STARTUP キーワード、lognoredeadline 178
 STARTUP キーワード、lognoteclock-out .. 178
 STARTUP キーワード、lognotedone 178
 STARTUP キーワード、lognoterefile 178
 STARTUP キーワード、lognoterepeat 178
 STARTUP キーワード、lognotereschedule .. 178
 STARTUP キーワード、logredeadline 178
 STARTUP キーワード、logrefile 178
 STARTUP キーワード、logrepeat 178
 STARTUP キーワード、logschedule 178
 STARTUP キーワード、noalign 178
 STARTUP キーワード、noindent 177
 STARTUP キーワード、nologdone 178
 STARTUP キーワード、nologredeadline 178
 STARTUP キーワード、nologrepeat 178
 STARTUP キーワード、nologreschedule 178
 STARTUP keyword、constcgs 179
 STARTUP keyword、constSI 179
 STARTUP keyword、customtime 179
 STARTUP keyword、even 179
 STARTUP keyword、hidestars 179
 STARTUP keyword、odd 179
 STARTUP keyword、showstars 179
 statistics, for checkboxes 52
 statistics, for TODO items 51
 STATUP キーワード、nolognoteclock-out .. 178
 Storm, Kim. F. 184
 structure editing 9
 structure of document 7
 sublevels, inclusion into tags match 54
 sublevels, inclusion into TODO list 95
 subtree cycling 7
 subtree visibility states 7
 subtree, cut and paste 9
 subtree, subtree visibility state 7
 subtrees, cut and paste 9
 syntax, noweb 171
 syntax, of formulas 26

T

table editor, built-in 19
 table editor, ‘table.el’ 183
 ‘table.el’ 183

- tables..... 19, 126
 - tables, in DocBook export..... 139
 - tables, in L^AT_EX export..... 135
 - tables, in other modes..... 188
 - tag completion..... 174
 - tag filtering, in agenda..... 104
 - tag inheritance..... 54
 - tag searches..... 56
 - tags..... 54
 - tags view..... 95
 - tags, setting..... 54
 - tangling..... 154
 - TaskJuggler export..... 140
 - tasks, breaking down..... 51
 - tasks, repeated..... 71
 - templates, for Capture..... 81
 - T_EX マクロ..... 121, 126
 - T_EX の解釈..... 121
 - T_EX symbol completion..... 174
 - text search..... 98
 - thanks..... 203
 - time clocking..... 73
 - time format, custom..... 69
 - time grid..... 100
 - time, reading in minibuffer..... 68
 - time-of-day specification..... 100
 - timeline, single file..... 97
 - timerange..... 66
 - times..... 66
 - timestamp, inactive..... 66
 - timestamp, with repeater interval..... 66
 - timestamps..... 66
 - timestamps, creating..... 67
 - TODO ワークフロー..... 43
 - TODO の状態の切り替え..... 42
 - TODO のためのツリーの抽出..... 42
 - TODO アイテム..... 42
 - TODO キーワードとしてのワークフローの状態..... 43
 - TODO キーワードのフェイス..... 45
 - TODO キーワードセット..... 44
 - TODO dependencies..... 46
 - TODO keyword matching..... 94
 - TODO keyword matching, with tags search.... 96
 - TODO keywords completion..... 174
 - TODO list, global..... 94
 - TODO types..... 43
 - transient-mark-mode..... 127, 128, 133, 138
 - translator function..... 191
 - trees, sparse..... 12
 - trees, visibility..... 7
 - tty 端末のキーバインディング..... 182
- ## U
- undoing remote-editing events..... 105
 - unison..... 148
 - updating, table..... 30
 - URL リンク..... 35
 - USENET リンク..... 35
 - UTF-8 でのエクスポート..... 127
- ## V
- vectors, in table calculations..... 26
 - ‘viper.el’..... 185
 - visibility cycling..... 7
 - visibility cycling, drawers..... 16
 - visible text, printing..... 12
 - VM リンク..... 35
- ## W
- WANDERLUST リンク..... 35
 - Wiegley, John..... 183
 - ‘windmove.el’..... 185
- ## X
- XEmacs..... 3
 - XOXO export..... 142
- ## Y
- ‘yasnippet.el’..... 185

Key index

\$		[
\$.....	106	[.....	102, 105
,]	
'.....	124].....	105
+		^	
+.....	106	^.....	123
,		-	
,.....	106	-.....	123
-		`	
-.....	106	`.....	123
.		\	
.....	102	\.....	105
/		 	
/.....	104	{.....	105
		}.....	105
:		A	
:.....	106	a.....	64
		a.....	106
;		B	
;.....	79	B.....	108
<		C	
<.....	64	c.....	108
<.....	69	C.....	109
<.....	91	C-#.....	31
<TAB>.....	19	C-'.....	91
		C-,.....	91
>		C_.....	105
>.....	64	C-0 C-c C-w.....	87
>.....	69	C-c !.....	67
>.....	91	C-c #.....	53
>.....	107	C-c \$.....	88
		C-c %.....	39
?		C-c &.....	39
?.....	201	C-c '.....	29, 119, 120, 153
		C-c '.....	184
		C-c *.....	11

C-c *	15	C-c C-a s	86
C-c *	31	C-c C-a z	85
C-c +	21	C-c C-b	9
C-c ,	50	C-c C-b	137
C-c -	15, 20	C-c C-c	15, 18, 19
C-c	67	C-c C-c	30
C-c /	12	C-c C-c	52
C-c /	185	C-c C-c	54, 59, 64, 65, 74, 80
C-c / a	71	C-c C-c	123, 155, 172, 180, 183
C-c / b	71	C-c C-c c	59
C-c / d	71	C-c C-c d	59
C-c / m	57, 60	C-c C-c D	59
C-c / p	60	C-c C-c s	59
C-c / r	12	C-c C-d	71
C-c / t	42	C-c C-d	106
C-c ;	118	C-c C-e	127
C-c <	67	C-c C-e a	127
C-c =	29	C-c C-e A	128
C-c >	67	C-c C-e b	129
C-c ?	29	C-c C-e c	143
C-c [91	C-c C-e d	134
C-c]	91	C-c C-e D	138
C-c ^	11	C-c C-e E	150
C-c ^	15, 20	C-c C-e F	150
C-c `	21	C-c C-e h	128
C-c \	57	C-c C-e H	129
C-c	19	C-c C-e i	143
C-c	22	C-c C-e I	143
C-c {	29, 123	C-c C-e j	140
C-c }	29, 30	C-c C-e J	140
C-c ~	184	C-c C-e l	133
C-c a !	98	C-c C-e L	134
C-c a #	98	C-c C-e m	142
C-c a ?	202	C-c C-e n	128
C-c a a	93	C-c C-e N	128
C-c a C	110	C-c C-e p	134
C-c a e	114	C-c C-e P	150
C-c a L	98	C-c C-e R	129
C-c a m	57, 60, 95	C-c C-e t	125
C-c a M	57, 60, 95	C-c C-e u	128
C-c a s	98	C-c C-e U	128
C-c a t	43, 94	C-c C-e v	12, 127
C-c c	80	C-c C-e V	138
C-c c C	81	C-c C-e v D	138
C-c C-*	15	C-c C-e v x	142
C-c C-a	85, 106	C-c C-e x	142
C-c C-a a	85	C-c C-e X	150
C-c C-a c	85	C-c C-f	9
C-c C-a d	86	C-c C-j	9
C-c C-a D	86	C-c C-k	8, 81
C-c C-a f	86	C-c C-l	37
C-c C-a F	86	C-c C-n	9
C-c C-a i	86	C-c C-o	18, 38
C-c C-a l	85	C-c C-o	67, 101
C-c C-a m	85	C-c C-o	172
C-c C-a n	85	C-c C-p	9
C-c C-a o	85	C-c C-q	30, 54
C-c C-a O	85	C-c C-r	8, 30

C-c C-s	71	C-c C-x C-s	106
C-c C-s	106	C-c C-x C-t	69
C-c C-t	42	C-c C-x C-u	65, 74, 193
C-c C-t	74	C-c C-x C-v	38
C-c C-u	9	C-c C-x C-w	10, 21
C-c C-v a	172	C-c C-x C-x	74
C-c C-v b	172	C-c C-x C-y	10, 21
C-c C-v C-a	172	C-c C-x e	78
C-c C-v C-b	172	C-c C-x f	17
C-c C-v C-f	172	C-c C-x g	86
C-c C-v C-l	172	C-c C-x G	86
C-c C-v C-p	172	C-c C-x i	65
C-c C-v C-s	172	C-c C-x M-w	10, 21
C-c C-v C-t	172	C-c C-x o	46, 53
C-c C-v C-z	172	C-c C-x p	59
C-c C-v f	172	C-c C-x p	158
C-c C-v g	172	C-c C-y	67, 74
C-c C-v h	172	C-c C-z	16, 106
C-c C-v i	155	C-c l	36
C-c C-v l	172	C-c l	120
C-c C-v p	172	C-c qrg-match-sparse-tree	60
C-c C-v s	172	C-c RET	20
C-c C-v t	154, 172	C-k	105
C-c C-v z	172	C-RET	9
C-c C-w	11, 80, 87	C-S-left	44, 105
C-c C-w	105	C-S-RET	10
C-c C-x ,	79	C-S-right	44, 105
C-c C-x -	78	C-TAB	89
C-c C-x	78	C-u C-c !	67
C-c C-x ;	79	C-u C-c *	31
C-c C-x <	91	C-u C-c	67
C-c C-x >	91	C-u C-c =	28
C-c C-x >	103	C-u C-c =	29
C-c C-x \	121, 122	C-u C-c c	81
C-c C-x 0	79	C-u C-c C-c	31
C-c C-x a	89	C-u C-c C-l	38
C-c C-x a	106	C-u C-c C-t	42
C-c C-x A	89	C-u C-c C-w	87
C-c C-x A	106	C-u C-c C-x ,	79
C-c C-x b	8, 101	C-u C-c C-x a	89
C-c C-x c	11	C-u C-c C-x C-s	88
C-c C-x C-a	88	C-u C-c C-x C-u	65, 74, 193
C-c C-x C-a	106	C-u C-u C-c *	31
C-c C-x C-b	52	C-u C-u C-c =	29
C-c C-x C-c	63, 103	C-u C-u C-c c	81
C-c C-x C-c	115	C-u C-u C-c C-c	31
C-c C-x C-d	74	C-u C-u C-c C-e	127
C-c C-x C-e	74	C-u C-u C-c C-t	44
C-c C-x C-e	78	C-u C-u C-c C-w	87
C-c C-x C-i	73	C-u C-u C-u C-c C-t	46
C-c C-x C-j	74	C-u C-u C-u TAB	8
C-c C-x C-k	71	C-u C-u TAB	8
C-c C-x C-l	123	C-up	172
C-c C-x C-n	39	C-v	69
C-c C-x C-o	73	C-x C-s	30
C-c C-x C-p	39	C-x C-w	109
C-c C-x C-r	74	C-x C-w	113
C-c C-x C-s	88	C-x n b	11

C-x n s..... 11
 C-x n w..... 11
 C-y..... 10

D

d..... 102
 D..... 102

E

e..... 63
 E..... 103

F

f..... 102
 F..... 101

G

g..... 63, 103
 G..... 103

H

H..... 109

I

i..... 108
 I..... 107

J

j..... 102
 J..... 102, 107

K

k..... 107
 k a..... 71
 k s..... 71

L

l..... 102
 L..... 101

M

m..... 107
 M..... 109
 M-a..... 20
 M-down..... 20
 M-down..... 30
 M-down..... 172
 M-e..... 20

M-g M-n..... 12
 M-g M-p..... 12
 M-g n..... 12
 M-g p..... 12
 M-left..... 10, 15, 20
 M-RET..... 9
 M-RET..... 14
 M-RET..... 21
 M-RET..... 79
 M-right..... 10, 15, 20
 M-S-down..... 10, 15, 20
 M-S-down..... 30
 M-S-left..... 10, 15, 20, 69
 M-S-RET..... 10, 14
 M-S-RET..... 53
 M-S-right..... 10, 15, 20, 69
 M-S-up..... 10, 15, 20
 M-S-up..... 30
 M-TAB..... 30
 M-TAB..... 45, 54
 M-TAB..... 59
 M-TAB..... 174
 M-up..... 20
 M-up..... 30
 M-v..... 69
 M-x org-iswitchb..... 91
 mouse-1..... 18, 38, 69
 mouse-2..... 18, 38
 mouse-2..... 101
 mouse-3..... 38
 mouse-3..... 101

N

n..... 63
 n..... 101

O

o..... 102
 O..... 107

P

p..... 63
 p..... 101
 P..... 106

Q

q..... 63, 109

R

r..... 63, 95, 103
 R..... 103
 RET..... 20, 38, 56, 69
 RET..... 101

S

S.....	109
S-down.....	14, 30
S-down.....	50, 67
S-down.....	69, 106
S-left.....	15, 30, 42, 44
S-left.....	59
S-left.....	63
S-left.....	67
S-left.....	69
S-left.....	74
S-left.....	107
S-M-left.....	64
S-M-RET.....	43
S-M-right.....	64
S-RET.....	21
S-right.....	15, 30, 42, 44
S-right.....	59
S-right.....	63
S-right.....	67
S-right.....	69
S-right.....	74
S-right.....	107
S-TAB.....	8
S-TAB.....	20
S-up.....	30
S-up.....	50, 67
S-up.....	69, 106
SPC.....	56
SPC.....	101

T

t.....	105
T.....	106
TAB.....	7, 10
TAB.....	14, 30

TAB.....	56
TAB.....	101
TAB.....	123

U

U.....	107
U.....	108

V

v.....	64
v [.....	102
v a.....	103
v A.....	103
v d.....	102
v E.....	103
v l.....	102
v L.....	102
v m.....	102
v R.....	103
v SPC.....	102
v w.....	102
v y.....	102

W

w.....	102
--------	-----

X

x.....	110
X.....	107

Z

z.....	106
--------	-----

Command and function index

C

Create a sparse tree with all matching entries. With a *C-u* prefix argument 60

L

lisp-complete-symbol 30

N

next-error 12

O

org-agenda-day-view 102
 org-agenda-action 107
 org-agenda-add-note 106
 org-agenda-archive 106
 org-agenda-archive-default-with-confirmation 106
 org-agenda-archive-to-archive-sibling ... 106
 org-agenda-archives-mode 103
 org-agenda-archives-mode 'files 103
 org-agenda-bulk-action 108
 org-agenda-bulk-mark 107
 org-agenda-bulk-remove-all-marks 107
 org-agenda-bulk-remove-all-marks 108
 org-agenda-clock-cancel 107
 org-agenda-clock-goto 102, 107
 org-agenda-clock-in 107
 org-agenda-clock-out 107
 org-agenda-clockreport-mode 103
 org-agenda-columns 103
 org-agenda-columns 115
 org-agenda-convert-date 109
 org-agenda-date-prompt 107
 org-agenda-deadline 106
 org-agenda-diary-entry 108
 org-agenda-do-date-earlier 107
 org-agenda-do-date-later 107
 org-agenda-entry-text-mode 103
 org-agenda-exit 110
 org-agenda-file-to-front 91
 org-agenda-filter-by-tag 104
 org-agenda-filter-by-tag-refine 105
 org-agenda-follow-mode 101
 org-agenda-goto 101
 org-agenda-goto-calendar 108
 org-agenda-goto-date 102
 org-agenda-goto-today 102
 org-agenda-holidays 109
 org-agenda-kill 105
 org-agenda-later 102
 org-agenda-list 93

org-agenda-list-stuck-projects 98
 org-agenda-log-mode 102
 org-agenda-manipulate-query-add 102
 org-agenda-month-view 102
 org-agenda-month-year 102
 org-agenda-next-line 101
 org-agenda-open-link 101
 org-agenda-phases-of-moon 109
 org-agenda-previous-line 101
 org-agenda-priority-down 106
 org-agenda-priority-up 106
 org-agenda-quit 109
 org-agenda-recenter 101
 org-agenda-refile 105
 org-agenda-remove-restriction-lock 91
 org-agenda-remove-restriction-lock 103
 org-agenda-reset-view 102
 org-agenda-rodo 103
 org-agenda-schedule 106
 org-agenda-set-restriction-lock 91
 org-agenda-set-tags 106
 org-agenda-show-and-scroll-up 101
 org-agenda-show-priority 106
 org-agenda-show-tags 106
 org-agenda-sunrise-sunset 109
 org-agenda-switch-to 101
 org-agenda-todo 105
 org-agenda-todo-nextset 105
 org-agenda-todo-previousset 105
 org-agenda-toggle-archive-tag 106
 org-agenda-toggle-diary 102
 org-agenda-toggle-time-grid 103
 org-agenda-tree-to-indirect-buffer 101
 org-agenda-undo 105
 org-archive-subtree 88
 org-archive-subtree-default 88
 org-archive-to-archive-sibling 89
 org-attach 85, 106
 org-attach-attach 85
 org-attach-delete-all 86
 org-attach-delete-one 86
 org-attach-new 85
 org-attach-open 85
 org-attach-open-in-emacs 85
 org-attach-reveal 86
 org-attach-reveal-in-emacs 86
 org-attach-set-directory 86
 org-attach-set-inherit 86
 org-attach-sync 85
 org-backward-same-level 9
 org-beamer-select-environment 137
 org-buffer-property-keys 197
 org-calendar-goto-agenda 108
 org-capture 80
 org-capture-finalize 80

org-capture-kill	81	org-export-as-ascii-to-buffer	128
org-capture-refile	80	org-export-as-docbook	138
org-check-after-date	71	org-export-as-docbook-pdf-and-open	138
org-check-before-date	71	org-export-as-freemind	142
org-check-deadlines	71	org-export-as-html	128
org-clock-cancel	74	org-export-as-html-and-open	129
org-clock-display	74	org-export-as-html-to-buffer	129
org-clock-goto	74	org-export-as-latex	133
org-clock-in	73	org-export-as-latex-to-buffer	134
org-clock-modify-effort-estimate	74	org-export-as-latin1	128
org-clock-modify-effort-estimate	78	org-export-as-latin1-to-buffer	128
org-clock-out	73	org-export-as-pdf	134
org-clock-report	74	org-export-as-pdf-and-open	134
org-clocktable-try-shift	74	org-export-as-taskjuggler	140
org-clone-subtree-with-time-shift	11	org-export-as-taskjuggler-and-open	140
org-columns	63	org-export-as-utf8	128
org-columns-delete	64	org-export-as-utf8-to-buffer	128
org-columns-edit-allowed	64	org-export-as-xoxo	142
org-columns-edit-value	63	org-export-icalendar-all-agenda-files	143
org-columns-narrow	64	org-export-icalendar-combine-agenda-files	143
org-columns-new	64	org-export-icalendar-this-file	143
org-columns-next-allowed-value	63	org-export-region-as-html	129
org-columns-previous-allowed-value	63	org-export-visible	127
org-columns-quit	63	org-feed-goto-inbox	86
org-columns-redo	63	org-feed-update-all	86
org-columns-set-tags-or-toggle	64	org-force-cycle-archived	89
org-columns-show-value	64	org-forward-same-level	9
org-columns-widen	64	org-global-cycle	8
org-compute-property-at-point	59	org-goto	9
org-copy-subtree	10	org-goto-calendar	67
org-cut-subtree	10	org-insert-columns-dblock	65
org-cycle	7, 10	org-insert-export-options-template	125
org-cycle	14	org-insert-heading	9
org-cycle-agenda-files	91	org-insert-heading	14, 79
org-date-from-calendar	67	org-insert-heading-respect-content	9
org-dblock-update	65, 74, 193	org-insert-link	37
org-deadline	71	org-insert-property-drawer	59, 197
org-delete-property	59	org-insert-todo-heading	10
org-delete-property-globally	59	org-insert-todo-heading	43, 53
org-demote	199	org-insert-todo-heading-respect-content	10
org-demote-subtree	10	org-map-entries	198
org-do-demote	10	org-mark-entry-for-agenda-action	71
org-do-promote	10	org-mark-ring-goto	39
org-edit-special	184	org-mark-ring-push	39
org-entry-add-to-multivalued-property	197	org-match-sparse-tree	57
org-entry-delete	197	org-move-subtree-down	10
org-entry-get	197	org-move-subtree-up	10
org-entry-get-multivalued-property	197	org-narrow-to-block	11
org-entry-member-in-multivalued-property	197	org-narrow-to-subtree	11
org-entry-properties	197	org-next-link	39
org-entry-put	197	org-occur	12
org-entry-put-multivalued-property	197	org-open-at-point	38
org-entry-remove-from-multivalued-property	197	org-open-at-point	67
org-evaluate-time-range	67, 74	org-paste-subtree	10
org-export	127	org-previous-link	39
org-export-as-ascii	127	org-priority	50, 199
		org-priority-down	50

org-priority-up.....	50	org-table-insert-row.....	20
org-promote.....	199	org-table-iterate.....	31
org-promote-subtree.....	10	org-table-iterate-buffer-tables.....	31
org-property-action.....	59	org-table-kill-row.....	20
org-property-next-allowed-value.....	59	org-table-move-column-left.....	20
org-property-previous-allowed-value.....	59	org-table-move-column-right.....	20
org-publish.....	150	org-table-move-row-down.....	20
org-publish-all.....	150	org-table-move-row-up.....	20
org-publish-current-file.....	150	org-table-next-field.....	19
org-publish-current-project.....	150	org-table-next-row.....	20
org-refile.....	11, 87	org-table-paste-rectangle.....	21
org-refile-cache-clear.....	87	org-table-previous-field.....	20
org-refile-goto-last-stored.....	87	org-table-recalculate.....	31
org-remove-file.....	91	org-table-recalculate-buffer-tables.....	31
org-reveal.....	8	org-table-rotate-recalc-marks.....	31
org-schedule.....	71	org-table-sort-lines.....	20
org-search-view.....	98	org-table-sum.....	21
org-set-effort.....	78	org-table-toggle-coordinate-overlays..	29, 30
org-set-property.....	59	org-table-toggle-formula-debugger.....	29
org-set-startup-visibility.....	8	org-table-wrap-region.....	21
org-set-tags-command.....	54	org-tags-view.....	57, 60, 95
org-show-todo-key.....	42	org-time-stamp.....	67
org-sort-entries-or-items.....	11	org-time-stamp-inactive.....	67
org-sparse-tree.....	12	org-timeline.....	98
org-speedbar-set-agenda-restriction.....	91	org-timer.....	78
org-store-agenda-views.....	114	org-timer-item.....	78
org-store-link.....	36	org-timer-start.....	79
org-table-align.....	19	org-timestamp-down-day.....	67
org-table-beginning-of-field.....	20	org-timestamp-down-down.....	67
org-table-copy-down.....	21	org-timestamp-up.....	67
org-table-copy-region.....	21	org-timestamp-up-day.....	67
org-table-create-or-convert-from-region..	19	org-todo.....	42
org-table-create-or-convert-from-region..	22	org-todo.....	74
org-table-create-with-table.el.....	184	org-todo.....	199
org-table-cut-region.....	21	org-todo-list.....	43, 94
org-table-delete-column.....	20	org-toggle-archive-tag.....	89
org-table-edit-field.....	21	org-toggle-checkbox.....	52
org-table-edit-formulas.....	29	org-toggle-heading.....	11
org-table-end-of-field.....	20	org-toggle-inline-images.....	38
org-table-eval-formula.....	28	org-toggle-ordered-property.....	46, 53
org-table-eval-formula.....	29	org-toggle-tag.....	199
org-table-export.....	22	org-toggle-time-stamp-overlays.....	69
org-table-fedit-abort.....	30	org-tree-to-indirect-buffer.....	8
org-table-fedit-finish.....	30	org-update-all-dbblocks.....	65
org-table-fedit-line-down.....	30	org-update-statistics-cookies.....	53
org-table-fedit-line-up.....	30	org-write-agenda.....	109
org-table-fedit-lisp-indent.....	30	org-write-agenda.....	113
org-table-fedit-ref-down.....	30	org-yank.....	10
org-table-fedit-ref-left.....	30	outline-next-visible-heading.....	9
org-table-fedit-ref-right.....	30	outline-previous-visible-heading.....	9
org-table-fedit-ref-up.....	30	outline-up-heading.....	9
org-table-fedit-scroll-down.....	30		
org-table-fedit-scroll-up.....	30		
org-table-fedit-toggle-ref-type.....	30		
org-table-field-info.....	29		
org-table-hline-and-move.....	20		
org-table-insert-column.....	20		
org-table-insert-hline.....	20		

P

pcomplete..... 59

previous-error..... 12

S

`show-all` 8
`show-branches` 8

W

`widen` 11

Variable index

This is not a complete index of variables and faces, only the ones that are mentioned in the manual. For a more complete list, use *M-x org-customize RET* and then click yourself through the tree.

C

cdlatex-simplify-sub-super-scripts..... 123
 constants-unit-system..... 26, 179

H

htmlize-output-type..... 113

L

~~TeX~~-verbatim-environments..... 190

O

org-adapt-indentation..... 181
 org-agenda-add-entry-text-maxlines..... 113
 org-agenda-columns-add-appointments-to-effort-sum..... 78
 org-agenda-confirm-kill..... 105
 org-agenda-custom-commands .. 12, 110, 111, 112, 195
 org-agenda-diary-file..... 109
 org-agenda-dim-blocked-tasks..... 46
 org-agenda-entry-text-maxlines..... 103
 org-agenda-exporter-settings..... 109, 113
 org-agenda-files..... 90, 100, 143
 org-agenda-filter-preset..... 104
 org-agenda-log-mode-items..... 102
 org-agenda-ndays..... 93
 org-agenda-overriding-header..... 194
 org-agenda-prefix-format..... 99
 org-agenda-restore-windows-after-quit.... 90
 org-agenda-show-inherited-tags..... 106
 org-agenda-skip-archived-trees..... 89
 org-agenda-skip-function..... 193, 194, 198
 org-agenda-skip-function-global..... 193
 org-agenda-skip-scheduled-if-done..... 70
 org-agenda-sorting-strategy..... 101
 org-agenda-span..... 93, 102
 org-agenda-start-with-clockreport-mode.. 103
 org-agenda-start-with-entry-text-mode... 103
 org-agenda-start-with-follow-mode..... 101
 org-agenda-tags-column..... 99
 org-agenda-tags-todo-honor-ignore-options..... 95
 org-agenda-text-search-extra-files.... 92, 98
 org-agenda-time-grid..... 100, 103
 org-agenda-todo-ignore-deadlines..... 95
 org-agenda-todo-ignore-scheduled..... 95
 org-agenda-todo-ignore-timestamp..... 95

org-agenda-todo-ignore-with-date..... 95
 org-agenda-todo-list-sublevels..... 51, 95
 org-agenda-use-time-grid..... 100, 103
 org-agenda-window-setup..... 90
 org-alphabetical-lists..... 13
 org-archive-default-command..... 88, 106
 org-archive-location..... 88, 176
 org-archive-save-context-info..... 88
 org-attach-directory..... 85
 org-attach-method..... 85
 org-babel-default-header-args..... 157, 158
 org-calc-default-modes..... 26
 org-clock-idle-time..... 76
 org-clock-into-drawer..... 73
 org-clock-modeline-total..... 73
 org-clocktable-defaults..... 75
 org-coderef-label-format..... 119
 org-columns-default-format.... 63, 78, 103, 115
 org-columns-skip-archived-trees..... 89
 org-combined-agenda-icalendar-file..... 143
 org-confirm-babel-evaluate..... 176
 org-confirm-elisp-link-function..... 176
 org-confirm-shell-link-function..... 176
 org-create-file-search-functions..... 41
 org-ctrl-c-ctrl-c-hook..... 188
 org-ctrl-k-protect-subtree..... 7
 org-cycle-emulate-tab..... 7
 org-cycle-global-at-bob..... 7
 org-cycle-include-plain-lists..... 14
 org-cycle-open-archived-trees..... 89
 org-cycle-separator-lines..... 7
 org-deadline-warning-days..... 70, 71
 org-default-notes-file..... 80, 82
 org-default-priority..... 50, 177
 org-display-custom-times..... 69, 146
 org-display-internal-link-with-indirect-buffer..... 38
 org-disputed-keys..... 184
 org-done (フェイス)..... 45
 org-drawers..... 16, 177
 org-effort-property..... 77
 org-empty-line-terminates-plain-lists.... 13
 org-enable-table-editor..... 19
 org-enforce-todo-dependencies..... 46
 org-entities..... 121, 140
 org-execute-file-search-functions..... 41
 org-export-ascii-links-to-notes..... 128
 org-export-author-info..... 146
 org-export-creator-info..... 146
 org-export-default-language..... 125, 146

- org-export-docbook-default-image-attributes 139
- org-export-docbook-doctype 140
- org-export-docbook-inline-image-extensions 140
- org-export-docbook-xsl-fo-proc-command.. 138
- org-export-docbook-xslt-proc-command.... 138
- org-export-docbook-xslt-stylesheet..... 138
- org-export-email..... 146
- org-export-exclude-tags..... 125, 146
- org-export-headline-levels 116, 146
- org-export-highlight-first-table-line ... 146
- org-export-html-expand..... 146
- org-export-html-extension..... 146
- org-export-html-extra..... 132
- org-export-html-inline-images..... 130, 146
- org-export-html-link-home..... 146
- org-export-html-link-org-files-as-html.. 146
- org-export-html-link-up 146
- org-export-html-postamble..... 146
- org-export-html-preamble..... 146
- org-export-html-style..... 132, 146
- org-export-html-style-default..... 132
- org-export-html-style-extra..... 146
- org-export-html-style-include-default... 132, 146
- org-export-html-style-include-scripts ... 146
- org-export-html-table-tag 130, 146
- org-export-html-tag-class-prefix..... 131
- org-export-html-todo-kwd-class-prefix ... 131
- org-export-html-use-infojs 133
- org-export-html-with-timestamp 146
- org-export-latex-classes 134
- org-export-latex-default-class 134
- org-export-latex-default-packages-alist 134
- org-export-latex-packages-alist 134
- org-export-preserve-breaks 146
- org-export-publishing-directory 146
- org-export-run-in-background..... 127
- org-export-section-number-format 146
- org-export-select-tags..... 125, 146
- org-export-skip-text-before-1st-heading 116, 146
- org-export-taskjuggler-default-reports.. 142
- org-export-taskjuggler-project-tag..... 141
- org-export-taskjuggler-resource-tag..... 141
- org-export-with-archived-trees..... 89, 146
- org-export-with-drawers 146
- org-export-with-emphasize..... 146
- org-export-with-fixed-width..... 146
- org-export-with-footnotes..... 146
- org-export-with-LaTeX-fragments..... 122, 146
- org-export-with-priority 146
- org-export-with-section-numbers 146
- org-export-with-special-strings 146
- org-export-with-sub-superscripts 121, 146
- org-export-with-tables..... 146
- org-export-with-tags..... 146
- org-export-with-Tex-macros 146
- org-export-with-timestamps 146
- org-export-with-toc 116, 146
- org-export-with-todo-keywords..... 146
- org-fast-tag-selection-include-todo..... 45
- org-fast-tag-selection-single-key..... 56
- org-file-apps 38, 85
- org-footnote-auto-adjust..... 17, 179
- org-footnote-auto-label..... 17, 179
- org-footnote-define-inline 17, 179
- org-footnote-section 17
- org-format-latex-header 122
- org-format-latex-options..... 122, 123
- org-from-is-user-regexp..... 84
- org-global-properties 58, 78
- org-goto-auto-isearch..... 9
- org-goto-interface 9
- org-hide (face) 181
- org-hide-block-startup..... 16
- org-hide-leading-stars..... 178, 181
- org-hierarchical-checkbox-statistics..... 52
- org-hierarchical-todo-statistics 51
- org-highest-priority 50, 177
- org-icalendar-alarm-time 143
- org-icalendar-categories 143
- org-icalendar-include-body 143
- org-icalendar-include-todo 143
- org-icalendar-store-UID 143
- org-icalendar-use-deadline 143
- org-icalendar-use-scheduled..... 143
- org-imenu-depth..... 183
- org-infojs-options 133
- org-insert-mode-line-in-empty-file..... 4
- org-irc-link-to-logs..... 37
- org-keep-stored-link-after-insertion..... 37
- org-latex-low-levels 134
- org-link-abbrev-alist 39, 177
- org-link-to-org-use-id..... 36
- org-list-automatic-rules..... 14, 15, 51
- org-list-demote-modify-bullet..... 14
- org-list-end-regexp..... 13
- org-list-ending-method..... 13
- org-log-done..... 48, 102, 178
- org-log-into-drawer 47, 106
- org-log-note-clock-out 73, 178
- org-log-refile..... 87
- org-log-repeat 72, 178
- org-log-states-order-reversed..... 47
- org-lowest-priority 50, 177
- org-M-RET-may-split-line..... 9, 14
- org-odd-levels-only..... 96, 178, 182, 194
- org-outline-path-complete-in-steps..... 87
- org-overriding-columns-format..... 115
- org-plain-list-ordered-item-terminator... 13, 15
- org-popup-calendar-for-date-prompt..... 69
- org-priority-faces 50

org-priority-start-cycle-with-default..... 50
 org-property-allowed-value-functions..... 197
 org-publish-project-alist 144, 147
 org-publish-use-timestamps-flag 150
 org-put-time-stamp-overlays 179
 org-read-date-display-live 69
 org-read-date-prefer-future 68
 org-refile-allow-creating-parent-nodes ... 87
 org-refile-targets 87
 org-refile-use-cache 87
 org-refile-use-outline-path 87
 org-remove-highlights-with-change 12, 74
 org-replace-disputed-keys 184
 org-return-follows-link 38
 org-reverse-note-order 87
 org-show-entry-below 12
 org-show-following-heading 12
 org-show-hierarchy-above 12
 org-show-siblings 12
 org-sort-agenda-noeffort-is-high 104
 org-sparse-tree-open-archived-trees 89
 org-special-ctrl-a/e 7
 org-special-ctrl-k 7
 org-speed-commands-user 175
 org-startup-align-all-tables 23, 178
 org-startup-folded 8, 177
 org-startup-indented 177
 org-startup-with-inline-images 39, 178
 org-store-link-functions 187
 org-stuck-projects 98
 org-support-shift-select 15, 184
 org-table-auto-blank-field 19
 org-table-copy-increment 21
 org-table-export-default-format 22
 org-table-formula 177
 org-table-formula-constants 26, 177, 183
 org-table-use-standard-references 24, 29

org-tag-alist 55, 179
 org-tag-faces 54
 org-tag-persistent-alist 55
 org-tags-column 54
 org-tags-exclude-from-inheritance 54
 org-tags-match-list-sublevels... 54, 57, 60, 95
 org-time-stamp-custom-formats 69
 org-time-stamp-overlay-formats 179
 org-time-stamp-rounding-minutes 67
 org-todo (フェイス) 45
 org-todo-keyword-faces 45
 org-todo-keywords 42, 43, 94, 180
 org-todo-repeat-to-state 71
 org-todo-state-tags-triggers 43
 org-track-ordered-property-with-tag... 46, 53
 org-treat-insert-todo-heading-as-state-
 change 10
 org-treat-S-cursor-todo-selection-as-state-
 change 42
 org-use-property-inheritance 60, 143, 197
 org-use-speed-commands 175
 org-use-tag-inheritance 54
 org-yank-adjusted-subtrees 10
 org-yank-folded-subtrees 10

P

parse-time-months 68
 parse-time-weekdays 68
 ps-landscape-mode 113
 ps-number-of-columns 113

U

user-full-name 125, 146
 user-mail-address 125, 146