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。

目次

1 Introduction

1.1 Summary

Org-mode は、高速かつ効率的に動作するプレーンテキストのシステムを用いて、ノートを保存したり、TODO リストを管理したり、プロジェクトを計画するための Emacs モードです。

プレーンなテキスト形式でリストやプロジェクトに関する情報を含む分散したノートから、Org-mode は組織的に結びついたタスク群を生成します。Org-mode はアウトラインモードを元に実装されているます。そのため、大きなファイルの内容をわかりやすく構造化した状態に維持できます。また、必要な部分だけを表示する機能と文書構造の編集機能がツリー形式の文書編集を手助けします。ビルトインされたテーブルエディタで簡単に表を作成できます。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。プレーンテキストで記述される 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とも結びついています。ノートを印刷したりノートを共有するために、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 形式のファイルへ Org-mode のファイルをエクスポートできます。また、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。

見出しとなるノードにメタデータを追加することで、Org-mode はプロジェクトを計画する環境となります。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* を生成できます。

Org-mode は Org-Babel 環境を含んでいて、次のようなことが可能になります。すなわち、ソースコードブロックが組み込まれたファイルで作業でき、コードを評価、文書化、そして、文芸的プログラミングを実践できます。

Org-mode の自動的かつ文脈を読み取る表編集機能は、表計算ソフトと互換性があり、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めます。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。Org-mode に複雑さはなく、それでいて、必要に応じて数多くの機能を利用できます。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、具体的には以下のようなものです。

- an outline extension with visibility cycling and structure editing
- an ASCII system and table editor for taking structured notes
- a TODO list editor
- a full agenda and planner with deadlines and work scheduling
- 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

最新バージョンへのリンクがある、Org-mode のためのウェブサイトがあります。関連情報、よくある質問 (FAQ)、または、チュートリアルなどへのリンクも集約されています。`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

重要: もしあなたが、*Emacs* の一部として組み込まれた *Org-mode* 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 *1.3*節「アクティベーション」*p.4* に移動してください。あなたの *Emacs* に含まれている *Org-mode* (存在する場合) 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。

すでにインターネットから Org-mode をダウンロードしているならば、‘.zip’か‘.tar’ もしくは Git アーカイブかを問いませんが、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-mode 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 バイナ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。システムディレクトリへのアクセス権を持っていないならば、‘lisp’（訳注：解凍したディレクトリの直下にあるサブディレクトリ）を Emacs のロードパスに加えることで、配布された Org-mode のディレクトリを使って、簡単に Org-mode を動かせます。このようにするためには、‘.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))
```

そして、シェルコマンドを使って Lisp ファイルをバイトコンパイルします。

```
make
```

解凍した Org-mode ファイルがあるディレクトリで Org-mode を動かすならば、これでインストール完了です。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ならば、管理者権限で次のコマンドを実行します。

```
make install
```

INFO ファイルのインストール方法はシステムに依存します。これは、‘install-info’ プログラムの違いに原因があります。Debian であれば、‘install-info’が INFO ファイルを正しいディレクトリにコピーし、INFO ディレクトリファイル（訳注：dir のこと）を変更します。その他の多くのシステムでは、各ファイルが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。システムのドキュメントを読んで、次のどちらのコマンドを必要とするかを調査してください。

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

最後に、以下の一行を‘.emacs’に追加します。これは、Org-mode の開始時には読み込まれないファイルにある関数を、Emacs が自動的に読み込むために必要です。

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

次節に解説されている Org-mode のアクティベーションを忘れずに実行してください。

1.3 アクティベーション

拡張子が‘.org’のファイルで必ず 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 のバッファは、フォントロックが有効になっている必要があります。これは Emacs の標準の設定です[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（つまり、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)
```

ここまでの設定を用いると、拡張子が‘.org’のファイルのすべてが 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; -*-
```

この設定が書き込まれたバッファは、ファイルの名前とは無関係に Org-mode が有効になります。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、リージョンがアクティブならば、そのリージョンに対して動作します。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 では次のように自分で設定する必要があります。

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

もし transient-mark-mode が好みでないならば、マウスを利用した領域の選択で、アクティブなリージョンを指定できます。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、投稿した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

[1] もしグローバルにフォントロックを使わない場合は、(add-hook 'org-mode-hook 'turn-on-font-lock) を使って、Org-mode のバッファのフォントロックを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。

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することです。

M-x org-submit-bug-report

この関数を実行すると、Emacs のメールバッファに必要なテンプレートを書きこんでくれるので、バグの説明だけを書き加えればよい状態になります。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 Org-mode を使っていてエラーが発生したら、バックトレースがとても役立ちます（作り方は次節を参照してください）。しばしば例となる小さなファイルが問題解決の手助けになります。それらには次のような明瞭な情報を含んでいます。

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

Org-mode の改善にご協力いただき感謝します。

有用なバックトレースを生成する方法

Org-mode を利用していて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を含みます。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 ファイル群を再度読み込みます。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。そのために、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`を選択します（XEmacs では、このオプションは`Troubleshooting`サブメニューにあります）。
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ルール

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、大文字で始めて残りが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（これはコマンドを利用する状況に依存しています)。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できます。

2 ドキュメントの構造

Org-mode は、Outline mode をベースとしており、ドキュメントの構造を編集するためにフレキシブルなコマンドを用意しています。

2.1 Outlines

Org-mode は outline mode の上で実行されます。アウトラインによって階層構造で体系化されたドキュメントが作られ、(少なくとも私にとっては) それによって、ノートや思考の最高の表現方法となります。ドキュメントの大きな部分を折りたたむ (隠す) ことによって、ドキュメントの骨格のみを表示したり、現在、作業している部分を表示したりして、ドキュメントの構造の全体を見渡すことができるのです。Org-mode は、全体を表示したり／隠したりする機能を、たったひとつのコマンド、`org-cycle`、それは `TAB`キーと結びついていますが、に圧縮することにより、アウトラインの使用を大変単純なものにしています。

2.2 Headlines

見出しは、アウトラインのツリーの構造を定義します。Org-mode の見出しは、左のマージン[1] 上にある 1 つもしくはそれ以上の数の「`*`」で始まります。例えば。

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

* Another top level headline
```

たくさんの「`*`」があるとうるさく感じ、空白のあとに、見出しの始まりとしてのひとつの「`*`」があるという形式のアウトラインを好む人もいるでしょう。このような形式の設定について、15.8 節「Clean view」p.189, で説明しています。

最後のサブツリーの直後の空白行は、そのサブツリーの一部と見なされます。そのためサブツリーが折り畳まれたときには、隠れてしまいます。しかしながら、すくなくとも 2 行の空白行を残したときは、折り畳んだビューを構造化するために、サブツリーを折り畳んだあとも、1 つの空白行は残ったままになります。この動作を修正したいときは、`org-cycle-separator-lines`を参照してください。

2.3 Visibility cycling

アウトラインによって、バッファの中で、テキストの一部を隠すことが可能となります。Org-mode はバッファ内での表示の状況を変更するために、`TAB`と *`S-TAB`*とに結びついた 2 つのコマンドを使用します。

`TAB` `org-cycle`

Subtree cycling：カレントのサブツリーの状態を順番に表示します。

[1] 見出しの中で、*`C-a`*、*`C-e`*および *`C-k`*の特別な作用を設定するために、`org-special-ctrl-a/e`、`org-special-ctrl-k`、および `org-ctrl-k-protect-subtree`の変数を参照してください。

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

これを動作[2] させるためにはカーソルが見出しの上に置かれている必要があります。カーソルがバッファの一番上の行にあり、そして最初の行が見出しでない場合は、TABが実際にグローバルな切り替えが実行されます。(下記を参照) [3] 前置引数 (*C-u TAB*) をつけて呼び出したときは、グローバルな切替が実行されます。

S-TAB org-global-cycle

C-u TAB　*Global cycling*：バッファ全体を交代で状態を変更する。

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

*S-TAB*が N という数字のついた前置引数と一緒に呼び出されたときは、レベル N 以上の見出しが CONTENTS ビューに表示されます。テーブルの中では、*S-TAB* は前のフィールドにジャンプするということに注意してください。

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

全てを表示する。引き出しを含む。

C-c C-r org-reveal

カレントエントリーや、下の見出しや上の階層を表示して、その場所でのコンテクストを表示する。ツリーの抽出コマンド（2.6 節「Sparse trees」p.12 を参照）やアジェンダのコマンド（10.5 節「Agenda commands」p.107 を参照）によって表示された場所の周辺で作業をするのに役立ちます。前置引数をつけることで、各階層での同一レベルの見出しを表示する。前置引数を 2 重に使った場合は、親のサブツリー全体を表示する。

C-c C-k show-branches

サブツリーの見出しを全て表示し、ひとつのサブツリーのためのコンテンツビューである。

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

間接的なバッファ[4]. の中にあるカレントのサブツリーを表示する。N という数値付きの前置引数をつけると、N 段階上の階層に上がるがそのツリーを捉える。もしも N がマイナスの値ならば、多くの階層まで遡る。*C-u*の前置引数をつけたならば、それ以前に使用された間接的なバッファを削除してはならない。

Emacs である Org-mode ファイルを最初に開いたときに、グローバルな状態としては、概観のビューで開くように設定されています。すなわち、最上位の階層の見出しのみが表示されています。これは、org-startup-folded変数によって設定されています。つまり、以下に示す行をバッファ上のどこかに追加することによって、ファイル毎に設定することができます。

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

[2] しかしながら、org-cycle-emulate-tabオプションを参照してください。

[3] org-cycle-global-at-bobオプション参照。

[4] 間接的なバッファとは、(see the Emacs manual for more information about indirect buffers) は全てのバッファを含んでいるが、カレントのツリーに制限されるだろう。間接的なバッファを編集することは、オリジナルのバッファに変更を加えることでもある。だがそのバッファの中での表示に影響を与えることはできない。

さらに、どのエントリーも ‘VISIBILITY’属性（第 7 章「Properties and Columns」p.61 を参照）を持っており、それを受けて適用された表示性をしめすでしょう。この属性のために許されている値は、folded、children、contentおよび allです。

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

そのバッファにおける起動時の表示条件に戻ります。すなわち、起動時のオプションで要求されている内容、そして個々のエントリーの中で設定されている ‘VISIBILITY’の属性に。

2.4 Motion

以下のコマンドはバッファの中で他の見出しにジャンプするものです。

C-c C-n outline-next-visible-heading

次の見出しへ。

C-c C-p outline-previous-visible-heading

前の見出しへ。

C-c C-f org-forward-same-level

次の同一階層の見出しへ。

C-c C-b org-backward-same-level

前の同一階層の見出しへ。

C-c C-u outline-up-heading

一つ上の階層の見出しに戻る。

C-c C-j org-goto

現在のアウトラインの表示状態を変更することなく、別の場所にジャンプする。現在のバッファの中で文書の構造を表示し、そこではあなたの目的の場所を見つけるために以下のようなキーを使用することができます。

TAB	表示を切り替える。
down / up	次の／前の表示されている見出しへ。
RET	この場所を選択する。
/	ツリーの抽出による検索を実行する

もしも org-goto-auto-isearch を停止したときには以下のキーが動作する

n / p	次の／前の表示されている見出しへ。
f / b	次の／前の同じ階層の見出しへ。
u	ひとつ上の階層へ。
0-9	数値の変数。
q	停止

org-goto-interface変数もまた参照のこと。

2.5 Structure editing

M-RET org-insert-heading

カレントの階層と同じ階層の新しい見出しを挿入します。もしもカーソルがプレーンなリストアイテムの中にあるならば、新しいアイテムが作成されます (2.7 節「Plain lists」p.13 を参照)。新しい見出しを強制的に作成するには前置引数を

つけます。このコマンドが行の途中で使用されたときは、その行が分割され、その行の残りの部分が新しい見出し[5] となります。もしも見出しの先頭でそのコマンドが使用されたときは、カレント行の前に新しい見出しが作られます。もしも見出し以外の行の先頭の場合は、その行の内容が新しい見出しとして作成されます。そのコマンドが折り畳まれているサブツリーの行末で使用されたならば (i.e. 見出しの最後の楕円の後)、カレントの見出しと同様な見出しが、サブツリーの末尾の後に挿入されるでしょう。

C-RET org-insert-heading-respect-content

*M-RET*とちょうど同じように、カレントの見出しの下に新しい見出しが付け加えられたときを除いて、新しい見出しは本文の前に置かれるかわりに、本文の後に置かれます。このコマンドはエントリーの中のどの場所からでも動作します。

M-S-RET org-insert-todo-heading

カレントの見出しと同じ階層の新しい TODO エントリーが挿入されます。org-treat-insert-todo-heading-as-state-change変数も同じように参照してください。

C-S-RET org-insert-todo-heading-respect-content

カレントの見出しと同一の階層に新しい TODO エントリーを挿入します。*C-RET* と同様に、新しい見出しはカレントのサブツリーの後に挿入されるでしょう。

TAB org-cycle

新しいエントリーでまだ文が書かれていない状態で、最初に TABを実行すると、そのエントリーの階層を下げ、その前の見出しの子になります。次に TABを実行すると、その見出しを親として、それによってトップの階層まで、作成します。さらに次の TABで、初期の階層にもどります。

M-left org-do-promote

カレントの見出しを 1 階層上げる。

M-right org-do-demote

カレントの見出しを 1 階層下げる。

M-S-left org-promote-subtree

カレントのサブツリーを 1 階層上げる。

M-S-right org-demote-subtree

カレントのサブツリーを 1 階層下げる。

M-S-up org-move-subtree-up

サブツリーを上に移動する。(同じ階層の前のサブツリーと交換する。)

M-S-down org-move-subtree-down

サブツリーを下に移動する。(同一階層の次のサブツリーと交換する。)

C-c C-x C-w org-cut-subtree

サブツリーをキルする。i.e. そのサブツリーをバッファから取り除くが、キルリングに保存する。N という数字付きの前置引数をつけたときは、N 個連続でサブツリーをキルする。

---

[5] もしも行を途中で分割したくないときは、org-M-RET-may-split-line変数をカスタマイズしてください。

C-c C-x M-w org-copy-subtree
サブツリーをキルリングにコピーする。N という数字付きの前置引数をつけたときは、N 個連続でサブツリーをコピーする。

C-c C-x C-y org-paste-subtree
キルリングからサブツリーを貼り付ける。これによると、貼り付けるポジションにうまく合わせて、ツリーに適合するようにサブツリーの階層を調整する。数字付きの前置引数をつけるか、‘****’のような星印のついた見出しの後に貼り付けることによって、貼り付ける階層を指定することができる。

C-y org-yank
org-yank-adjusted-subtreesと org-yank-folded-subtreesという変数によって、Org-mode の内部の yankコマンドは、賢い方法で、*C-c C-x C-y*と同等のコマンドを用いて、折り畳まれているサブツリーを貼り付けることができるでしょう。デフォルトの設定では、階層の調整は行われませんが、貼り付けられたツリーは、既に表示されているテキスト受け入れない限り、折り畳まれたままでしょう。このコマンドに対して何らかの前置引数をつけることで、渡されたプレフィックスに応じて、通常の yankを実行させることになります。通常の yank を実行する良い方法は *C-u C-y*です。yank の後で yank-popを使うと、階層の調整や折り畳みをすることなく、それ以前に kill したアイテムをプレーンに yank します。

C-c C-x c org-clone-subtree-with-time-shift
たくさんのそれと同じ兄弟のコピーを作成することで、サブツリーの複製を作ります。たくさんのコピーの作成を実行したいならば、そのエントリーに含まれているタイムスタンプも調整されるように指定することもできます。この機能は便利です。例えば、準備している一連の講義に関連した沢山のタスクを作成するという場合のように。もっと詳細な情報が必要ならば、org-clone-subtree-with-time-shiftコマンドの解説を参照してください。

C-c C-w org-refile
エントリーやリージョンを別の場所に保管します。9.5 節「Refiling notes」p.92 を参照してください.

C-c ^ org-sort-entries-or-items
同じ階層のエントリーを並び替えられます。アクティブなリージョンがあるときに、そのリージョンにあるすべてのエントリーは順番に並びます。もう一方で、カレントの見出しの子供の階層も並び替えられます。並び替えの形式をコマンドで入力します。すなわちアルファベット順、数字順、時間順（実行するために参照される作成日、予定日、期限などの最初のタイムスタンプ）、優先順位順、TODO キーワード順（設定の中で定義された一連のキーワードの中で）あるいは属性の価値の順に並べ替えるために。並び順を反転することも同様に可能です。並び替えのキーを拡張するために自分自身の関数を用意することもできます。*C-u C-u* という二重の前置引数を使用すると、複製されたエントリーは削除されます。

C-x n s org-narrow-to-subtree
カレントのサブツリーのためにバッファをナローイングします。

C-x n b org-narrow-to-block
カレントのブロックのためにバッファをナローイングします。

C-x n w widen
ナローイングを取り除きバッファを広げます。

*C-c ** org-toggle-heading
普通の行やプレーンなリストアイテムを見出しに変更します。(そのため、それらの場所によってはサブの見出しになります。) 星汁うぃを取り除くことによって見出しを普通の行に変更することもできます。もしもアクティブなリージョンあるならば、その領域のすべての行が見出しに変更されます。もしもその領域の中の最初の行がアイテムだったら、そのアイテムの行のみが見出しに変更されます。最後に、もし最初の行が見出しならば、その領域の中の全ての見出しから星印が取り除かれます。

アクティブなリージョンがあるときには (Transient Mark mode)、そのリージョンのすべての見出しの階層を上げたり、下げたり作用することができる。あるリージョンの見出しを選択するためには、行の先頭にポイントを置いてマークし、最初の見出しの先頭でマークし、変更する最後の見出しの次の行にポイントを置くのが良い方法である。カーソルがテーブル (第 3 章「Tables」p.20 を参照) の中にあるときに、Meta-Cursor キーは異なる機能性を持つことに注意してください。

2.6 Sparse trees

Org-mode の重要な特徴の一つに、あるアウトラインのツリーに含まれている選択された情報のために *sparse trees* (ツリーの抽出) を作ることができるということがあります。そのため文書全体が最大限畳まれていても、その[6] 上に見出し構造に沿って表示することができるのです。試してみて、それがどんなに素早く動作するかを見てください。

Org-mode にはそういうツリーを作成するためのいくつものコマンドがあります。これらのコマンドの全てはディスパッチャーを通してアクセスすることができます。

C-c / org-sparse-tree
これは、ツリーの抽出を選択するためのコマンドを作成する追加のキーを入力する。

C-c / r org-occur
発生。正規表現のための入力と全ての一致したものについてのツリーの抽出を表示する。もしもその一致した言葉が見出しの中にあるならば、その見出しが表示される。もしもその一致した言葉がエントリーの本文の中にあるならば、見出しと本文が表示される。最小の内容を区分するために、その一致した言葉のある見出しの階層全体が表示され、同様にその一致した言葉に続く見出しも表示される。どの一致した言葉もハイライトされる。そのハイライトはバッファが編集コマンド[7]　によって変更されるか、*C-c C-c*を押すことで消える。*C-u*前置引数が呼ばれたときは、以前のハイライトは維持される。そのため何度もこのコマンドを呼び出すと積み重ねることができる。

M-g n or *M-g M-n* next-error
そのバッファの中の次のツリーの抽出部分にジャンプする。

---

[6] 検索に一致したときに、どの範囲の内容を表示するかを詳細にコントロールするために、org-show-hierarchy-aboveorg-show-following-heading、org-show-siblings 、そして org-show-entry-below変数を参照のこと

[7] これは org-remove-highlights-with-changeオプションに依存する。

M-g p or *M-g M-p* previous-error
そのバッファの前のツリーの抽出部分にジャンプする。

特定の検索文字列によるツリーの抽出を何度も使用するために、org-agenda-custom-commands変数を使って特定のツリーの抽出に、素早くキーボードからアクセスする定義をすることができる。これらのコマンドはアジェンダディスパッチャー (10.2 節「Agenda dispatcher」p.97 を参照) を通してアクセスすることができる。例えば。

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

'FIXME'という文字列にマッチするツリーの抽出するためのショートカットとして、*C-c a f*を定義します。

他のツリーの抽出のためのコマンドは、TODO キーワード、タグ、あるいは属性に基づいて見出しを選択するもので、このマニュアルの後の部分で議論されるだろう。

抽出したツリーを印刷するためには、Emacs の ps-print-buffer-with-facesというコマンドを使用することができます。それを使うと文書[8] のうちの表示されていない部分は印刷されません。あるいは、文書の見えている部分をエクスポートするために、*C-c C-e v*コマンドを使用し、エクスポートしたファイルを印刷することができます。

2.7 Plain lists

アウトラインのエントリーの中に、手動でフォーマットしたリストによって、別の構造化された項目を追加することができます。そのリストを使って、チェックボックス (5.6 節「Checkboxes」p.54 を参照) のリストを作成する方法が提供されています。Org-mode ではそういうリストの編集をサポートしており、そしてすべてのエクスポート機能 (第 12 章「Exporting」p.132 を参照) はそれらのリストの構文を解析しフォーマット化することができます。

Org-mode では、数字付きのリスト、順序のないリスト、そして記述リストを解釈します。

- 順序のないリストアイテムは、'-'、'+'、または箇条書きの太い中黒としての '*'[9] が文頭に付きます。
- 順番のあるリストアイテムは、数字のあとにピリオドか右括弧[10] がついた形ではじまっています。例えば、org-alphabetical-listsを設定することによって、'1.'や'1)'[11] のように。もしもあなたがリストをこれら以外の値 (e.g. 20) で始めたいと思ったら、そのアイテムの最初の文字を [@20][12] のような文字で始めます。
- 説明のリストアイテムは順序のないリストアイテムで、説明内容と用語の記述を区別するために ' :: 'といった区分するための記号を含んでいます。

---

[8] このコマンドは、XEmacs では動作しません。というのは、XEmacs では、テキスト属性の部分ではなく、アウトラインの選択して表示するために使用するものだからです。

[9] '*'を箇条書きの太い中黒として使用するときは、それらの行はインデントが設定されている必要があります。そうでなければ、それらの行は見出しのトップ階層と見なされてしまいます。また、わかりやすいアウトラインビューを得るために、先頭の星印を隠しているときは、プレーンなリストアイテムの場合は、本当の見出しと区別が付かなくなります。簡単に言えば、'*'をサポートしているもののプレーンなリストアイテムのためには使用しない方が良いかもしれません。

[10] org-plain-list-ordered-item-terminatorの設定によって、それらのリストをはずすことが可能です。

[11] 'a.'、'A.'、あるいは'a)'といった形式も可能です。通常のテキストの混乱を最小限にするために、1 つの文字のみに限定されています。この制限を超えると、数字に替えて bullet が自動的に使用されます。

[12] そのアイテムの中にチェックボックスがある場合は、そのクッキーは、チェックボックスの前に置かれなければなりません。もしも活性化されたアルファベットのついたリストがあるならば、[@b]といったカウンターを使用することもできます。

同じリストに属しているアイテムは、最初の行と同じインデントでなければならない。特に、もしも順番のついたリストが‘10.’番に到達したら、その２つの数字の番号は、そのリストの中の他の番号とおなじく左寄せで書かれなければなりません。アイテムは、次の行が、その bullet ／数字よりも少ないか等しいインデントの場合の前までで終わります。

リストを終わらせるために２つの方法[13] が用意されています。ひとつのリストは、それぞれのアイテムが終了すると終わります。そのことは、トップのレベルのアイテムよりも少ないか等しいインデントの行の前までであるということを意味しています。また、空行[14] が 2 行あると終了します。その場合、すべてのアイテムが閉じていることになります。うまく管理するには、org-list-end-regexpの中のどれかのパターンの設定でリストを終わることです。事例を紹介します。

```
** ロードオブザリング
   渡しの大好きなシーンは（この順で）
   1. the Rohirrim の攻撃
   2. Eowyn が魔法使いの王と一緒に戦うところ
      + これはもともと本を読んだときも私のお気に入りのシーンだった
      + 私は Miranda Otto が本当に好きだ。
   3. Peter Jackson が Legolas に撃たれる
      - DVD だけで
      そのとき彼は本当に面白い顔をした。
   しかし、結局、映画全体を通して個性的なシーンがない。
   この映画での重要な俳優は:
   - Elijah Wood :: Frodo 役
   - Sean Austin :: Sam 役, Frodo の友達。私は今でも The Goonies
     の中で Mikey Walsh 役として素晴らしい役回りを演じたことを覚えている。
```

Org-mode では、これらのリストを正しく[15] 取り扱うために、埋め込んだり包んだりするコマンドをチューニングし、適切にエクスポートする (第 12 章「Exporting」p.132 を参照) ことによって、これらのリストに対応しています。これらのリストの構造を管理しているのがインデントであるため、#+BEGIN_...ブロックのような多くの構造的な構成を、特別なアイテムに帰属しているという目印のためにインデントを設定することができます。

（カレントのリストの階層のために使用するというよりも）サブリストのために、異なる bullet を使うことが読みやすくできると思ったら、org-list-demote-modify-bullet変数をカスタマイズしてください。

あるアイテムの最初の行（bullet または数字のついている行）にカーソルがあるときに、以下のコマンドがアイテムに作用します。それらのコマンドのいくつかは、リストの構造を完全なままたもつために自動的なルールのアプリケーションであることを暗示しています。これらのコマンドの動作のいくつかを独自のやりかたにしたいならば、それらを個別に無効にするために、org-list-automatic-rulesを設定してください。

TAB org-cycle

アイテムは見出しの階層と同じように折り畳むことができます。通常これらはカーソルがプレーンなリストアイテムの上にあるときに限り動作します。もっと

[13] これらどちらも無効にするためには、org-list-ending-methodを設定します。

[14] org-empty-line-terminates-plain-listsを参照してください。

[15] Org-mode では Emacs 用のみの埋め込みの設定を変更できます。XEmacs 用としては、Kyle E. Jones の ‘filladapt.el’を使用しなければなりません。この設定を起動するためには、‘.emacs’:に (require 'filladapt)を記述しておく必要があります。

詳しく理解するには、org-cycle-include-plain-lists変数を参照してください。もしもこの変数がintegrateに設定されているときは、プレーンなリストアイテムは下の階層の見出しと同様に取り扱われます。そのため筆のアイテムの階層は bullet または数字のインデントによって決定されます。アイテムは実際の見出しに常に従属しているのです。しかしながら、階層構造は完全に区別されたままになります。

M-RET org-insert-heading

新しいアイテムをカレントの階層に挿入します。前置引数を用いると、新しい見出し (2.5 節「Structure editing」p.9 を参照.) となります。もしもこのコマンドがアイテムの途中で使用されるならば、そのアイテムは 2 つに分割されます。そして 2 番目の部分は新しいアイテム[16] となります。もしもこのコマンドが、本文の前で実行されるならば、新しいアイテムは、カレントのアイテムの前に作成されます。

M-S-RET チェックボックス (5.6 節「Checkboxes」p.54 を参照) のついた新しいアイテムを挿入します。

TAB org-cycle

テキストがまだ書かれていない新しいアイテムの中で、最初の TAB でそのアイテムをその前のアイテムの子の階層に移動します。それに続けて TABを入力していくと、そのアイテムをリスト上で意味のある階層に移動し、そして最終的にもとあった一に戻ります。

S-up
S-down カレントのリストの中で、前の／次のアイテムにジャンプします。ただし org-support-shift-select がオフになっている時だけです。もしもそうなっていないなら、*C-up* と *C-down*のようなパラグラフのジャンプコマンドと全く同様の効果が現れるように使用することができます。

M-S-up
M-S-down サブアイテムを持っているアイテムを上下（同じインデントのついたアイテムの前後と入れ替わる）に移動します。もしもリストに序列があるならば、自動的に採番しなおします。

M-left
M-right 一つのアイテムのインデントを増減します。子のアイテムを残したままで。

M-S-left
M-S-right

サブアイテムを含んだまま、アイテムのインデントを増減します。初期設定では、アイテムのツリーはカレントのインデントに基づいて選択されます。直接連続してこれらのコマンドが何度も実行されたときは、たとえ新しいインデントが異なる階層であるとわかっていても、初期に選択されたリージョンが使用されます。新しい階層を使用するために、カーソルを移動させるコマンドの連鎖をブレイクする必要があります。

特別な場合として、リストの本当に最初のアイテムの上で、このコマンドを使用することで全てのリストを動かすことができます。この動作はorg-list-automatic-rulesを設定することで無効にすることができます。あ

[16] もしもアイテムを分割したくないならば、org-M-RET-may-split-line変数をカスタマイズしてください。

るリストのグローバルなインデントは、そのリストの後のテキストにはなんの影響も与えません。

C-c C-c　アイテムの行にチェックボックス (5.6 節「Checkboxes」p.54 を参照) がある場合には、チェックボックスの状態を切り替えます。ともかく、リスト全体について bullet とインデントの整合性を検証します。

C-c -　`org-plain-list-ordered-item-terminator`の設定により、異なる箇条書き/番号付きの bullet(‘`-`’, ‘`+`’, ‘`*`’, ‘`1.`’, ‘`1)`’) 、またはそれらのサブセットをもとにして全体のリストの階層、リストのタイプ、リストの位置[17] を切り替えます。N という数字の付いた前置引数を使用すると、これらのリストの中の N 番目の bullet が選択されます。もしも、このコマンドを呼び出したときにアクティブなリージョンがあるならば、選択された文章は普通のアイテムに変更されます。前置引数を使うと、すべての行がリストアイテムに変換されます。もしも最初の行がすでにリストアイテムだったならば、どのアイテムの符号もリストから削除されるでしょう。最後に、アクティブなリージョンでない場合でも、リストアイテムに変換されます。

C-c *　プレーンなリストのアイテムを見出しに変更します（そのロケーションによってはサブの見出しになることもあります。）2.5 節「Structure editing」p.9 を参照してください.、ここに詳しい説明があります。

C-c C-*　プレーンなリスト全体を可憐との見出しのサブツリーに変換します。チェックボックス (5.6 節「Checkboxes」p.54 を参照) は、チェックされていないとき（またはチェックされているとき）は TODO（または DONE）キーワードになるでしょう。

S-`left`/`right`
bullet 上、またはアイテムの行のどこかにカーソルが置かれているときに、このコマンドは、また、bullet のスタイルを切り替えます。詳細は `org-support-shift-select`に依存します。

C-c ^　プレーンなリストを並び替えます。次の並び替えの方法を入力します。数字順、アルファベット順、時間順、あるいはカスタムな機能の順番に。

2.8 Drawers

あるエントリーに関連する情報を保持していたいときがあるが、普段はその情報を見たくはないということがあります。こういうときのために、Org-mode は引き出しという機能を持っています。引き出しは `org-drawers`[18] 変数で設定する必要があります。引き出しはこんな形をしています。

```
** これは見出しです
   ここはまだ引き出しの外側です
   :DRAWERNAME:
   これは引き出しの内側です。
   :END:
   引き出しの後です。
```

[17] もっと多くの情報が必要ならば、`org-list-automatic-rules`の中の `bullet`を参照してください。

[18] `#+DRAWERS: HIDDEN PROPERTIES STATE`というような行を使ってファイル毎に引き出しを定義することができます。

見出し上で表示の切り替え (2.3 節「Visibility cycling」p.7 を参照) を行うとエントリーを隠したり表示したりすることができますが、引き出しの部分は 1 行に畳まれたままの状態になります。引き出しの中身を見るためには、カーソルを引き出しの行に移動し、そこで TAB キーを押すことが必要です。Org-mode は属性 (第 7 章「Properties and Columns」p.61 を参照) を保持するために、PROPERTIESという引き出しを使用します。そしてノート (5.3.2 節「Tracking TODO state changes」p.50 を参照) と時刻 (8.4 節「Clocking work time」p.77 を参照) の変化の状態を用意するために LOGBOOKという引き出しの中に保存をすることができます。もしも、状態の変化のためと似たような方法で素早くノートを LOGBOOK の引き出しに保存したいときには、このように使います。

C-c C-z　LOGBOOK のための引き出しにタイムスタンプ付きのノートを追加します。

2.9 Blocks

Org-mode はソースコードの例 (11.3 節「Literal examples」p.125 を参照) から時刻のログ情報 (8.4 節「Clocking work time」p.77 を参照) を記録することまで、いろいろな目的のために、begin...end というブロックを使用します。このブロックでは、行の先頭で TAB を押すことによって、折り畳んだり、折り畳みを解いたりすることができます。org-hide-block-startup変数を設定するか、以下のようなファイル毎の設定をすることで、起動時に全てのブロックを折り畳んでおくこともできます。

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

2.10 Footnotes

Org-mode は脚注の作成をサポートしています。Org-mode は、‘footnote.el’パッケージと対照的に、1 回限りの E メールのような文書だけでなく、大きな文書上で動作するよう設計されています。基本構文は ‘footnote.el’の構文で使われているのと良く似ており、i.e. インデントが認められていない、カラム 0 の角カッコの中の脚注の印によって始まるパラグラフの中で定義されます。もしも脚注の中でパラグラフを改行したいならば、LaTeX の用語である ‘\par’を使用します。脚注の参照は、テキストの中の単純な角カッコの中の記号です。例えば。

> Org-mode のホームページ [fn:1] は以前に比べて現在は相当改良されていると思います。
>
> ...
>
> [fn:1] リンク先は: http://orgmode.org

Org-mode では数字をベースとした構文を、名前のついた脚注とオプションのインラインでの定義へと拡張しています。プレーンな数字を (‘footnote.el’で行えるように) マーカーとして使うことは、下位の互換性としてサポートされていますが、LaTeX の snippet(11.7 節「Embedded LaTeX」p.128 を参照) と衝突する危険性があるのでお奨めはできません。以下に確かな参考資料を説明します。

`[1]`　プレーンな数字付きの脚注用の記号です。‘footnote.el’と互換性はありますが、‘[1]’のような記号は、snippet のコードとかぶることが多いので推奨しません。

`[fn:name]`
名前付きの脚注参照、そこでは nameがユニークな言葉によるラベルとなっており、さもなければ簡単に自動的に作成される、数字が用いられます。

`[fn:: これは脚注のインラインの定義です]`
: 参照のポイントに直接定義がなされる LaTeX のような無記名の脚注。

`[fn:name: a definition]`
: 脚注のインラインでの定義、それはまた、ノートのための名前を明確に規定します。Org-mode は同じノートに対して多重の参照を許容するので、新たな参照を作成するために、`[fn:name]`を使用することができます。

脚注のラベルは自動的に作成することができます。そうしないならば、あなた自身で名前を作成することができます。これは `org-footnote-auto-label`変数で操作され、`#+STARTUP` キーワードに対応します。詳細については変数の説明を参照してください。

以下のコマンドはプロパティを操作する助けとなります。

C-c C-x f　脚注の動作のコマンド。

カーソルが脚注参照上にあるときに、定義部分にジャンプします。カーソルが定義部分にあるときに、(最初の) 参照されている部分にジャンプします。

そうでなければ、新しい脚注を作成します。`org-footnote-define-inline`[19] によって、参照の一部として、または、`org-footnote-section`変数によって決定される場所の中に区分されて、テキストの中に正しく定義が配置されるでしょう。

前置引数と一緒にコマンドが呼び出された場合は、追加のオプションのメニューが提示されます。

`s`　参照の順場によって、脚注の定義は並び替えられます。編集している間は、
Org-mode は特定の並びの中に脚注の定義を並び替える努力はしません。
もしもそれらを並び替えたいならば、このコマンドを使用してください。
それによって `org-footnote-section`に従ってエントリーをまた移動します。
挿入／削除のあとに自動的に並び替えるには、変数を使うことで設定する
ことができます。

`r`　単純な `fn:N` の脚注を思い出してください。挿入／削除それぞれのあとの
自動的な採番は、`org-footnote-auto-adjust`変数を使う
ことで設定できます。

`S`　最初の `r`のショートカットで、`s`はアクションです。

`n`　すべての定義（インラインの定義もふくみます）を特別なセクションに集める
ことによって脚注を標準化します。そしてそれからそれらの定義を順番に採番

[19] 対応するインバッファの設定は、`#+STARTUP: fninline`または `#+STARTUP: nofninline`となります。

します。参照先にも番号がふられます。これは、ひとつのドキュメントを終了する前の最後の段階であることを意味します。(e.g. E メールを発送する)。
エクスポート機能はこのことを自動的に行い、
`message-send-hook`のような何かを行います。

`d` そのポイント、およびそれについての参照先の定義を削除します。

`org-footnote-auto-adjust`[20] と `nofnadjust`の変数に依存し、それぞれの挿入と削除のあとに、番号の振り直しと脚注の並び替えが自動的に行われます。

C-c C-c もしもカーソルが脚注の参照の上にあるときは、定義部分に飛びます。もしもそれがある定義ならば、参照先にジャンプして戻ります。前置引数と一緒に、脚注の場所を呼び出すときに、*C-c C-x f*として同じメニューが提供されています。

C-c C-o or *mouse-1/2*
脚注のラベルはまた、定義／参照先に対応してリンクを貼ります。そして通常のコマンドでこれらのリンクをフォローするための通常のコマンドを使用することができます。

2.11 The Orgstruct minor mode

もしも Org-mode の構造の編集とリストのフォーマットの動作について直感的な方法を好むのならば、あなたは Text モードや Mail モードと同じような他のモードのコマンドを使用したいと思うでしょう。`orgstruct-mode`というマイナーモードでそれが可能になります。*M-x orgstruct-mode*を使ってモードを切り替えるか、例えば Message モードの中で、デフォルトでそれを作動させるか、次のいずれかを用いて、

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

このモードがアクティブで、カーソルが見出しやリストアイテムの最初の行のような Org-mode のような行の上にあるときには、ほとんどの構造の編集のためのコマンドは動作するでしょう。たとえ、あなたが使っているメジャーモードの中で、同じキーが普通に異なる機能を持っているとしても。もしもカーソルがそういった特別の行の一つの中に置かれていなくても、Orgstruct モードは影の中で静かに隠れています。`orgstruct++-mode`を使ったときは、Org-mode は、それらのモードの中に、インデントやオートフィルの設定を書き出すでしょう。そして、アイテムの最初の行の後にアイテムのコンテクストを見つけるでしょう。

[20] 対応するインバッファのオプションは `fnadjust`と `nofnadjust`になります。

3 Tables

Org-mode は、高速で直感的なテーブルエディタを備えています。Emacs の ‘calc’パッケージを用いることで、スプレッドシートのような計算がサポートされています。（Emacs Calculator の詳細は、同パッケージのマニュアルを参照してください。）

3.1 組み込まれたテーブルエディタ

Org-mode はプレーンな ASCII 形式でのテーブル編集を容易にします。どのような行でも、空白文字を除く最初の文字が ‘|’であるとき、テーブルの一部であるとみなされます。‘|’は、列を区分するセパレータとしても使われます。Org-mode のテーブルは、次のような見た目になるでしょう。

```
| 名前   | 電話番号   | 年齢   |
|------+----------+------|
| 佐藤   | １２３４   | １７   |
| 田中   | ４３２１   | ２５   |
```

テーブルの中で TAB や RET、もしくは *C-c C-c* を押す度に、テーブルは自動的に表示が更新されます。TAB を押すとカーソルが次のフィールドに移ります（RET の場合は次の行へ）。また、テーブルの端もしくは水平ラインの直前の行で押せば、テーブルに新しい行が追加されます。テーブルのインデントは一行目によって設定されます。‘|-’で開始するどんな行も水平ラインとして解釈され、次にテーブルが更新される時に、テーブル幅いっぱいに水平ラインは拡張されます。つまり、上記のテーブルを作成するために、次のように入力するだけでよかったのです。

```
|名前|電話番号|年齢|
|-
```

そして TAB を押してテーブルを整列し、フィールドへの入力を始めます。さらに素早いテーブルの作成方法は、|名前|電話番号|年齢に続いて *C-c RET* を入力することです。

フィールドに文字を入力すると、Org-mode は DEL と Backspace、そしてすべての文字キーを特別な方法で扱います。文字の挿入と削除によって他のフィールドがズレてしまうことを避けるためです。また、TAB、*S-TAB* もしくは RET によって新しいフィールドへカーソルが移動した直後に文字を入力すると、自動的空白が挿入されます。もしもこの動作が気に入らない場合には、変数 org-enable-table-editor と org-table-auto-blank-field を調節してください。

テーブルの作成と変換

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

アクティブリージョンをテーブルに変換します。もし全ての行が少なくとも一つのタブを含んでいるならば、この関数は処理対象をタブ区切りの表であると想定します。もし全ての行がコンマを含んでいるならば、コンマ区切り（いわゆる CSV）であると想定されます。それ以外の場合は空白文字で区切られていると想定します。プレフィックスを用いることで、区切り文字を強制できます。すなわち、*C-u* で CSV、*C-u C-u* で TAB 区切りを指定できます。また、プレフィックスの引き数として整数値 N を用いれば、少なくとも N 個の連続した空白を区切りとして指定できます。条件に合わなければ、代わりとして TAB が区切りとして使われるでしょう。
もしアクティブリージョンが存在しないならば、このコマンドは空のテーブルを

生成します。しかし、|名前|電話番号|年齢 *RET* |- *TAB*のようにタイプしてテーブル作成を始める方が簡単です。

テールの整列とフィールドの動き

C-c C-c　org-table-align
カーソルを動かさずにテーブルを整列する。

<*TAB*>　org-table-next-field
テーブルを整列し、カーソルを次のフィールドに移す。必要ならば新たな行を生成する。

S-TAB　org-table-previous-field
テーブルを整列し、カーソルを前のフィールドに移す。

RET　org-table-next-row
テーブルを整列し、次の行にカーソルを下げる。必要ならば新しい行を追加する。行頭もしくは行末にカーソルがあるときの RETは、NEWLINE を意味し、テーブルの分割に使われます。

M-a　org-table-beginning-of-field
フィールド内の文頭にカーソルを移動する。もしくは、前のフィールドの文頭に移動する。

M-e　org-table-end-of-field
フィールド内の文末にカーソルを移動する。もしくは、次のフィールドの文末に移動する。

テーブルの列と行の編集

M-left　org-table-move-column-left
M-right　org-table-move-column-right
カーソルがある列を左右に移動する。

M-S-left　org-table-delete-column
カーソル位置の行を削除する。

M-S-right　org-table-insert-column
カーソル位置の左に新しい列を追加する。

M-up　org-table-move-row-up
M-down　org-table-move-row-down
カーソル位置の行を上下に移動する。

M-S-up　org-table-kill-row
カーソル位置の行もしくは水平ラインを削除する。

M-S-down　org-table-insert-row
カーソル位置の上に新しい行を追加する。プレフィックスを使うと、カーソル位置の下に追加される。

C-c -　org-table-insert-hline
カーソル位置の下に水平ラインを追加する。プレフィックスを使うと、カーソル位置の上に追加される。

C-c RET　org-table-hline-and-move
カーソル位置の下に水平ラインを追加し、カーソルを追加された水平ラインの次の行に移動する。

C-c ^　org-table-sort-lines

リージョンに含まれるテーブルの各行をソートする。ポイントのある列の情報がソートに利用され、ソート対象となる範囲は、最も近い水平ラインの行まで、もしくは、テーブル全体が指定される。カーソル位置がテーブルの第一列よりも前にあるときは、ソートに利用する列を指定するためにプロンプトが表示されます。すでにアクティブリージョンが存在する場合、マークは第一行とソートに利用する列を指定し、同時にポイントは、ソート対象に含まれる最終行に位置しなければなりません。このコマンドが表示するプロンプトは、ソートの種類（アルファベット順、数値順、もしくは時間順）を指定するものです。プレフィックスを利用すると、大文字と小文字が区別されます。

リージョン

C-c C-x M-w　org-table-copy-region

テーブルの矩形領域を特別なクリップボードにコピーします。ポイントとマークは、矩形領域を構成する末端となるフィールドを決定します。もしアクティブリージョンがなければ、カーソル位置のフィールドだけをコピーします。この処理は、テーブルの水平ラインを無視します。

C-c C-x C-w　org-table-cut-region

テーブルの矩形領域を特別なクリップボードにコピーし、領域内の全てのフィールドを空にします。つまりこれは「カット」操作です。

C-c C-x C-y　org-table-paste-rectangle

テーブルに矩形領域をペーストします。領域の左上がカーソル位置のフィールドに上書きされます。ペーストする領域に重なるすべてのフィールドは上書きされます。対象とするテーブルに矩形領域が合わないならば、必要に応じてテーブルは拡張されます。この処理は、テーブルの水平ラインを無視します。

M-RET　org-table-wrap-region

カーソル位置でフィールドの文字列を分割し、カーソル以降を一つ下のフィールドの文頭に移動します。アクティブリージョンが存在し、またポイントとマークの両方が同じ列にあるとき、列に含まれるテキストは、与えられた行数を最小化するように改行されます（訳注：余計な空白が消される）。プレフィックスで指定する整数値は、希望する行数に合わせるために使われます（訳注：M 行を N 行に圧縮できます）。もし選択領域がない状態でプレフィックスを指定すると、カーソル位置のフィールドは空白になり、元々あった文字列は一つ上のフィールドの文末に付け加えられます。

計算機能

C-c +　org-table-sum

カーソル位置の列、もしくは、アクティブリージョンで定められた矩形領域に含まれる数値を合計する。計算結果はエコー領域に表示され、*C-y*で挿入できる。

S-RET　org-table-copy-down

カーソル位置のフィールドが空白のとき、上にある空白でないフィールドから文字列をコピーする。空白でないときには、値を次の行のフィールドにコピーし、カーソルも移動させる。org-table-copy-incrementの値に依存して、フィールドが整数値のときは値を一つ増やしてからコピーされるでしょう。大きすぎる値の整数の場合は値は増やされません。また、プレフィックスで 0を用いれば、一

時的に値の増加を防げます。このキーバインドは、shift-selection とこれに関連するモードでも使われています（15.10.2 節「Conflicts」p.192 を参照）。

その他の機能

C-c ‘　`org-table-edit-field`

個別のウィンドウでカーソル位置のフィールドを編集する。この昨日は、フィールド全体が表示されていないときに便利です（3.2 節「列幅と整列」p.23 を参照）。プレフィックス *C-u* と伴って関数が呼ばれると、フィールドの全ての内容が表示されるため、フィールド内で編集できます。

M-x org-table-import

ファイルをテーブルとしてインポートする。テーブルは、タブもしくは空白で区切られている必要があります。たとえば、スプレッドシートのテーブルやデータベースの情報をインポートするために利用します。というのも、これらのプログラムは一般的にタブ区切りのテキストフィールドを書き出すことが可能なためです。このコマンドは、ファイルの内容をバッファに挿入することで作動し、領域をテーブルに変換します。どのようなプレフィックスがコンバータに与えられても、セパレータを決定するために利用されます。

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

org-mode のバッファにテーブル状のテキスト（訳注：文字列がタブで区切られているテキスト領域など）をペーストすることでも org-mode のテーブルを生成することができます。*C-x C-x* でバッファにペーストされているテキストを選択し、*C-c |* で org-mode のテーブルに変換します（前述の *Creation and conversion* を参照してください）。

M-x org-table-export

テーブルをエクスポートする。エクスポートされるファイルでは標準でタブ区切りが使われる。たとえば、スプレッドシートやデータベースプログラムと情報を交換するために使います。ファイルのエクスポートに使われるフォーマットは、変数 `org-table-export-default-format` で調節することができます。また、ファイル名を指定するためにプロパティ `TABLE_EXPORT_FILE` を、サブツリーでのテーブルエクスポートのフォーマットを指定するためにプロパティ `TABLE_EXPORT_FORMAT` を指定できます。org-mode はエクスポートされたテーブルについて極めて汎用的なフォーマットをサポートします。エクスポートのフォーマットは、Orgtbl のラジオテーブルで使われているものと同じです。より詳しい説明は“char 65.5.3 節「Translator functions」p.200 を参照してください。

‘|’で始まる行を思い通りに編集するために自動的なテーブルの編集が好みでない場合は、次のコマンドでこの機能を停止することができます。

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

こうすると、テーブルのコマンドで利用できるのは、*C-c C-c* によるマニュアルな再整列だけになります。

3.2 列幅と整列

テーブルの各列の幅はテーブルエディタによって自動的に決定されます。また、列の配置も自動的に決定されます。具体的には、列の中で、数値でないフィールドに対する数値（と解釈できる）フィールドの割合に応じて決まります。

単一かもしくはごく少数のフィールドでより多くのテキストを扱おうとすると、困ったことに列幅が広がってしまいます。もしくは、フィールドの内容にかかわらず、固定幅の列でテーブルを作成したいと思うかもしれません。列の幅を指定するためには、列のどこにあってもよいですが、一つのフィールドが文字列 '<N>'だけを含む必要があります。ここで 'N'は、列の幅を指定する整数値の文字列です[1]。次に行なわれるテーブルの整列では、この数値で列の幅が設定されます。

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

指定された幅よりも広いフィールドは一部が切り取られ、文字列 '=>'で終わります。フィールド内に表示されていたテキストは、バッファ内部にそのまま存在し、表示が隠されていることに注意してください。隠されたテキストも含めてすべてを表示するためには、対象とするフィールドにマウスカーソルを合わてください。ツールチップが現われて、フィールドが含むすべての内容を表示されます。このようなフィールドを編集するには、*C-c* ' を使います (*C-c*に続いてバッククオートを入力します)。フィールドの全ての内容を表示した新しいウィンドウが開かれます。フィールドの内容を編集し、*C-c C-c*で完了します。

幅を狭くした列のあるテーブルを含むファイルを訪問するとき、文字列の隠蔽はまだ実行されていません。そして、希望する見た目にするにはテーブルを整列する必要があります。オプションの org-startup-align-all-tables を設定すると、ファイルを訪問するときにファイルにある全てのテーブルが整列されます。ただしスタートアップが少し遅くなります。次を利用すればファイルごとにこのオプションを設定することもできます。

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

数値の多い列を右揃えにして、文字列の多い列を左揃えにする自動的な整列を無効にしたいならば、'<r>'、'c'[2]、もしくは '<l>'を似たような方法で利用できます。'<|10>'のようにすれば列の揃えとフィールドの幅を同時に指定できます。

書式の設定情報のみを含む行は、ドキュメントをエクスポートするときに自動的に削除されます。

3.3 Column groups

org-mode のテーブルをエクスポートすると、標準で垂直ラインを表示しません。これは、一般に視覚的な満足度をより高めるためです。しかし場合によっては、テーブルを列のグループで構造化するために垂直ラインが役に立つます。これは水平ラインがいくつかの行をグループ化するために役立つことと同じです。列のグループを指定するために、最初のフィールドが '/'だけを含む特別な行を使います。それ以降のフィールドについては、'<'を含むとき、その列がグルーピングされる列の始めであることを意味します。'>'を含む場合は、グループの終了を表します。もしくは、'<>'を含む列はこれ自体を一つのグループにします。列のグループの境界は、エクスポート時に垂直ラインが表示されるようになります。以下に例を示します。

[1] この機能は、XEmacs では動作しません

[2] Emacs の表示上は中央揃えにはできませんが、HTML にエクスポートするときに中央揃えにできます

```
| 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)))
```

表示させたいすべての垂直ラインの後ろに列のグループ開始を指定するだけでも十分です。

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

3.4 Orgtbl マイナーモード

org-mode のテーブルエディタの直感的な動作を気に入ったら、テキストモードやメールモードのように他のモードで利用したくなるかもしれません。これはマイナーモードの Orgtbl モードが実現してくれます。*M-x orgtbl-mode*でトグルできます。標準で Orgtbl モードを有効にするには、たとえばメッセージモードのときに、次の設定を使います。

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

さらに、いくつかの特別な処理を追加することで、orgtbl モードの任意のシンタックスでテーブルをメンテナンスできます。たとえば、簡単に、そして orgtbl モードの機能で LaTeX のテーブルを構築できます。これは表計算機能も含んでいます。さらなる詳細は、“char 65.5 節「Tables in arbitrary syntax」p.197. を参照してください。

3.5 The spreadsheet

org-mode のテーブルエディタは、表計算機能を実装するために Emacs の‘calc’を利用します。他のフィールドの値から別なフィールドの値を導くために Emacs Lisp の書式も評価できます。十分な機能があるものの、org-mode での実装は他の表計算ソフトと全く同等というわけではありません。たとえば、org-mode は列数式の概念を理解しています。これは、関連する各フィールドに数式をコピーすることなく、ヘッダーを除く列の全てのフィールドに適用されます。数式のデバッガもあります。また、数式が参照しているフィールドに対応したフィールドを、テーブル内でハイライトする機能や矢印キーでリファレンスに移動する機能がある数式エディタもあります。

3.5.1 References

テーブル内部のフィールドを他のフィールドの値から計算するためには、必ず数式が他のフィールドか範囲を参照していなければなりません。org-mode では、名前、絶対的または相対的な位置によってフィールドを参照することができます。フィールドの位置がどこかを特定するためには、そのフィールドで *C-c ?*を押してください。もしくは、グリッド表示をトグルするために *C-c }*を使用してください。

フィールドの参照

数式は別なフィールドの値を 2 つの方法で参照できます。他の表計算ソフトと同じように、B3のような文字と数値の組み合わせでフィールドを参照できます。三行目の第二フィールドを意味しています。

org-mode では、もう一つの方法を好みます[3]。より一般的な次のような表記です。

`@row$column`

また、相対的な参照も認めています。すなわち、フィールドの列と行に対する値が計算されいる参照です。このような相対的な参照は、数式を一度だけ記録すればよく、数式のコピーや変更せずにたくさんのフィールドで利用できます。

列の参照は、‘1’、‘2’、...‘*N*’のように絶対的に表されるか、もしくは、カーソル位置の列に対して相対的に ‘+1’、‘-2’のように表されます。$>はテーブルの最終列を参照します。さらに、$>-2のようなオフセットを指定できます。この場合、一番右から三番目の列を表します。

行はデータを含む行のみをカウントして、水平ライン（hline）は無視します。列と同様に、‘1’...‘*N*’のように絶対的な行の番号を利用できます。また、‘+3’や ‘-1’ のようにカーソル位置の行に対する相対的な位置を表し、@>でテーブルの最終行を参照します[4]。ある水平ラインに対する相対的な行を指定することもできます。‘I’は最初の hline への参照です[5]。‘II’は二番目の水平ラインというように指定します。‘-I’は、カーソル位置の行の上方にある最初の水平ラインを参照し、‘+I’は下方にある最初の水平ラインを参照します。‘III+2’のように指定すると、テーブルの三番目の水平ラインから二番目の行を表します。

‘0’はカーソル位置の行と列を参照します。また、もし参照について列と行のいずれかを無視すれば、行または列が暗黙に参照されます。

org-mode の符号無し数値の参照は、静的な参照です。これは、二つの異なるフィールドにある数式の中で、同じ参照を利用すれば、常に同じフィールドが参照されます。符号付き数値の参照は、動的な参照です。これは、見た目上同じ参照であっても、数式で計算されるフィールドに依存して、異なるフィールドを参照できるためです。

いくつかの例を示します。

@2$3	第二行、第三列
C2	同上
$5	現在行の第五列
E&	同上
@2	現在列、第二行
@-1$-3	カーソル位置から一つ上、左に三つ目
@-I$2	カーソル位置の上方の水平ライン直下の第二列

範囲参照

複数のフィールドで構成する矩形範囲を参照できます。この範囲は、二つの参照を二つのドット ‘..’で接続することで指定します。二つの参照が共にカーソル位置の行にあるとき、単純に ‘$2..$7’と指定できます。しかし、少なくとも一つのフィールドが異なる列にあるときは、少なくとも一方のフィールドについて@row$columnのような一般的な形式を使う必要があります（つまり、正しく解釈させるために参照を ‘@’で始めなければなりません）。具体例は次のようになります。

---

[3] org-mode はユーザが指定する ‘B4’のような参照を理解しますが、編集のための数式を提供するときはこのシンタックスは使われません。変数 org-table-use-standard-referencesを使うことで、この動作を変更できます。

[4] 後方互換のために、‘$LR5’や ‘$LR12’のような特別な名前も利用できます。これらは、テーブル最終行の 5 番目と 12 番目への確実な参照です。しかしながら、このシンタックスは廃止予定であり、新たな文書で使うべきではありません

[5] テーブルの各行を分離している水平ラインのみカウントされることに注意してください。ヘッダーの上に水平ラインがあるテーブルでは、その水平ラインをカウントしません

`$1..$3`	カーソル位置の行の始めの 3 フィールド
`$P..$Q`	列の名前を使った範囲（以下の詳細を見てください）
`@2$1..@4$3`	二つのフィールド間にある 6 フィールド
`A2..C4`	同上
`@-1$-2..@-1`	カーソル位置の左の列の 2 つ上方の行の 3 フィールド
`@I..II`	第一と第二の水平ラインに挟まれた領域（`@I..@II`の短縮表記）

範囲参照は Calc のベクトル関数に代入可能な値のベクトルを返します。範囲に含まれる空のフィールドは、普通は除去されます。これはベクトルが空ではないフィールドのみを含むようにするためです（ただし下記の‘E’モードスイッチも参照してください）。もし、すべてのフィールドが空ならば、数式のシンタックスエラーを避けるために‘[0]’が返されます。

数式中のフィールドの座標

Calc 形式と Lisp 形式の数式では、数式の演算結果が指すフィールドの行と列の番号を取得するために`@#`と`$#`を利用できます。伝統的な Lisp の数式に相当するのは `org-table-current-dline`と `org-table-current-column`です。たとえば、

`if(@# % 2, $#, string(""))`	奇数行に限定して列の番号を抽出する
`$3 = remote(FOO, @@#$2)`	テーブル FOO の二列目をコピーし、 現在のテーブルの三列目に上書きする

二つ目の例では、テーブル FOO は少なくとも、カーソル位置のテーブルが持つ行数と同じ数の行がなければなりません。たくさんの行があると処理が重くなる[6] ことに気を付けてください。

名前付き参照

‘`$name`’は、列の名前として解釈されます。パラメータや定数を扱います。定数は、変数`org-table-formula-constants`を利用してグローバルに定義されます。また、次のような一文を追加して、ファイルのローカル変数として定義されます。

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

プロパティ（第 7 章「Properties and Columns」p.61 を参照）もテーブルの数式で定数として扱われます。プロパティ‘`:Xyz:`’については‘`$PROP_Xyz`’という名前を使います。そして、このプロパティは現在のアウトラインエントリーと上位を階層的に検索されます。もし‘`constants.el`’パッケージを読み込んでいるならば、これも定数を決めるために使われます。このパッケージには、プランク定数‘`$h`’のような物理定数、そしてキロメーター‘`$km`’のような単位が含まれています[7]。列の名前とパラメータは、特別なテーブルのラインで設定できます。詳細は後述します（3.5.8 節「Advanced features」p.32.）。すべての名前は文字から始まり、それ以降は文字と数値で構成します。

リモート参照

異なるテーブルの定数、フィールドそして範囲を参照できます。現在のファイルでも、異なるファイルにある場合も参照できます。シンタックスは、

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

---

[6] この計算の規模は、O(N^2) のオーダーです。テーブル FOO が、各フィールドをコピーするためにパースされるのが原因です。

[7] ‘`constants.el`’は、`SI`と `cgs`の二つの異なる単位系で定数の値を提供します。どちらが利用されるかは、変数`constants-unit-system`の値に依存します。カレントバッファで値を設定するために、`#+STARTUP`でオプション `constSI`と `constcgs`を指定します。

NAME は別なファイルにあるテーブルの名前で、テーブルの前の行に`#+TBLNAME: NAME`と設定しておきます。エントリーの ID も利用でき、別なファイルにあるものも指定できます。この場合はエントリーに含まれる最初のテーブルを参照します。REF は前述したような絶対的なフィールドか範囲となる参照で、`@3$3`や`$somename`と表され、参照で指定したテーブルにおいて有効になります。

3.5.2 Formula syntax for Calc

数式は、Emacs の‘`Calc`’パッケージが理解できる、任意の代数表現になります。**通常の計算とは異なる、‘`Calc`’の慣例に気を付けてください。‘`/`’は‘`*`’よりも低く優先されます。つまり、‘`a/b*c`’は、‘`a/(b*c)`’として解釈されます。**。`calc-eval`（*GNU Emacs Calc Manual* の “Calling Calc from Your Lisp Programs” 節 を参照）で評価される以前に、先ほどのルールに従って変数が代入されます。範囲指定するベクトルは Calc‘`vmean`’や‘`vsum`’のようなベクトル関数に直接渡されます。

数式は、セミコロンの後に続くオプションモードの文字列を含むことができます。この文字列は実行時に Calc や他のモードに作用するフラグで構成されます。デフォルトでは、org-mode は標準の Calc モード（精度=12 桁、角度単位=度、分数／シンボリックモード=OFF）を使います。ただし、表示フォーマットは、テーブルをコンパクトに保つために、`(float 8)`に変更されています。

`p20`	Calc の内部計算精度を 20 桁に設定
`n3 s3 e2 f4`	通常表記、科学指数、工学指数、固定小数点 の Calc の出力結果が org-mode に戻る。 Calc の計算精度が表示上の精度よりも高い限り、 Calc 表記は精度上の制限を受けない。
`D R`	角度モード（度／ラジアン）
`F S`	分数／シンボリックモード
`N`	全フィールドを数値として解釈。非数値は 0 を使用
`T`	強制的に文字列として解釈
`E`	領域中のフィールドを空に保つ
`L`	リテラル

大きな整数値を使用したり、浮動小数点での高精度な計算と表示を行なわないならば、Calc がすでにフォーマットした結果ではなく、org-mode に戻された後の Calc の出力結果を再フォーマットするために、`printf`によるフォーマット指定を代替として与えることができます[8]。いくつかの例を示します。

`$1+$2`	第一と第二フィールドの和
`$1+$2;%.2f`	同上。ただし、小数点以下 2 桁表示
`exp($2)+exp($1)`	関数も利用可能
`$0;%.1f`	小数点以下 1 桁に再フォーマット
`($3-32)*5/9`	華氏から摂氏への変換
`$c/$1/$cm`	周波数 [Hz] から波長 [cm] への変換（constants.el を使用）
`tan($1);Dp3s1`	角度計算（3 桁精度、科学指数 1 桁）
`sin($1);Dp3%.1e`	同上。ただし、printf での表示指定
`vmean($2..$7)`	列の平均値、ベクトルを利用

[8] `printf`による再フォーマットは、精度の影響を受けます。`integer`や `double`に変換された値が渡されるためです。`integer`は、符号付き 32 ビット整数値に丸め込まれます。`double`は、全体が 64 ビット精度に制限され、近似的に 16 ビットの 10 進数の有意桁数があります。

`vmean($2..$7);EN`　　同上。ただし、空フィールドを 0 とする
`taylor($3,x=7,2)`　　$3 の 2 次のテーラー級数で x=7 の値

Calc は論理演算の完全な集合も含んでいます。例として次があります。

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

3.5.3 数式としての Emacs Lisp 形式

Emacs Lisp で数式を記述することもできます。Calc の機能が不十分なら、文字列操作と構造の制御に役立ちます。開き括弧が後ろに続くシングルクォートで数式が始まるとき、Lisp 形式として解釈されます。評価値は、文字列か数値で返ります。‘Calc’の数式と同じように、セミコロンの後にモードと printf フォーマットを指定できます。Emacs Lisp 形式では、フィールドの参照が、Lisp 形式で挿入されることを意識する必要があります。デフォルトでは、参照はフィールドを含む（ダブルクォートで括られた）Lisp の文字列として挿入されます。もし‘N’モードスイッチが指定すると、全ての参照された要素は数値になり（非数値のフィールドは 0 になる）、クォートなしで、Lisp 形式の数値として挿入されます。‘L’フラッグを指定すると、全てのフィールドは、クォートなしで、そのままの内容で挿入されます。すなわち、もし参照が Lisp 形式の文字列として挿入されることを望むならば、`"$3"`のように、ダブルクォートで参照のオペレータ自体を包んでください。範囲はスペースで区切られたフィールドとして挿入されます。そのため、リストやベクトルシンタックスに埋め込んだりできます。いくつかの例を示します。Lisp を用いて計算をするときに‘N’モードがどのように使われるかを注意してください。

一列目の内容について、一文字目と二文字目を入れ替える
```
'(concat (substring $1 1 2) (substring $1 0 1) (substring $1 2))
```
一列目と二列目を加算する。Calc の`$1+$2`と同じ
```
'(+ $1 $2);N
```
列 1 から列 4 の合計を計算。Calc の `vsum($1..$4)`と同じ
```
'(apply '+ '($1..$4));N
```

3.5.4 Field and range formulas

特定のフィールドに数式を割り当てるためには、‘:=’に続けて、直接フィールドに書き込みます。たとえば、‘`:=vsum(@II..III)`’のようにします。カーソルがフィールドにある状態で、TAB や RET、もしくは *C-c C-c* を押すと、入力した数式はそのフィールドのための数式として保存され、評価されたのち、フィールドの表示が演算結果で置き換わります。

数式はテーブルの下にある‘`#+TBLFM:`’で始まる特別な行に保存されます。テーブルの中で第三行目の 4 番目のフィールドで数式を入力すると、この数式は‘`@3$4=$1+$2`’のように記述されます。適当なコマンドで列と行の挿入／消去／入れ替えを行なうとき、保存された数式の中の絶対参照（相対参照では関係ありません）は、同一のフィールドを参照するために変更されます。もちろん通常の編集コマンドを用いてテーブルの構造を編集するときは正しくありません。したがって、マニュアルで数式を補正しなければなりません。フィールドに数式を記入する代わりに、次のコマンドも使用できます。

C-u C-c =　　`org-table-eval-formula`

現在のフィールドに新しい数式をインストールします。このコマンドは、‘`#+TBLFM:`’行から選ばれたデフォルトの数式を表示し、現在のフィールドに適用して、さらに保存します。

数式に異なるフィールドの数値を割り当てるために、等式の左側は特別表現を指定できます。範囲数式を入力するショートカットはありません。これを入力するには、数式エディタ

（3.5.6 節「Editing and debugging formulas」p.30 を参照）を用いるか、#+TBLFM:行に直接記述します。

$2=　列の数式、列の全体に対して有効。よく利用されるため、org-mode はこれらの数式を特別な方法で扱います。3.5.5 節「Column formulas」p.30. を参照してください。

@3=　行の数式、特定な行の全てのフィールドに適用する。@L=は最終行を意味する。

@1$2..@4$3=
　範囲数式、与えられた矩形領域の全てのフィールドに適用する。これはある行の全てのフィールドではなく一部に対して数式を割り当てることにも利用できます。

$name=　名前付きフィールド（3.5.8 節「Advanced features」p.32 を参照）

3.5.5 Column formulas

数式に$3=のような単純な列の参照を指定すると、同様の数式がその列の全てのフィールドで使用されます。これには非常に有用な条件があります。(1) テーブルが水平ラインを含む場合は、最初の水平ラインよりも上に存在する全ての行がテーブルのヘッダーの一部であると考えられ、列の数式によって変更されません。フィールドや範囲の数式からすでに値を取得しているフィールドは、列の数式は関与しません。これらの条件によって列の数式はとても使いやすくなっています。

列に数式を割り当てるためには、列のフィールドに直接記述します。イコールの後に続くように、'=$1+$2'としまう。同じフィールド内で TAB や RET もしくは *C-c C-c* を押すと、入力した数式はそのフィールドのための数式として保存され、評価されたのち、フィールドの表示が演算結果で置き換わります。フィールドが '=' だけの場合、以前に列で保存した数式が利用されます。各列について org-mode は、直前に利用した数式だけを記憶します。'#+TBLFM:' 行で、列の数式は '$4=$1+$2' のように保存されます。列数式のイコールの左側には、列の名前を置くことができず、数値の列の参照か$>を置かなければなりません。

フィールドに数式を書き入れる代わりに、次のコマンドも利用できます。

C-c =　org-table-eval-formula
　現在の列に新しい数式を書き込み、数式の演算結果で置き換える。このコマンドは '#+TBLFM:' の列から取得したデフォルトの数式を表示し、カーソル位置のフィールドに適用してから保存します。数値のプレフィックス（たとえば *C-5 C-c =*）を用いると、現在の列で連続したフィールドに同じ効果を与えます。

3.5.6 Editing and debugging formulas

ミニバッファかもしくは、直接フィールド内で、個々の数式を編集できます。org-mode はテーブルに含まれるすべてのアクティブな数式がある特別なバッファも準備しています。数式を編集しようとすると、org-mode は、もし可能ならば参照を標準のフォーマット（B3や D& のような記法）に変換します。もし内部フォーマット（@3$2や$4のような記法）を用いて編集したい場合は、変数 org-table-use-standard-references を設定してください。

C-c = or *C-u C-c =*　org-table-eval-formula
　ミニバッファでカーソル位置の列やフィールドに結びついた数式を編集する。3.5.5 節「Column formulas」p.30 と 3.5.4 節「Field and range formulas」p.29 を参照してください。

C-u C-u C-c =　org-table-eval-formula
カーソル位置のフィールドにアクティブな数式（フィールドの数式、もしくは列の数式）を再挿入します。これはフィールド内で直接的に編集できるようにするためです。ミニバッファでの編集葉理も有利なのは、*C-c ?*を使えることです。

C-c ?　org-table-field-info
テーブルのフィールド内で数式を編集するときに、数式内でカーソルが置かれている参照によって指し示されたフィールドをハイライトする。

C-c }　オーバーレイ（org-table-toggle-coordinate-overlays）を使用して、テーブルの行と列の番号を表示するようにトグルする。テーブルが整列される度に表示が変わり、*C-c C-c*で強制的に表示を更新できます。

C-c {　数式のデバッガを ON ／ OFF する（org-table-toggle-formula-debugger）。

C-c '　org-table-edit-formulas
特別なバッファで現在のテーブルのすべての数式を編集する。バッファでは数式はラインごとに一つずつ表示される。カーソル位置のフィールドにアクティブな数式があるとき、カーソルエディタのカーソルはその数式をマークします。特別なバッファの中では、org-mode は、カーソルがあるどんなフィールドや範囲参照も自動的にハイライトします。数式の編集、削除、追加ができることに加え、次のコマンドが利用できます。

C-c C-c or *C-x C-s*　org-table-fedit-finish
数式エディタを抜けて、修正された数式を保存する。プレフィックス*C-u*を使うと、新しい数式をテーブル全体に適用します。

C-c C-q　org-table-fedit-abort
変更を破棄して数式エディタを抜ける。

C-c C-r　org-table-fedit-toggle-ref-type
数式エディタの全ての参照について、(B3のような）標準の記法と（@3$2 のような）内部の記法についてトグルする。

TAB　org-table-fedit-lisp-indent
カーソル位置の Lisp 数式を見やすくしたりインテンドする。ラインに Lisp 数式が含まれると、Emacs Lisp のルールに基づいて数式を整える。さらに TABを押すと、整えられた数式を崩して元の状態に戻す。開いた数式では、Emacs Lisp モードのように TABでさらにインテンドする。

M-TAB　lisp-complete-symbol
Emacs Lisp モードのように Lisp の記号を補完する。

S-up/down/left/right
カーソル位置の参照を移動する。たとえば、参照が B3で *S-right*を押すと、C3に変化する。これは相対参照や水平ラインの参照についても同じように動作する。

M-S-up　org-table-fedit-line-up
M-S-down　org-table-fedit-line-down
org-mode のバッファにある列の数式へのテストラインを上下に移動する。

M-up　org-table-fedit-scroll-down
M-down　org-table-fedit-scroll-up
テーブルを表示するウィンドウをスクロールする。

C-c }　テーブルの座標グリッドを ON ／ OFF する

テーブルのフィールドを空欄にしても、そのフィールドに結びつけられた数式は削除されません。これは数式が別な行（‘#+TBLFM:’行）に保存されているためです。次に再計算が行なわれる際に、再びフィールドに数式の結果が戻ります。フィールドから数式を削除するためには、数式を表示させて空にするか、‘#+TBLFM:’行を編集する必要があります。

‘#+TBLFM:’行を直接編集することができ、変更した数式を再適用できます。これはラインにおいて *C-c C-c*を押すか、テーブルで通常の再計算コマンドを発行することで実行できます。

数式のデバッグ

数式を評価してエラーが生じた時は、フィールドの文字列が‘#ERROR’に変わります。バグを見つけるために、変数の代入と計算をする間に何が起こっているかを調べたいならば、Tbl メニューの数式のデバッグを有効にして、計算をやり直してください。たとえば、フィールド内で *C-u C-u C-c = RET*と押します。すると、詳細な情報が表示されます。

3.5.7 Updating the table

テーブルの再計算は、通常は自動的に行なわれず、コマンドにより実行する必要があります。3.5.8 節「Advanced features」p.32 を参照してください。少なくとも半自動で計算を行ないます。

テーブルのある行もしくはテーブル全体を再計算するために、次のコマンドを使います。

*C-c **　org-table-recalculate
現在行を最初に保存されている列の数式を用いて、左から右に、現在行の全てのフィールドと範囲数式を再計算する。

*C-u C-c **
C-u C-c C-c
行ごとにテーブル全体を再計算する。最初の水平ラインの以前のどんな行も再計算されません。これらはテーブルのヘッダであると想定します。

*C-u C-u C-c **　or　*C-u C-u C-c C-c*　org-table-iterate
計算結果の変化が生じなくなるまでテーブルの再計算を繰り返す。これは、計算の流れにおいて後で計算されるフィールドの値を利用している、いくつかのフィールドを計算するときに必要になります。

M-x org-table-recalculate-buffer-tables
現在のバッファに含まれる全てのフィールドを再計算する

M-x org-table-iterate-buffer-tables
テーブル間の依存関係を収束するために、カレントバッファの全てのテーブルを反復計算する。

3.5.8 Advanced features

もしも自動的にフィールドを再計算したい、もしくは、フィールドと列に名前を割り当てたいならば、テーブルの第一列を特別なマーキング文字を格納するために予約しておく必要があります。

C-#　org-table-rotate-recalc-marks

第一行の計算用マーカを‘ ’、‘#’、‘*’、‘!’、‘$’の順番に循環する。アクティブリージョンがあれば、その領域のすべてのマーカを変更する。

例として、学生の試験結果を集めて、自動再計算の機能を使うテーブルを示します。

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

重要: このような特別なテーブルについて、*C-u C-c **を使った再計算は、‘#’もしくは‘*’でマークされた行、また、自身に対して数式が適用されているフィールドだけに影響することに気をつけてください。最初のフィールドが空な行については、列の数式は適用されません。

マーキング文字には、次のような意味があります。

‘!’　行に含まれるフィールドが、対応する列の名前を表す。これは‘$6’ではなく‘$Tot’として列を参照できるようにするため。

‘^’　上側の列のフィールドについて名前を定義する。この定義を用いることで、テーブルのどのような数式でも、値‘10’を参照するために‘$m1’を利用できる。また、名前を定義するフィールドに数式を割り当てると、‘$name=...’として保存される。

‘_’　‘^’とほぼ同じだが、下側の列のフィールドに対する名前を定義する。

‘$’　この行のフィールドは、数式のパラメータを定義する。たとえば、‘$’が指定された行で、フィールドが‘max=50’を含むならば、テーブルの数式は‘$max’ を用いて値‘50’を参照できる。パラメータは、正に定数として動作し、テーブルごとに定義される。

‘#’　行の中で TAB、*RET*、または *S-TAB*を押すと、この行のフィールドは自動的に再計算される。また、この行は *C-u C-c **によるグローバルな再計算のために選択される。このコマンドでは、マークされていない行はそのままに維持される。

‘*’　*C-u C-c **によるグローバルな再計算のためにこの行を選択する。ただし、自動的な再計算には利用されない。自動再計算が編集速度を著しく低下させる場合に利用します。

‘マークされていない行は、*C-u C-c **による再計算から除外される。再計算’
されるべきすべての行は、‘#’や‘*’でマークされるべきである。

‘/’　この行はエクスポートしない。表示幅を狭くする‘<N>’マーカ、もしくは列のグループマーカを含む列について便利である。

最後に、素晴らしい‘calc.el’パッケージができることを学ぶ知識欲を刺激するために、一つのテーブルを示します。このテーブルは、いくつかの関数に対して、xにおける n次のテーラー級数を計算します。

```
|---+-------------+---+-----+--------------------------------------|
|   | 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 は、org-mode のテーブルに保存された情報による 2 次元と 3 次元のグラフを生成できます。‘Gnuplot’ http://www.gnuplot.info/ と‘gnuplot-mode’ http://cars9.uchicago.edu/~ravel/software/gnuplot-mode.htmlを利用しています。動作を確認するためには、Gnuplot と Gnuplot モードの両方がシステムにインストールされていることを確かめてください。その上で、次に示すテーブルで org-plot/gnuplotを呼び出してください。

```
#+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 |
```

Org-Plot は、テーブルのヘッダをラベルとして適用できます。ラベル、タイプ、コンテンツ、プロットの外観は、テーブルの上の#+PLOT:で始まる行によって制御可能です。以下の Org-Plot オプションの完成したリストを見てください。さらなる情報と例は、http://orgmode.org/worg/org-tutorials/org-plot.htmlにある Org-Plot のチュートリアルを参照してください。

プロットオプション

set　　グラフの描画時に設定される gnuplotのオプションを指定する。

title　　プロットのタイトルを設定する。

ind　　x軸として利用するテーブルの列を指定する。

deps　　Lisp スタイルのリストで描画するように列を指定する。括弧で括られ、スペースで分離されます。たとえば、dep:(3 4)とすると、第三と第四列を描画します（デフォルトでは、indの列を除いて他の全ての列が描画される）。

type　　プロットの種別（2d、3dもしくは grid）を指定する。

`with`　`with`オプションで挿入される、各プロット点の表示種別を指定する。たとえば、`lines`、`points`、`boxes`、`impluses`などで、デフォルトは `lines`。

`file`　プロット結果を外部ファイルに出力したいときに、"*path/to/desired/output-file*"のように指定する。

`labels`　`deps`で利用されるラベルのリストを指定する。標準では列のヘッダが使われる(存在する場合)。

`line`　Gnuplot のスクリプトに、記述内容がそのまま挿入される行を設定する。

`map`　プロットの種別で `3d`もしくは `grid`を指定する場合に、このオプションを `t`にすると、`3d`の立体的な傾斜がわかる表示ではなく、平らな表示（訳注：平面に射影した状態）でプロットする。

`timefmt`　Gnuplot が解釈するような形式に org-mode のタイムスタンプを整える。デフォルトでは、'`%Y-%m-%d-%H:%M:%S`'が使われる。

`script`　Gnuplot をさらに制御するために、プロットに利用されるスクリプトファイル（ファイル名をダブルクォーテーションで囲んでください）を指定できます。プロットを開始する前に、このスクリプトファイルに含まれる`$datafile`の記述は、プロット点を格納するファイルへのファイルパスで置き換えられます。たとえこのオプションを指定しても、プロット種別を制御したいならば、`type`オプションが優先的に適用されます。

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なものです:

`[[リンク][項目名]]`　　または　　`[[リンク]]`

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 (12.5 節「HTML export」p.135 を参照) 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 (4.4 節「Handling links」p.38 を参照)。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ことが多いでしょう。たとえば、以下のように。

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

HTML でのエクスポート（12.5 節「HTML export」p.135 を参照）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つきのアンカーになります[1]。

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

[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。

[2] 見出しへのリンクを挿入するときは、バッファー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。現在のバッファー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。

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。

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

`wl:folder#id`　WANDERLUST のメールメッセージへのリンク
`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`　RMAIL のメールメッセージへのリンク
`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、“char 65.3 節「Adding hyperlink types」p.195 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 (4.1 節「Link format」p.36 を参照)。たとえば、以下のように。

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

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 (12.5 節「HTML export」p.135 を参照) 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、どのバッファー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 のバッファー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、現在のバッファーが何であるかによります。

Org-mode バッファー

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 バッファー内で使うと、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 バッファー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 バッファーでは、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。

ウェブブラウザー: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 バッファー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 (4.7 節「Search options」p.42 を参照) 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。4.8 節「Custom searches」p.43 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 1.2 節「Installation」p.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、バッファー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、バッファー[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。

[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ー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。

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 (4.6 節「Link abbreviations」p.41 を参照) 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るでしょう。もし、*prefix*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) ウェブブラウザーを開きますし、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。

[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。

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。

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
バッファー内の前後のリンクへ移動します。バッファーの端では、移動は、いったんエラーになり、もう一度行うと回り込みます。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 のバッファーでも、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。

---

[6] org-display-internal-link-with-indirect-buffer変数を参照してください。
[7] 対応する#+STARTUPinlineimagesと inlineimagesと共に。

[[リンク語句:タグ][項目名]]

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")))
```

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 バッファー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
```

バッファー内補完 (15.1 節「Completion」p.182 を参照) は、‘[’以降でリンクの省略記法を補完するために使用することができます。*C-c C-l*でリンクを挿入することへの特別な支援 (たとえば、補完) を実行する org-PREFIX-complete-link関数を定義することもできます。このような関数はいかなる引数も受け入れず、接頭辞付きのリンクを返します。

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 (4.4 節「Handling links」p.38 を参照)、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 (4.2 節「Internal links」p.36 を

[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。

参照してください。)。HTML エクスポート (12.5 節「HTML export」p.135 を参照) 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。

`/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イ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無くなり、TODO アイテムを作成するのに使用した全文書が常に最新であることになります。

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

5.1 基本的な TODO の機能

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

```
*** TODO サム フォーチュンに手紙を書く。
```

TODO 項目を入力するときの最も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（10.5 節「Agenda commands」p.107 を参照参照）を入力することで「間接的に」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、5.2.5 節「Per-file keywords」p.47 と 6.2 節「Setting tags」p.57 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。(5.2 節「TODO extensions」p.45 を参照).`shift-selection-mode`との連携については、15.10.2 節「Conflicts」p.192 も参照してください。変数 `org-treat-S-cursor-todo-selection-as-state-change`も参照してください。

C-c / t　　`org-show-todo-key`

ツリーの抽出機能を使って TODO を確認します（2.6 節「Sparse trees」p.12 を参照）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」とその上の見出しの階層を表示します。前置引数をつけることで (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。

[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。

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 (第 10 章「Agenda Views」p.96 を参照) 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。(10.5 節「Agenda commands」p.107 を参照). 10.3.2 節「Global TODO list」p.100 を参照してください. を参照してください。

S-M-RET　org-insert-todo-heading

新しい TODO を現在の位置にある TODO の前に入力します。

TODO の状態の変更は、タグの変更をすることになることがあります。詳細については、オプション org-todo-state-tags-triggersのドキュメント文字列を参照してください。

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 つしかありません。さらに Org-mode は、*TODO* キーワード「org-todo-keywordsに指定されています。」を使って、より複雑に TODO アイテムを分類できます。特別な設定により、TODO キーワードシステムは、ファイルによって異なる働きにすることできます。

注記、タグは見出しと特に TODO アイテムを分類するもう一つの方法です。(第 6 章「Tags」p.57 を参照).

5.2.1 ワークフローの状態としての TODO キーワード

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

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

縦線は、TODO キーワード「アクションが必要な状態」と DONE 状態「アクションが必要ない状態」を分けます。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 から FEEDBACK へ、それから VERIFY、最後に DONE、DELEGATED というように順番に切り替えます。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、すぐに 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、バッファ内補完機能（15.1 節「Completion」p.182 を参照）か特別な一つのキーによる選択により特定のキーワードをバッファに入力することができます。(5.2.4 節「Fast access to TODO states」p.47 を参照)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 5.3.2 節「Tracking TODO state changes」p.50 を参照)

[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。

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、いろいろな種類のやるべき事として表す使用法です。例えば、項目を「仕事」または「家」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 TODO キーワードとして使って直接やるべき事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。:

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

この場合、それぞれのキーワードは作業の順序を表しているのではなく、別々のタイプを表すことになります。そのため、通常の作業の流れとしてタスクを一人に割り振ることになり、その後の DONE になります。Org-mode は、このような形式をサポートするため、「*C-c C-t*」コマンドの動作が少し変化します ([3]) 。まずは、適当なタイプを選択するのに、繰り返し押すことで、順番にキーワードの名称が表示されます。しばらく間をおいてその項目に戻ってきて、「*C-c C-t*」を再度実行すると、そのときは、すぐ DONE に切り替えられます。前置引数か補完を使えば、適当なタイプをすぐに選ぶことができます。さらに *C-c / t*に前置引数を指定することにより、抽出されたツリーの中で特定の TODO タイプの項目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には、グローバルな todo リストを作成し、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もう一つの状態を使用したい場合などです「つまり DONE ではないが、次のアクションが必要ない場合」。その場合の設定は次のようになります:

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

キーワードは、すべて異るようにすべきで、そうすることで Org-mode が、今入力されたキーワードから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こともできます:

C-u C-u C-c C-t
C-S-right
C-S-left　これらのキーは、ある TODO のサブグループから次へとジャンプします。上の例では、*C-u C-u C-c C-t*または、*C-S-right*は、TODOか DONEから REPORT、そして、二番目のサブグループのどれかの状態から CANCELEDへジャンプします。注意として *C-S-*キーバインディングは、`shift-selection-mode` (15.10.2 節「Conflicts」p.192 を参照) と衝突します。

[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。

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に切り替えいきます。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、15.10.2 節「Conflicts」p.192 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、各キーワードの後に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:

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

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

5.2.5 ファイル別にキーワードを設定する

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

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

(解釈を明確にするために#+SEQ_TODOと記入してもいいです。しかし、#+TODOと記入するのと同じです。)、もしくは、

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

同時に複数のキーワードグループの設定には:

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

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

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

---

[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 (6.2 節「Setting tags」p.57 を参照)、この二つを混ぜて使いたいならですが。この場合、それぞれのキーワードセットに単一なキーを準備する必要があります。

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

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えば上手くいくはずですが、いつもうまくいってるように見えないかもしれません。必要であれば、特別なフェイスを定義してそれ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

Org ファイルの構成「階層とリスト」は、TODO の依存関係の定義を容易にします。通常、親 TODO タスクは、すべてのサブタスク「子タスクと定義されている」が終るまでは、DONE にするべきではありません。そして時折、多く「サブ」タスクに対して論理的な順序があるので、あるタスクがその前にあるすべての関連したタスクが終るまで始められないこともあります。もし、変数 org-enforce-todo-dependenciesをカスタマイズすれば、Org は、未完了の子タスクが終わるまで DONE への状態の変更を防ぎます。さらに、もし、あるアイテムに ORDEREDのプロパティが設定されていると、その前の関連したタスクがすべてが DONE になるまで、そのそれぞれの子タスクは、変更できないようになります。ここに例があります:

```
* TODO この TODO は、２番が終るまで変更できない。
** DONE １番
** TODO ２番

* 親
  :PROPERTIES:
    :ORDERED: t
  :END:
** TODO a
** TODO b, (a) が終わるのを待つ必要があります。
** TODO c, (a) と (b) が終わるのを待つ必要があります。
```

C-c C-x o　org-toggle-ordered-property

ORDEREDプロパティを現在のアイテムに対してトグルします。プロパティが、この動作に使われるのは、タグのように継承するのではなく、現在のアイテムに対してのみ動作させるためです。しかし、もし見やすいようにタグを使ってプロパティの値を記録したいのであれば、変数 org-track-ordered-property-with-tag をカスタマイズしてください。

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

変更できない状態のものでも TODO の状態を変更します。

変数 org-agenda-dim-blocked-tasksを設定すれば、依存関係のせいで閉じることのできない TODO エントリーをアジェンダビューで薄暗いフォントにして表示するか、さらに見えなくすることもできます。(第 10 章「Agenda Views」p.96 を参照).

チェックボックスを見ることで TODO の状態の変更を妨げることもできます (5.6 節「Checkboxes」p.54 を参照)。変数 org-enforce-todo-checkbox-dependenciesを設定すれば、チェックされていないチェックボックスをもつエントリーが DONE になるのを妨げることもできます。

もし、より複雑な依存関係の構造が必要であれば、例えば、異なるツリーやファイルのエントリー同士の依存関係、付属モジュールの ‘org-depend.el’を参照してみてください。

5.3 Progress logging

Org-mode は、TODO アイテムに DONE という完了の印をつけたときや、TODO アイテムの状態を変更したときはいつでも、自動的にタイムスタンプとメモを記録をすることができます。かなり柔軟に設定することが可能で、キーワードごとに設定したり、ファイルやサブツリーごとに設定することもできます。タスクの時間管理についての情報は、8.4 節「Clocking work time」p.77 を参照してください。

5.3.1 Closing items

一番基本的な時間の記録機能は、いつTODO アイテムが完了したかを記録することです。これは、次のようにしてください[1]

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

この後、毎回 TODO「未完了」から DONE の状態に移行したとき、見出しの後に‘CLOSED: [タイムスタンプ]’の行が挿入されます。切り替えていく間に一つのエントリーに対して TODO の状態に戻すと挿入された行はまた削除されます。タイムスタンプと一緒にメモも記録したいのであれば、次のようにしてください[2]

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

この設定によりメモの入力を聞いてきます。そのメモは‘Closing Note’という見出しの下に挿入されます。

タイムライン「10.3.4 節「Timeline」p.104 を参照」とアジェンダ「10.3.1 節「Weekly/daily agenda」p.99 を参照」上で、l キー使用して TODO アイテムと‘CLOSED’タイムスタンプを日ごとに表示することができます。何が完了しているかのサマリも提供されます。

5.3.2 Tracking TODO state changes

TODO キーワードがワークフローの状態 (5.2.1 節「Workflow states」p.45 を参照) として使われる時に、いつその状態の変化が起きたか記録し、さらにメモを取りたくなるかもしれません。その場合、タイムスタンプかタイムスタンプ付きメモを記録することができます。これらは、見出しの後に一番新しいものを先頭として[3] 項目ごとにリストされ挿入されます。たくさんのメモをとっている場合、そのメモを引き出しの中に入れて隠したいようになるかもしれません「2.8 節「Drawers」p.16 を参照」。その場合、変数 org-log-into-drawer を編集してください。オススメの引き出しは、LOGBOOKと呼ばれています。さらにサブツリーのために、この変数の設定も LOG_INTO_DRAWERプロパティを修正すれば、その効果を無視して上書きすることができます。

通常、すべての状態でメモを記録するのはやりすぎになるので、Org-mode は、キーワードごとに設定されると想定します。これは、特別なマーカー‘!’「タイムスタンプ用」、‘@’「メモ用」をキーワードの後に括弧に入れることでできるようになります。例えば、以下のようになります。

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

---

[1] これに対応するイン-バッファ定義は:#+STARTUP: logdone

[2] これに対応するイン-バッファ定義は:#+STARTUP: lognotedone

[3] 変数 org-log-states-order-reversedを確認してください

これで、グローバル TODO キーワードとショートカットキーを定義するだけでなく、DONE に状態が変更された際、時間も記録されるようにも定義できます。[4] そして、WAIT か CANCELED に状態が変化したときにメモが記録されます。WAIT の設定は、さらに特別です:斜線の後の '!'は、その状態に最初に切り替わる際に挿入されるメモの記録だけでなく、WAIT の状態から次にの状態に変わる時に、タイムスタンプも記録されます「でもこれは、次の状態が切り替わる時に記録する設定がない場合にのみ有効です」。そのため、WAIT から DONE に切り替わる際には影響がありません、なぜなら DONE は、タイムスタンプを記録するだけとして設定されているからです。しかし WAIT から TODO に戻る場合は、TODO になにも記録するように設定されてなくても、WAIT の '/!'設定が、タイムスタンプを挿入するようになります。

まったく同じ構文を使ってバッファ内のみ有効な設定を使用できます。

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

サブツリーまたは、一つのアイテムだけ局所的にログの設定を定義したい場合は、そのエントリーに LOGGING プロパティを定義してください。空ではない LOGGING プロパティは、すべてのログの設定を nil にリセットします。この後、`lognotedone`か`logrepeat`のような STARTUP キーワード、そして `TODO(!)`のような状態に特化した設定を追加して、特定のツリーに対してログを開始するようにできます。例えば

```
* TODO 各状態のタイムスタンプだけをログを取る
  :PROPERTIES:
  :LOGGING: TODO(!) WAIT(!) DONE(!) CANCELED(!)
  :END:
* TODO WAIT に切り替えられ、さらに繰り返されたときだけログを取る
  :PROPERTIES:
  :LOGGING: WAIT(@) logrepeat
  :END:
* TODO 何もログを取らない
  :PROPERTIES:
  :LOGGING: nil
  :END:
```

5.3.3 習慣の追跡

Org-mode には、「habits(習慣)」と呼ばれる TODO の特別なカテゴリーの一貫性を記録するための機能があります。habit には、以下の性質があります:

1. 変数 `org-modules`をカスタマイズすることで `habits`モジュールを有効にしてある。
2. habit は、TODO 一種であり、TODO キーワードを使い未解決を表します。
3. プロパティ`STYLE`に、`habit`を値として定義してある。
4. この TODO は、スケジュールされた日付があり、通常.+スタイルで繰り返される間隔を表します。++スタイルは、時間制限があるような場合に有効でしょう。例えば、週末にしなければいけないことなどです。+スタイルは、遅れることがあるような通常の習慣ではないような場合「例:週次報告書」に適しています。

[4] Org-mode は、`org-log-done`と状態変化の際の記録機能を使えば、二つのタイムスタンプを記録することも可能です。それでも、二つのメモをするように聞かれることはできません。もし実際に、両方の機能を設定した場合、状態変化の際の記録機能の方が優先されて、'`Closing Note`'は、使われません。

5. この TODO は、最短から最長の期間を‘.+2d/3d’のようなシンタックスで指定できます。この例の場合、このタスクを少なくともで 3 日ごとか、多くて 2 日ごとにこなすと指定しています。
6. 記録されていたデータが一貫性のあるグラフに表記されるように、状態のログを取る DONE を使用できるようにしておく必要があります。使用できるようになっていない場合は、エラーにはなりませんが、一貫性を表すはずのグラフがまったく意味のないものになります。

上記にある定義が実際にはどのようになるか分かってもらうために、ここに経過の記録情報と共に実際の habit があります。

```
** TODO 髭剃り
   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:
```

この habit が表しているのは、髭剃りを多くて 2 日ごとか「SCHEDULEDにより指定された日付と繰り返される間隔」、少なくとも 4 日ごとにする。もし、今日が１５日とすると、この habit は、二日後の１０月１７日にアジェンダに表示されて、4 日後の１９日には、期限切れとして表示されます。

habits の本当に使い易いところは、定期性グラフと表示されることです。これは、過去にどれぐらいタスクが定期的に完了したかを見るためのものです。このグラフは、毎日過去 3 週間のタスクが完了したかを色分けして表示します。各色は、以下を表します:

青　　まだ、その日までにタスクが完了してなくていい場合。

緑　　その日に完了できたはずのタスクの場合。

黄　　明日になると期限切れになるタスクの場合。

赤　　その日に期限切れのタスクの場合。

日ごとの色分けだけでなく、その日に終わったタスクに関してはアスタリスクでマークされ、感嘆符が、グラフ中の今日の日付の部分に付きます。

アジェンダ上で habits が表示方法を変える幾つかの設定変数があります。

`org-habit-graph-column`
: 定期的グラフを表記させるバッファ列。これにより指定された列にある文字列を上書きします。そのため、habit のタイトルを短く要点を付くようにするといいでしょう。

`org-habit-preceding-days`
: 今日より前に、定期的グラフに何日分の日付表示するかの指定。

`org-habit-following-days`
: 今日より後に、定期的グラフに何日分の日付表示するかの指定。

`org-habit-show-habits-only-for-today`
: nil 以外が指定されている場合、habits を今日のアジェンダビューだけに表示する。これは、初期値で真に設定されています。

最後に、アジェンダバッファで *K*を押すことにより habits を一時的に使用不可にし、まったく表示させないようにできます。もう一度 *K*を押すと元にもどります。例えば、もし特定の前後の内容によてのみ必要な habits の場合は、タグのフィルタリングの影響も受けます。

5.4 Priorities

もし、Org-mode よく使うのであれば、最終的に TODO アイテム量が増え、優先順位付けをした方がいいとなるかもしれません。優先順位付けは TODO アイテムの見出しに、次のように優先順位クッキーを置くことで可能になります:

```
*** TODO [#A] サム フォーチュンに手紙を書く。
```

初期値として Org-mode は、次の 3 つの優先順位付けをサポートします:‘A’、‘B’、‘C’。‘A’が一番高い優先度です。クッキーなしの場合は、‘B’の優先度として扱われます。優先順位付けは、アジェンダの順番を付けるときのみ影響します。「10.3.1 節「Weekly/daily agenda」p.99 を参照アジェンダ外では、Org-mode で継承されたりしません。クッキーは、変数`org-priority-faces`をカスタマイズすることにより、特別なフェイスを使ってハイライトすることもできます。

優先順位付けどんなアウトラインモードにも付けるとができます。つまり TODO アイテムである必要はありません。

C-c ,
: 現在の見出しの優先順位付けをする「`org-priority`」。このコマンドは、優先順位付けのための文字‘A’、‘B’、または、‘C’を聞いてきます。その代わりに `SPC`を押すと、優先順位付けのクッキーが見出しから削除されます。優先順位は、タイムラインまたは、アジェンダバッファからも`,`コマンドで遠隔的に変更できます。「10.5 節「Agenda commands」p.107 を参照」。

S-up `org-priority-up`
S-down `org-priority-down`
: 現在の見出し優先度を上下する。[5]. これらのキーはタイムスタンプを修正するのにも使うので注意してください。「8.2 節「Creating timestamps」p.71 を参照」`shift-selection-mode`との相互利用に関しては次を参照してください。15.10.2 節「Conflicts」p.192

変数`org-highest-priority`、`org-lowest-priority`、`org-default-priority`を設定することで、変更できる優先度の範囲を変えることができます。バッファごとに、「上限、下限、既定値」の設定を以下のようにすることができます。「必ず一番上の優先度の文字が、一番下の優先度よりもアルファベットの並びで前の文字であるようにしてください。」:

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

[5] 次のオプションも参照してください。`org-priority-start-cycle-with-default`.

5.5 タスクをサブタスクに細分化する。

通常、大きなタスクは小さくて管理しやすいサブタスクに細分化することをお勧めします。これは TODO アイテムの下にアウトラインツリーをさらに詳細なサブタスクをそのツリーを付けて作成することで可能です。[6] すでに完了したサブタスクの進捗状況を表示してさせておくには、‘[/]’か‘[%]’ヘッドラインのどこかに挿入してください。これらのクッキーは、サブタスクの TODO の状態が変わるかクッキー上で *C-c C-c* を押すたびに更新されます。例えば:

```
* パーティーの準備をする [33%]
** TODO 出席者に電話する [1/2]
*** TODO ピーター
*** DONE サラ
** TODO 食べ物を買う
** DONE 近所の人と話す
```

もし、見出しがチェックボックスと子 TODO を両方持っていた場合、統計クッキーは、あいまいなものになります。この問題を解決するには、COOKIE_DATA プロパティを‘checkbox’か‘todo’に設定してください。

もし統計クッキーに「直下の TODO だけでなく」すべてのサブツリーの TODO エントリーを含めたい場合は、変数 org-hierarchical-todo-statistics を設定してください。これを一つのサブツリーに行うには、‘recursive’というキーワードを COOKIE_DATA プロパティの値として設定してください。

```
* 親キャプチャ統計 [2/20]
  :PROPERTIES:
  :COOKIE_DATA: todo recursive
  :END:
```

もしすべての子タスクが終了後、TODO エントリーを自動的に DONE に切り替えたい場合は、以下の設定を行なってください:

```
(defun org-summary-todo (n-done n-not-done)
  "すべてのサブツリーが終了すると DONE に切り替え、その他の場合は、TODO になする。"
  (let (org-log-done org-log-states)   ; 記録「logging」を終了
    (org-todo (if (= n-not-done 0) "DONE" "TODO"))))
(add-hook 'org-after-todo-statistics-hook 'org-summary-todo)
```

その他の方法としては、チェックボックスをつかって「階層化された」多量のサブタスクがいくつあるか調べることもできます。「5.6 節「Checkboxes」p.54 を参照」

5.6 Checkboxes

プレーンなリスト[7]「2.7 節「Plain lists」p.13 を参照」は、‘[ ]’で開始することでチェックボックスにすることができます。この機能は、TODO アイテムに似ていてますが「第 5 章「TODO Items」p.44 を参照」、より気軽につかえます。チェックボックスは、グローバル TODO リストに追加されません。そのためタスクを分岐するのに便利です。もしくは、買い

[6] サブタスクをグローバル TODO リストに含めないようにするには、org-agenda-todo-list-sublevels を参照してください。

[7] これは概要リスト以外という意味ですが、このリストも org-list-automatic-rules を修正することで可能です。

物リストに使えます。チェックボックスをチェックした状態するには、*C-c C-c*を使ってください。もしくは、マウスでクリックしてください。「Piotr Zielinski の ‘org-mouse.el’に感謝」。

以下は、チェックボックスリストの例です:

```
* TODO パーティの準備 [2/4]
  - [-] みんなに連絡 [1/3]
    - [ ] ピーター
    - [X] サラ
    - [ ] サム
  - [X] 食べ物を注文
  - [ ] どんな音楽を掛けるか考える
  - [X] 近所の人と話す
```

チェックボックスは、階層化に対応しています。そのため、もしあるチェックボックス下の項目が複数のチェックボックスである場合、チェックボックスが全くチェックされていないか、いくつかされているか、全てされているかによりその内の一つのチェックボックスをチェックした状態にすると親チェックボックスに影響します。

最初と二列目の ‘[2/4]’と ‘[1/3]’はクッキーであり、いくつのチェックボックスがそのエントリーでチェックされているかとそのすべてのチェックボックス数が表示されています。これにより、いくつのチェックボックスが残っているか、折りたたまれていても分かるようになっています。クッキーは見出しかプレインなリスト「の最初の列」に置くことができいます。各クッキーは、クッキーのある見出し／項目の直下にある子構造であるチェックボックスを表しています。[8] 自分でクッキーを ‘[/]’もしくは、‘[%]’をタイプして入力しなければなりません。‘[/]’を入力すると上記したように、‘m 個の内の n 個’となります。‘[%]’の場合は、何パーセントのチェックボックスがチェックされているかが情報として得られます。「上記の例では、それぞれ ‘[50%]’と ‘[33%] となります」’. 見出しの下では、クッキーは、見チェックボックスか子 TODO の状態の数を数えます。また、最後の変更に基づいて表示されます。この問題を解決するには、プロパティ COOKIE_DATA を ‘checkbox’か ‘todo’に設定してください。

アウトラインノードに ORDERED プロパティが設定されている場合、チェックボックスは、連続でチェックされていなければなりません。上部のチェックボックスがチェックされていない状態でその下部のチェックボックスをチェックしようとするとエラーがスローされます

以下のコマンドでチェックボックスを操作できます:

C-c C-c　　org-toggle-checkbox

チェックした状態をトグルするか「前置引数と実行すると」チェックボックスを作成します。ダブル前置引数だと、‘[-]’ が設定されます。これは、中間の状態を表します。

C-c C-x C-b　　org-toggle-checkbox

チェックした状態をトグルするか「前置引数と実行すると」チェックボックスを作成します。ダブル前置引数だと、‘[-]’が設定されます。これは、中間の状態を表します。

– アクティブなリージョンがある場合は、そのリージョンの最初のチェックボックスをトグルします。そして残りのボックスを最初のボックスと同じ状態にします。前置引数と使用すると、リージョン内のすべてのチェックボックスを作成するか削除します。

[8] もし直下だけでなくクッキーのしたにあるすべてのチェックボックスを網羅したい場合は、変数 org-hierarchical-checkbox-statistics を設定してください。

- カーソルが見出し上にある場合、現在の見出しから次の見出しのリージョン内のチェックボックスをトグルします。「つまりサブツリー全体ではない」
- アクティブなリージョンがなければ、その場所のチェックボックスをトグルします。

M-S-RET `org-insert-todo-heading`
新しい項目をチェックボックスと共に挿入します。これは、プレインなリスト (2.7 節「Plain lists」p.13 を参照) 内にカーソルがすでにある場合にのみ動作します。

C-c C-x o `org-toggle-ordered-property`
ORDEREDプロパティを設定します。これは、連続でチェックボックスがチェックされていなければならないと指定します。プロパティが使用されます、なぜならこの指定はローカルに影響するべきでタグのように継承されるないからです。しかし、見やすいようにプロパティの値をタグを使って記録したい場合は、`org-track-ordered-property-with-tag`変数をカスタマイズしてください。

C-c # `org-update-statistics-cookies`
現在のアウトライン内の統計クッキーを更新します。*C-u*引数と呼び出されるとファイル全体を更新します。*C-c C-c*でチェックボックスをトグルした場合と *M-S-RET*で新しいチェックボックス項目が作成された場合、チェックボックス統計クッキーは、自動的に更新されます。TODO 状態を変更すると統計クッキーも更新されます。手動でチェックボックスや項目を削除したり、それらを追加したり変更した場合は、このコマンドをつかって状態を更新してください。もしくは、単にコマンドを二度トグルしてください「*C-c C-c*チェックボックスを作成など」。

6 Tags

相互に関係する情報のためにコンテキストやラベルをつけるためのすばらしい方法の一つは見出しにタグを対応づけることです。Org-mode はタグについて幅広く対応しています。

全ての見出しはタグのリストを取ることができて、タグは見出しの最後に置かれます。タグはいわゆる普通の単語で利用される文字や数字、‘_’、‘_’を利用できます。タグはコロンで始まりコロンで終わらなければなりません。例えば、‘:work:’です。複数のタグはこのように書きます。‘:work:urgent:’。タグはデフォルトでは太字のフェイスで見出しと同じ色で表示されます。TODO のキーワード (5.2.6 節「Faces for TODO keywords」p.48 を参照) と同じように変数 org-tag-facesを変更することで特別なフェイスを設定することも可能です。

6.1 Tag inheritance

タグはアウトラインツリーの階層構造を利用します。もしある見出しに特定のタグがついていれば、全ての下位レベルにタグが継承されます。以下に例を示します:

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

最後の見出しには明示的にマークされていない、‘:work:’、‘:boss:’、‘:notes:’、‘:action:’のタグがついています。ファイル全体の見出しに継承される見出しをつけることもでき、これは仮定的にレベル 0 の見出しとして定義され、ファイル全体に反映させることができます。この例を次に示します[1]:

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

特定のタグについてタグの継承を制限するには変数 org-use-tag-inheritance、org-tags-exclude-from-inheritanceを利用します。

タグの継承が有効の場合にタグの検索で見出しが一致したとき、その見出しの全てのサブツリーも同じように一致します[2]。始めにタグが一致したサブツリーのみを表示したいのであれば、変数 org-tags-match-list-sublevelsを設定してください (非推奨).

6.2 Setting tags

簡単に見出しの最後へタグは入力することができます。コロンの次に *M-TAB*でタグの補完をします。他にもタグの入力のための特別なコマンドが以下に続きます:

C-c C-q　　org-set-tags-command

現在の見出しに新しくタグを入力します。org-mode は補完を始めるか、特別なワンキーのインタフェースを提示します。ワンキーのインタフェースは後で説明します。タグを入力し、RETキーをタイプするとタグが挿入され、org-tags-column の列に整列されます。数引数 (*C-u* prefix) をつけて呼び出すと、カレントバッファの全てのタグがきれいに見えるように整列されます。TAGS は自動的に昇格や降格した後に再整列され、TODO の状態が変わります (5.1 節「TODO basics」p.44 を参照)。

---

[1] これら全てのバッファ内の設定は、*C-c C-c*とタイプすることでその行の変更を有効にすることができる

[2] タグ以外の条件を加えて検索するとこの限りではない (これはもしその検索が、より複雑な属性を含む条件を伴わないときだけ正しい)(7.3 節「Property searches」p.63 を参照)

C-c C-c　org-set-tags-command

カーソルが見出し行にある時は *C-c C-q*と同じ動作をします。

Org-mode は *list of tags* に基づくタグの挿入をサポートしています。このリストはデフォルトでは動的に構築され、バッファで使われているタグを全て含みます。変数 org-tag-alist にタグのリストを設定することでグローバルにタグを指定しておくこともできます。デフォルトのタグを次のようにファイルに記述することでもデフォルトのタグリストを設定できます。

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

もし変数 org-tag-alistにタグを設定することでタグリストがグローバルに定義されていたとして、そのリストよりもファイルから生成された動的なタグリストを利用したければ、次のような空の TAGS オプションをファイルに指定します

```
#+TAGS:
```

もし全てのファイルで使うつもりのタグや、ファイルごとの TAGS オプションで定義されたタグの集合を優先したければ、変数 org-tag-persistent-alistに明記すればよい。これを止めるにはファイルごとに STARTUP オプションを次のように書きます。

```
#+STARTUP: noptag
```

デフォルトで Org-mode はタグの入力に標準的なミニバッファでの補完を使います。しかし *fast tag selection* と呼ばれるもう一つの速い実装もあります。これはタグの選択や除外をワンキーで可能にするものです。これをうまく動かすためにあなたはユニークな文字を良く使うタグにアサインするべきです。変数 org-tag-alistを ‘.emacs’に設定することでグローバルにこの機能を使うことができます。例えば複数のファイルの多くの項目に ‘:@home:’タグをつけるという需要があったとします。この場合次のような設定なるでしょう。

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

もしそのタグが作業中のファイルにおいてのみふさわしいのであれば、かわりに次のように TAGS オプションを書くこともできます。

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

タグインタフェースはスプラッシュウィンドウにとりうるタグを表示します。もし特定のタグの後に新しい行から始めたいのであればタグリストに ‘\n’を挿入します。

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

もしくは 2 行に分けて書きます。

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

また次のようにブレース (波括弧{}) を使うことで相互排除したグループにタグをまとめることもできます。

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

これは ‘@work’、‘@home’、‘@tennisclub’のうち少なくとも一つは選択されることを意図しています。そのようなグループを複数持つことも可能です。

これらの変更を有効にするために、その行にカーソルを置いて C-c C-cを押すことを忘れないようにしてください。

これらの相互排除グループを変数 org-tags-alistで設定するためには、ブレースの代わりにダミーのタグとして:startgroupと:endgroupを使わなければなりません。同様に改行を表現するためにダミーのタグとして:newlineを使うことができます。直前の例を設定すると次のような記述になります。

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

もし少なくとも一つのタグに選択用キーが設定されていると、*C-c C-c*が押されたときに自動的に特別なインタフェースが表示されます。そのインタフェースは継承されたタグ、現在の見出しのタグ、全ての有効な付随するタグを提示します[3]。このインタフェースでは以下のキーが利用できます。

`a-z...`　アサインされたキーを押すことで現在の行にタグが追加または削除されます。相互排除グループのタグを選択することでそのグループの他のタグは無効になります。

`TAB`　たとえあらかじめ定義されていないタグでも、ミニバッファでタグを入力します。バッファ内の全てのタグを補完することができます。コンマで区切ることで複数のタグを追加することも可能です。

`SPC`　この行のタグを全てクリアします。

`RET`　変更された集合を確定します。

C-g　変更を破棄します。

q　もし *q*がタグにアサインされていなければ *C-g*のように変更を破棄します。

!　相互排除しているタグのグルーピングを無効にします。これはそのグループ内のタグを (例外的に) 複数個、タグづけるときに使います。

C-c　次の変更後の自動終了をトグルします (下記参照)。もしもしエキスパートモードを使っているのであれば、最初の *C-c*が選択ウィンドウに表示されます。

このメソッド (特別なインタフェース) であなたはとても少ないタイプで見出しにタグをつけられます。上記の設定で、現在のタグをクリアして、‘`@home`’、‘`laptop`’、‘`pc`’のタグをつけるには次のように入力します: *C-c C-c SPC h l p RET*。タグ ‘`@home`’を ‘`@work`’へ付け替えるには *C-c C-c w RET*とタイプするか代わりに *C-c C-c C-c w*とタイプします。定義されていないタグ ‘`Sarah`’を追加するには *C-c C-c TAB S a r a h RET RET*すればよいのです。

もしタグを決定するのをただひとつのキーを押すだけにする必要があれば、変数`org-fast-tag-selection-single-key`を設定します。これによってタグを選択するキーを押した後に *RET*を押す必要がなくなります。もしキー入力が必要であれば *C-c*で無効化できます (つまりタグの選択を始めるために *C-c C-c C-c*の代わりに *C-c C-c*を使います)。もしこの変数に`expert`を設定すれば、タグのワンキー選択用の特別なウィンドウを表示しなくなります。*C-c*を追加してはじめてそのウィンドウが表示されるようになります。

6.3 Tag searches

一度タグシステムが設定されると、関連する情報を特殊なリストに集めるのに使われます。

C-c / m or *C-c *　`org-match-sparse-tree`
タグ検索にマッチした全てのツリーを抽出する。*C-u*のプレフィックスをつけて呼び出すことで、TODO の見出しのみに限定する。

[3] キーが設定されていないタグにはは自動的にキーがアサインされます。

C-c a m　org-tags-view
全てのアジェンダファイルにおいてタグにマッチしたものの一覧を作成する。10.3.3 節「Matching tags and properties」p.101 を参照してください.

C-c a M　org-tags-view
全てのアジェンダファイルにおいてタグにマッチしたものの一覧を作成する。ただし TODO の項目に限定し、サブツリーの項目の検索を強制する (変数 org-tags-match-list-sublevels参照)。

これらのコマンドは全てマッチ文字列を問います。マッチ文字列はタグ ‘boss’と ‘urgent’ を含み、‘project1’を含まないものを検索する ‘+boss+urgent-project1’や、タグ ‘Kathy’または ‘Sally’がつけられているエントリーを検索する ‘Kathy|Sally’ のような基本的な論理構造が使用可能です。検索文字列の全ての構文はリッチで TODO キーワードやエントリーのレベル、プロパティにもマッチします。完全な説明と多くの例は 10.3.3 節「Matching tags and properties」p.101 を見てください。

7 プロパティ（属性）とカラム（列）

プロパティはエントリーに関連付けられたキーと値を持つペアのあつまりです。Org-mode では、プロパティのための主要なアプリケーションが 2 つあります。1 つ目は、プロパティはタグのようですが値を持つことです。2 つ目は、Org-mode のバッファで (とても基本的な) データベース機能を実装するためにプロパティを使えることです。1 つ目のアプリケーションの例に、ソフトウェアのリリース計画とバグを文章化するファイルを管理することを想像して下さい。`:release_1:`、`:release_2:`のようなタグを使う代わりに、`:Release:`というプロパティを使い、異なるサブツリーの中に `1.0`や `2.0`のような異なる値を持たせれば良いのです。プロパティの 2 つ目のアプリケーションの例に、音楽 CD のトラックを管理する事を想像して下さい。そこではアルバム名、アーティスト名、リリース日、トラックの数などがプロパティとなるでしょう。

プロパティはカラムビューで便利に編集、閲覧できます (7.5 節「Column view」p.64 を参照)。

7.1 Property syntax

プロパティはキーと値のペアです。それらは、名前 `PROPERTIES`を持つ特別な引き出し (2.8 節「Drawers」p.16 を参照) の中にある必要があります。各プロパティは最初に (コロンで囲われた) キーを持ち、その後に値を持つ 1 行で記述されます。以下に例を示します。

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

プロパティ‘`:Xyz_ALL:`’のように設定する事で、特定なプロパティ‘`:Xyz:`’の許容値を定義できます。この特別なプロパティは、もしレベル 1 のエントリに設定されたならば全てのツリーに適用されるように継承されます。許容値を定義すると、対応するプロパティの設定が簡単になり、入力ミスを防ぐ事ができます。CD コレクションの例では、以下のように 1 つのボックスの中に発売元とディスクの数をあらかじめ定義できます。

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

1 つのファイル全体で継承されるプロパティを設定したいならば、以下の行のようにします。

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

グローバル変数 `org-global-properties`に設定するプロパティの値は全ての Org-mode のファイルに継承されます。

以下のコマンドはプロパティを操作する助けとなります。

M-TAB pcomplete
行にある最初のコロンのうしろで、プロパティのキーを補完します。現在のファイルで使われた全てのキーは可能な補完候補として提供されます。

C-c C-x p org-set-property
プロパティを設定します。プロパティ名と値の入力を促します。必要なら、プロパティの引き出しがさらに作られます。

M-x org-insert-property-drawer
現在のエントリーの中にプロパティの引き出しを入れます。引き出しはエントリーのはじめに入りますが、デッドラインのような計画情報を持つ行の後となります。

C-c C-c org-property-action
プロパティの引き出しの中にカーソルがあるとき、プロパティコマンドを実行します。

C-c C-c s org-set-property
現在のエントリにプロパティを設定します。プロパティと値の両方共、補完を使って入力できます。

S-right org-property-next-allowed-value
S-left org-property-previous-allowed-value
ポイントのプロパティを次/前の許容値に切り替えます。

C-c C-c d org-delete-property
現在のエントリからプロパティを削除します。

C-c C-c D org-delete-property-globally
プロパティを現在のファイルにある全てのエントリからグローバルに削除します。

C-c C-c c org-compute-property-at-point
ポイントにあるプロパティを最も近い列のフォーマット定義からオペレータやスコープを使って計算します。

7.2 Special properties

以前の章で述べた TODO 状態や、エントリの優先度のようなスペシャルプロパティは、Org-mode の機能への別のアクセス方法を提供します。このインターフェイスはカラムビュー (7.5 節「Column view」p.64 を参照) にそれらの状態を含めたり、クエリにそれらを使ったりする事で生じます。次のプロパティ名は特別 (`:CATEGORY:`を除いて) で、プロパティの引き出しでキーとして使われません。

`TODO`	エントリの TODO キーワード
`TAGS`	見出しに直接定義されたタグ
`ALLTAGS`	継承されたタグも含む全てのタグ
`CATEGORY`	エントリのカテゴリ
`PRIORITY`	1 文字の文字列である、エントリの優先度
`DEADLINE`	山括弧 (<>) のないデッドライン時刻文字列
`SCHEDULED`	山括弧 (<>) のないスケジュールタイムスタンプ
`CLOSED`	いつこのエントリがクローズされたか
`TIMESTAMP`	エントリで最初のキーワードのないタイムスタンプ

`TIMESTAMP_IA` エントリで最初のアクティブでないタイムスタンプ
`CLOCKSUM` サブツリーでの CLOCK インターバルの合計。`org-clock-sum\n` が値を計算するために最初に実行されなければならない。\n
`BLOCKED` "t" であれば、タスクが子供や兄弟に現在ブロックされている
`ITEM` エントリの内容
`FILE` エントリのあるファイル名

7.3 Property searches

プロパティに基いて選択した特別なリストやツリーの抽出を作成するために、タグ検索 (6.3 節「Tag searches」p.59 を参照) の場合と同じコマンドが使えます。

C-c / m or *C-c * `org-match-sparse-tree`
全てのマッチしたエントリについて抽出したツリーを作成します。前置引数 *C-u* をつけると、TODO 行でない見出しは無視します。

C-c a m `org-tags-view`
全てのアジェンダファイルからマッチするタグ・プロパティのグローバルなリストを作成します。10.3.3 節「Matching tags and properties」p.101 を参照してください.

C-c a M `org-tags-view`
全てのアジェンダファイルからマッチするタグのグローバルなリストを作成します。しかし、TODO 項目と下位項目の強制チェック (変数 `org-tags-match-list-sublevels`参照) のみチェックします。

検索文字列のための文法は、10.3.3 節「Matching tags and properties」p.101 で説明されています。

1 つのプロパティに基いて抽出したツリーを作成するための特別なコマンドもあります。

C-c / p プロパティの値に基いて抽出したツリーを作成します。これは最初にプロパティ名、次にその値の入力を促します。抽出したツリーは与えられた値でこのプロパティを定義する全てのエントリで作られます。その値を大括弧で括っていたならば正規表現として解釈され、プロパティ値に対してマッチします。

7.4 プロパティの継承

Org-mode 文章のアウトライン構造はプロパティの継承モデルを適用しています。ツリーの親があるプロパティを持っているならば、子はこのプロパティを継承します。Org-mode はこれをデフォルトで有効としていません。これはプロパティの検索を遅くしてしまうためとあまり必要とされないためです。しかしながら、継承が役立つ場面があるならば、変数 `org-use-property-inheritance`を設定する事で有効とできます。この変数は、全てのプロパティを親から継承する `t`、継承されるべきプロパティのリスト、継承されるプロパティにマッチする正規表現を設定できます。もしプロパティが値 '`nil`'を持つならば、継承検索がこの値で停止し `nil`を返すように、明示的に未定義のプロパティであると解釈されます。

Org-mode は、継承がハードコードされているプロパティがいくつかあります。少くともそれらを使う特別なアプリケーションがあります。

`COLUMNS` `:COLUMNS:`プロパティは、カラムビュー (7.5 節「Column view」p.64 を参照) のフォーマットを定義します。`:COLUMNS:`プロパティが定義されているレベルがカ

ラムビューテーブルの開始地点として使われるという意味で継承されます。そして、列表示が有効である場所とサブツリーの場所とは無関係です。

CATEGORIES
: アジェンダビュー用です。:CATEGORY:プロパティを通して設定されたカテゴリがサブツリー全体に適用されます。

ARCHIVE
: アーカイブ用です。:ARCHIVE:プロパティは、サブツリー全体のアーカイブ位置を定義します (9.6.1 節「Moving subtrees」p.93 を参照)。

LOGGING
: LOGGING プロパティは、エントリやサブツリーのログ取得設定について定義します (5.3.2 節「Tracking TODO state changes」p.50 を参照)。

7.5 Column view

アウトラインツリーにあるプロパティを閲覧、編集するための最も良い方法はカラムビューです。カラムビューでは、各アウトラインノードがテーブル行に変換されます。このテーブルにある列がエントリのプロパティへのアクセスです。Org-mode は、各項目の見出し上にテーブル構造をオーバレイすることで列を実装します。見出しはテーブル列に変換されますが、アウトラインツリーの見た目もまだ変えられます。例えば、CONTENTS ビュー (*S-TABS-TAB*、もしくは、カラムビューがアクティブであるときに単純に *c*) にスイッチする事でコンパクトなテーブルを得られますが、まだ各見出し以下のエントリを開いたり、読んだり、編集したりもできます。また、ツリーの抽出コマンドを実行した後にカラムビューに切り替える事もでき、この方法では選択した項目のみのテーブルを得ます。カラムビューは利用可能な複数のファイルから選択した項目を集めたクエリのあるアジェンダバッファ(第 10 章「Agenda Views」p.96 を参照) でも動作します。

7.5.1 Defining columns

最初にカラムビューを設定するために、カラムを定義する必要があります。これはカラムフォーマット行を定義する事によってなされます。

7.5.1.1 Scope of column definitions

カラムフォーマットを定義するために、次のように行を記載します。

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

指定したツリーに適用するだけのフォーマットを指定するために、:COLUMNS:プロパティをそのツリーの一番上のノードに追加します。例えば、

```
** カラムビューの一番最初のノード
   :PROPERTIES:
   :COLUMNS: %25ITEM %TAGS %PRIORITY %TODO
   :END:
```

もし:COLUMNS:プロパティがエントリに現れると、そのエントリ自身とそれ以下のサブツリー全体の列を定義します。列定義は文章の階層構造の一部なので、全てのサブレベルに十分一般的なレベル 1 の列を定義でき、また、ツリーのより深い部分を編集するとき、より下位の指定された列を定義できます。

7.5.1.2 Column attributes

列定義は、列の属性の集りです。一般的な定義は以下のようになります。

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

パーセントとプロパティ名を除いて、全ての項目はオプションです。個々のパーツは次の意味を持ちます。

width　列の幅を文字数で指定する整数。
省略すると、幅は自動的に決定されます。

property　この列で編集できるプロパティ。
メタデータを示す特別なプロパティがここで許容されます。
(7.2 節「Special properties」p.62 を参照)

title　列の見出しテキスト。省略するとプロパティ名が使われます。

{*summary-type*}　サマリタイプ。
指定すると親ノードの列の値は子から計算されます。
サポートされるサマリのタイプは以下です。

{+}	この列にある数の和。
{+;%.1f}	'+'と似ているが、フォーマットが'%.1f'となる。
{$}	通貨。'+;%.2f'の略。
{:}	時刻の合計。HH:MM。数値は時間。
{X}	チェックボックスの状態。 子が全て'[X]'ならば、'[X]'です。
{X/}	チェックボックスの状態。'[n/m]'。
{X%}	チェックボックスの状態。'[n%]'。
{min}	列の最も小さい数値。
{max}	最も大きい数値。
{mean}	数の算術平均。
{:min}	列に最も小さい時間数値。
{:max}	最も大きい時間数値。
{:mean}	時刻の算術平均。
{@min}	最小時刻 (日/時間/分/秒)。
{@max}	最大時刻 (日/時間/分/秒)。
{@mean}	時刻の算術平均 (日/時間/分/秒)。
{est+}	低-高見積りを追加。

含めるあらゆるプロパティに 1 つのサマリタイプしかを持てないという事に気をつけて下さい。同じプロパティを参照する後にくる列は全て同じサマリの情報を表示します。

est+サマリタイプはもう少し説明が必要です。それは低-高の幅で表現される見積りを組み合わせるために使われます。例えば、特定のタスクが 5 日必要であると見積る代わりに、どのくらいの仕事が必要とされるか公正に確実に見積もるならば 5 日から 6 日と、それをなすのに必要な時間が本当に分からないならば、1-10 日と見積るでしょう。両者の幅の平均は 5.5 日ですが、前者はより予測可能な発言を示しています。

そのような見積りを組み合わせるとき、単純に低と高を追加すると非現実的な幅の結果を作ります。代わりに、est+は合計から最終的な見積りを生成する事で統計的意味やサブタスクの分散を追加します。例えば、10 個のタスクがあるとき、各々が 0.5 から 2 日の仕事であると見積られました。全てが非常によく進む、もしくは悪く進むと期待する事による計算で、ストレートな追加は 5 日から 20 日の見積りであると生成します。対照的に、est+は全ての仕事より現実的に 10 日から 15 日であると見積ります。

以下は許可値にそって列定義を計算する例です。

```
:COLUMNS:  %25ITEM %9Approved(Approved?){X} %Owner %11Status \[1]
                  %10Time_Estimate{:} %CLOCKSUM
:Owner_ALL:    Tammy Mark Karl Lisa Don
:Status_ALL:   "In progress" "Not started yet" "Finished" ""
:Approved_ALL: "[ ]" "[X]"
```

最初の列、‘%25ITEM’はその項目の 25 文字を意味します。すなわち見出しのです。おそらく常に‘ITEM’指示子を持つ列定義をはじめるべきでしょう。その他の指示子は許容値の名前のリストを持つ列‘Owner’、4 つの異なる利用可能な値を持つ‘Status’、チェックボックスフィールドを持つ‘Approved’を生成します。‘%’文字の後に幅が与えられていないとき、列は全ての値を完全に表示するために必要な幅と同じぐらい正確に広くなります。‘Approved’列は修正れたタイトル (クエスチョンマークのある‘Approved?’) を持っています。サマリは、HH:MM のように表現する持続時間を追加することで‘Time_Estimete’を、全ての子がチェックされているなら‘[X]’を提供することで‘Approved’を作ります。‘CLOCKSUM’は特別で、サブツリーにある CLOCK インターバルの合計をリストします。

7.5.2 Using column view

カラムビューのオン・オフ

C-c C-x C-c　　org-columns

カラムビューを有効とします。カーソルがそのファイルの最初の見出しより前にあるならば、カラムビューは#+COLUMNS定義を使う事によりファイル全体に対して有効となります。カーソルがアウトラインの内側のどこかにあるならば、このコマンドはフォーマットを定義する:COLUMNS:プロパティをポイントから上部の階層に向かって検索します。1 つ見つかったとき、カラムビューテーブルは:COLUMNS:プロパティを含むエントリではじまるツリー用に設立されます。そのようなプロパティが見付からないときは、フォーマットは#+COLUMNS行もしくは、変数 org-columns-default-formatから取得され、カラムビューは現在のエントリとそのサブツリーのために設置されます。

r　　org-columns-redo

バッファにある最近作られた変更を反映するためにカラムビューを再生成します。

g　　org-columns-redo

*r*と同じです。

q　　org-columns-quit

カラムビューを抜けます。

値を編集する

left right up down

フィールドからフィールドへカラムビューを通じて移動します。

S-left/right

フィールドの次と前の許可値を切り替えます。このために、プロパティの指定された許可値を持つ必要があります。

---

[1] COLUMNS 定義
は 1 行でなければならないことに注意して下さい。書式の制約のためここでラップされます。

1..9,0　直接に N 番目の許可値を選択します。*0*は 10 番目の値を選択します。

n　org-columns-next-allowed-value
p　org-columns-previous-allowed-value
*S-left/right*と同じです。

e　org-columns-edit-value
ポイント下のプロパティを編集します。特別なプロパティでは通常そのプロパティを変更するために使うのと同じインタフェースを呼び出します。例えば、TAGS プロパティを編集するとき、タグ補完や高速選択インタフェースがポップアップします。

C-c C-c　org-columns-set-tags-or-toggle
チェックボックスがあるならば、それを切り替えます。

v　org-columns-show-value
このプロパティの完全な値を表示します。列の幅がその値よりも小さいときに便利です。

a　org-columns-edit-allowed
このプロパティの許可値のリストを編集します。リストが階層で見つかるならば、修正された値はそこに保存されます。リストが見つからないならば、新しい値は現在のカラムビューの一部である最初のエントリに保存されます。

テーブル構造を編集する

<　org-columns-narrow
>　org-columns-widen
1 文字分列を狭く・広くする

S-M-right　org-columns-new
現在の列の左に新しい列を挿入する。

S-M-left　org-columns-delete
現在の列を削除する。

7.5.3 カラム表示の保存

カラムビューはバッファへのオーバレイのみなので、直接にエクスポートや印字ができません。カラムビューをキャプチャしたいならば、`columnview`ダイナミックビュー (“char 65.6 節「Dynamic blocks」p.202 を参照) を使って下さい。このブロックのフレームは以下のように見えます。

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

#+END:
```

ダイナミックブロックを更新するには以下のコマンドを使用します。

`:id`　これは最も重要なパラメータです。カラムビューはある (サブ) ツリーによくローカライズされる機能であり、キャプチャブロックはファイル内の異なる位置にあるかもしれません。キャプチャするビューをもつツリーを識別するために、4 つの値を使えます。

`local`　キャプチャブロックに位置するツリーを使います。
`global`　そのファイル内の全ての見出しを含む、
グローバルビューを作ります。
`"file:`*`path-to-file`*`"`
このファイルの一番上のカラムビューを実行します。
`"`*`ID`*`"`　値 *label*である`:ID:`プロパティを
持つツリーにあるカラムビューを呼び出します。現在の
エントリ用にグローバルにユニークな ID を作り、
kill-ring にそれをコピーするために
*`M-x org-id-copy`*を使えます。

`:hlines`　tのとき、全ての行の後に横線を挿入します。数値 *N*のとき、レベル`<=` *N*を持つ各見出しの前に縦線を挿入します。

`:vlines`　tに設定するとき、列グループに縦線を強制します。

`:maxlevel`
数値を設定すると、そのレベル以下のエントリをキャプチャしません。

`:skip-empty-rows`
tに設定すると、カラムビューの空でない指定子が`ITEM`のみである行をスキップします。

次のコマンドはダイナミックブロックを挿入、更新します。

`C-c C-x i`　`org-insert-columns-dblock`
カラムビューをキャプチャするダイナミックブロックを挿入します。そのビューのスコープや ID の入力を促されます。

`C-c C-c` or *`C-c C-x C-u`*　`org-dblock-update`
ポイント下のダイナミックブロックを更新します。カーソルはダイナミックブロックの`#+BEGIN`行にある必要があります。

`C-u C-c C-x C-u`　`org-update-all-dblocks`
全てのダイナミックブロックを更新します (“char 65.6 節「Dynamic blocks」p.202 を参照)。複数のクロックテーブルブロックや列キャプチャブロック、その他のダイナミックブロックがバッファにあるとき便利です。

カラムビューテーブルに計算式を追加し、テーブルの前にプロットする命令を追加できます。これらはブロックの更新があっても生き残ります。テーブルの後に`#+TBLFM:`があるならば、テーブルは実際に更新の後に自動的に実際に再計算されます。

テーブルの中でプロパティ値を処理したりキャプチャする別の方法は Eric Schulte の‘`org-collector.el`’によりできます。それは寄付されたパッケージ[2] です。あるスコープにあるエントリからプロパティを集めるための一般的な API やテーブルやダイナミックブロックの中に挿入する前にそれらの値を処理する任意の Lisp 式を提供します。

[2] 寄付されたパッケージは Emacs の一部ではありませんが Org のメインの配布物と共に配布されます (`http://orgmode.org`を訪ずれて下さい)。

7.6 プロパティAPI

プロパティにアクセス、変更するための完全な API があります。この API はプロパティと共に動作するために、また、それらを元とした機能を実装するために Emacs Lisp プログラムから使われます。詳細な情報は“char 65.9 節「Using the property API」p.206 を参照して下さい。

8 日付と時刻

プロジェクトのプランニングを補助するため、TODO アイテムは日付または時刻でラベリングすることができます。このような形でフォーマットされた日付および時刻の情報を含む文字列は Org-mode ではタイムスタンプと呼ばれています。一般的な用法では、タイムスタンプは何かを作成したときや最後に変更したときの記録を示しますので、若干紛らわしいかもしれませんが、Org-mode ではタイムスタンプという用語をより広い意味で用います。

8.1 タイムスタンプ、デッドラインおよびスケジューリング

タイムスタンプは、例えば ‘<2003-09-16 Tue>’、‘<2003-09-16 Tue 09:39>’、‘<2003-09-16 Tue 12:00-12:30>’といった独自形式での日付（場合によっては時刻および時間間隔を含む）の指定方法です[1]。タイムスタンプは Org ツリー構造の見出し、本文のいずれにも挿入できます。タイムスタンプを指定することにより、指定された日付のアジェンダ (10.3.1 節「Weekly/daily agenda」p.99 を参照) にそのエントリーが表示されます。その際には以下の区別が行われます。

プレーンなタイムスタンプ、イベント、アポイント
: 項目に対して単一の日付または時刻を割り当てるシンプルなタイムスタンプです。紙の予定表に予定あるいはイベントを記入するのとほぼ同じ感覚です。タイムラインおよびアジェンダを表示すると、プレーンなタイムスタンプが指定されたエントリーの見出しはまさにその指定された日付に表示されます。

  ```
  * ピーターと映画を見に行く<2006-11-01 Wed 19:15>
  * 気候変動についてのディスカッション<2006-11-02 Thu 20:00-22:00>
  ```

リピート間隔を指定したタイムスタンプ
: タイムスタンプにはリピート間隔を含めることができます。すなわち単一の日時だけでなく、N 日 (d)、N 週間 (w)、Nヶ月 (m) あるいは N 年 (y) といった一定の間隔で繰り返すようなケースに対応しています。例えば、毎週水曜日のアジェンダに表示する場合は以下のようになります。

  ```
  * 学校までサムを迎えに行く<2007-05-16 Wed 12:30 +1w>
  ```

ダイアリー形式の S 式項目
: より複雑な日付の指定方法として、Org-mode では Emacs の calendar または diary パッケージで実装されている S 式のダイアリー項目を使用することができます。例えば以下のような形式です。

  ```
  * 毎月第 2 木曜日のオタクの集まり
    <%%(diary-float t 4 2)>
  ```

日付または時刻の間隔
: 2 つのタイムスタンプを ‘--’でつなげることにより、時間間隔を表現できます。時間間隔の指定されたヘッドラインは、間隔の始めと終わりの日、およびその間の現在表示されている日付の項目に表示されます。以下のような形です。

  ```
  ** アムステルダムでのミーティング
     <2004-08-23 Mon>--<2004-08-26 Thu>
  ```

[1] この表記は標準的な ISO8601 の日付、時刻フォーマットをもとに考案されています。代替フォーマットの使用については、8.2.2 節「Custom time format」p.73 を参照してください。

アクティブでないタイムスタンプ
: プレーンなタイムスタンプと同様ですが、<>ではなく [] で囲むことによりアクティブでないタイムスタンプとなります。このようなタイムスタンプが指定されたエントリーは、アジェンダに表示されません。

```
* ジリアンが 5 度目の遅刻　[2006-11-01 Wed]
```

8.2 Creating timestamps

Org-mode がタイムスタンプを認識するためには、特定のフォーマットを用いる必要があります。以下のコマンドのいずれを用いても正しいフォーマットでタイムスタンプを生成することができます。

C-c . `org-time-stamp`
: 日付を入力して、それに対応するタイムスタンプを挿入します。既にバッファ内に存在するタイムスタンプにカーソルが置かれている場合は、このコマンドは新たなタイムスタンプを挿入する代わりに、既にあるタイムスタンプを変更します。このコマンドを 2 回連続で使用すると、時間間隔を指定することができます。

C-c ! `org-time-stamp-inactive`
: コマンド *C-c .*と同様ですが、アクティブでない（アジェンダのエントリーに反映されない）タイムスタンプを生成します。

C-u C-c .
C-u C-c !
: *C-c .*および *C-c !*と同様ですが、日付と時刻を含む代替フォーマットを使用します。標準では、時刻は 5 分間隔で丸められます。`org-time-stamp-rounding-minutes`のオプションを参照して下さい。

C-c < `org-date-from-calendar`
: カレンダーのカーソルに対応したタイムスタンプを挿入します。

C-c > `org-goto-calendar`
: 現在時刻の Emac カレンダーにアクセスします。現在の行に既にタイムスタンプが存在する場合は、それに対応する日付にアクセスします。

C-c C-o `org-open-at-point`
: タイムスタンプおよび時間間隔で指定された日付でアジェンダにアクセスします (10.3.1 節「Weekly/daily agenda」p.99 を参照)。

S-left `org-timestamp-down-day`
S-right `org-timestamp-up-day`
: カーソル一の日付を 1 日変更します。このキーバインドはシフト選択およびそれに関連するモードと衝突します (15.10.2 節「Conflicts」p.192 を参照)。

S-up `org-timestamp-up`
S-down `org-timestamp-down-down`
: カーソルのあるタイムスタンプの項目を変更します。カーソルが年、月、日、時間あるいは分の上に置かれている場合に使用できます。例えば、タイムスタンプが‘15:30-16:30’のように時間間隔を含む場合、左の時刻を変更すると自動的に右の時刻も変更され、間隔は一定の長さに保たれます。間隔の長さを変更するには、右の時刻を変更して下さい。ただし、カーソルがタイムスタンプではなく見出し上にある時には、同じキー操作により項目の優先度が変更されますので気を

つけて下さい (5.4 節「Priorities」p.53 を参照)。このキーバインドはシフト選択および関連するモードとも衝突します (15.10.2 節「Conflicts」p.192 を参照)。

C-c C-y org-evaluate-time-range
開始日時と終了日時の差を計算することにより、時間間隔を計算します。前置引数を指定することにより、計算結果をタイムスタンプの後に挿入できます（テーブルの中では隣の列に挿入されます)。

8.2.1 The date/time prompt

Org-mode が日付または時刻をプロンプトに表示するとき、標準では標準フォーマットによる形式が表示されるため、そのフォーマットで入力することが必須だと勘違いしそうになります。ところが、実際には日付または時刻の情報を含む任意の文字列を入力することができ、Org-mode はかなり利口に入力された時間情報を解釈します。例えば、*C-y*により電子メールの文面からコピーした文字列（複数行でも可）を挿入することができます。Org-mode は文面の中の時間情報を見つけ出し、そこで指定されていない情報はデフォルトの日付時刻を用います。デフォルトは通常は現在の日付および時刻ですが、既にあるタイムスタンプを変更する場合や、時間間隔の 2 つ目の項目を入力する場合には、バッファ内のタイムスタンプから情報が取得されます。情報を解釈する際に、Org-mode は多くの場合では入力したい時間が未来の時間であると推測します。例えば年月の情報を省略して、今日より前の時刻を指定しようとすると、Org-mode は未来の時刻を意図しているものと推測します[2]。日付が自動的に未来にシフトされた場合、プロンプトには ‘(=>F)’が表示されます。

例えば、今日が **2006 年 6 月 13 日**であるとしたとき、以下の左ような入力は右のように解釈されます。Org-mode により推定された部分を**太字**で示します。

```
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            ⇒ 直近の金曜日 (基準日かそれより後)
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             ⇒ 現在の年（2006 年）の ISO 週番号
2012 w4 fri    ⇒ 2012 年の ISO4 週目の火曜日の日付
2012-w04-5     ⇒ 上と同様
```

さらに、相対的な日付を入力するための方法として、入力の最初にプラスまたはマイナス記号、数値および文字 ([dwmy]) により日、週、月あるいは年の変化を指定する方法があります。単一のプラス/マイナスを入力すると、常に今日に対する相対的な日付が指定されます。2 つのプラスまたはマイナスが入力されると、標準の日付に対する相対値となります。一文字の代わりに時間に関する省略文字列を指定すると、N 番目の該当する日が指定されます。以下に例を示します。

```
+0             ⇒ 今日
```

[2] org-read-date-prefer-futureの変数を参照。この変数 timeに該当する変数を変更することにより、現在時刻より前の時刻を明日にシフトすることも可能です。

```
.           ⇒ 今日
+4d         ⇒ 今日から 4 日後
+4          ⇒ 上と同様
+2w         ⇒ 今日から 2 週間後
++5         ⇒ 標準日時から 5 日後
+2tue       ⇒ 今日から数えて 2 回目の火曜日
```

この機能では、英語の月および曜日の省略記法に対応しています。省略しない記法や他の言語の記法を使用したい場合には、変数 parse-time-monthsおよび parse-time-weekdays を変更して下さい。

時間間隔は、開始時刻と終了時刻を入力するか、開始時刻とその継続時間 (HH:MM の形式) を入力することにより指定できます。前者の場合は分離記号として '-' あるいは '-{}-' を使用し、後者の場合は分離記号として'+' を使用して下さい。例えば以下の通りです。

```
11am-1:15pm     ⇒ 11:00-13:15
11am--1:15pm    ⇒ 上と同様
11am+2:15       ⇒ 上と同様
```

ミニバッファのプロンプトと並行して、カレンダーがポップアップします[3]。カレンダー内の日付をクリックするか、RETを入力することにより日付のプロンプトを抜けると、カレンダーで選択した日付とプロンプトで入力された情報が組み合わされます。カレンダーはミニバッファから自由に操作することがきます。

```
RET             カレンダーのカーソル地点の日付を選択する。
mouse-1         クリックにより日付を選択する。
S-right/left    1 日分進める/戻る
S-down/up       1 週間分進める/戻る
M-S-right/left  1ヶ月分進める/戻る
> / <           カレンダーを 1ヶ月前/後ろにスクロールする
M-v / C-v       カレンダーを 3ヶ月前/後ろにスクロールする
```

文章の説明では、日付時刻プロンプトの動作は複雑に思えるかもしれませんが、徐々に慣れてくると、これ以外の方法で日付および時刻を入力することのほうが面倒に感じることでしょう。動作の仕組みに対する理解を助けるため、入力された情報に対するその時点での解釈がミニバッファに表示されます。[4]。

8.2.2 Custom time format

日付や時間を表現するため、Org-mode は ISO8601 で定義されているような標準的な ISO の記法を使用しています。もしこの記法に不慣れで、別の日付や時間の記法のほうが好みである場合は、変数 org-display-custom-timesおよび org-time-stamp-custom-formatsをカスタマイズすることができます。

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

カスタムフォーマットの日付および時間の表示をトグルします。

Org-mode は文字列のスキャンニングのためにデフォルトのフォーマットを必要とするため、カスタムフォーマットの日付および時刻は標準フォーマットを置き換えません。その代わりに、テキストのプロパティを用いて標準フォーマットに上書きされます。これが原因で以下のような動作が生じます。

---

[3] カレンダーの表示が不要の場合、変数 org-popup-calendar-for-date-promptを変更して下さい。

[4] もしミニバッファの表示が目障りな場合は、org-read-date-display-liveで表示しないよう設定することができます。

- タイムスタンプの上にカーソルを置くことはできなくなり、その前後にしかカーソルが移動しなくなります。
- *S-up/down*のキー操作は、タイムスタンプの各要素を変更するために使用できなくなります。カーソルがスタンプの前にある場合、*S-up/down* によりスタンプを 1 日だけ変更します。これは *S-left/right*と同様です。スタンプの後ろにある場合、時間が 1ヶ月ずつ変更されます。
- タイムスタンプが時間間隔や繰り返し時刻を含む場合、これらは上書きされずに元の形式のままバッファに表示されます。
- タイムスタンプを一文字ずつ消去した場合、(隠れた)ISO 標準フォーマット文字列の全てを削除した場合のみカスタムフォーマットのタイムスタンプが消去されます。
- もし、カスタムフォーマットのタイムスタンプが標準フォーマットより長く、テーブル内で用いられている場合、テーブルの整形が崩れます。標準フォーマットより短い場合には、期待通りに動作します。

8.3 Deadlines and scheduling

プランニングを補助するため、タイムスタンプの前に所定のキーワードを置くことができます。

`DEADLINE`

意味:タスク (多くの場合は TODO アイテムですが、それに限りません) はタイムスタンプで示された日のうちに終了するものと見なされます。

デッドラインが付けられた場合は、そのタスクはアジェンダの中に記載されます。それに加えて、今日のアジェンダはデッドラインが近づいたり、それを超過したりした場合に警告を発します。警告は期限の `org-deadline-warning-days`だけ前から表示され、エントリが DONE とされるまでは消えません。以下に例を示します。

```
*** TODO ガイド誌の地球についての記事を書く。
    担当編集者は [[bbdb:Ford Prefect]]
    DEADLINE: <2004-02-29 Sun>
```

以下の構文を用いることにより、個別のデッドラインについて異なる警告のリードタイムを指定することができます。以下は 5 日間の警告期間を指定する場合の例です `DEADLINE: <2004-02-29 Sun -5d>`。

`SCHEDULED`

意味:指定された日に、そのタスクに取りかかる予定であることを示します。

見出しは指定された日付の下に記載されます[5]。それに加えて、スケジューリングされた日付を超過した場合には今日のリストにエントリが DONE となるまでリマインダが表示され続けます。すなわち、タスクは完了するまで自動的に後回しにされます。

```
*** TODO トリリアンに大晦日のデートについて電話する。
    SCHEDULED: <2004-12-25 Sat>
```

重要: Org-mode で項目をスケジューリングすることは、ミーティングをスケジューリングすることと同様であるという理解は正しくありません。ミーティングをセッ

[5] 項目が DONE とマークされた場合でも、指定日の項目に表示され続けます。この設定が好みでなければ、変数 `org-agenda-skip-scheduled-if-done`を指定して下さい

トするのは単なるアポイントですが、このような場合はエントリーにはプレーンなタイムスタンプを使用し、日付が来れば項目が表示されるように設定するべきです。これはユーザーがしばしば誤解する点です。Org-mode ではスケジューリングは何らかのアクションアイテムに取りかかる際に、日付を設定することを意味します。

スケジューリングやデッドラインの項目には、繰り返しを含むタイムスタンプを使用することが可能です。Org-mode は、タイムスタンプが繰り返し日付の直近の日付を表すものと推測して事前あるいは事後の警告を発します。しかし、スケジューリングやデッドラインにおいては<%%(diary-float t 42)>のような日記の S 式項目は限定的にしか使用できません。Org-mode はこれら S 式項目の内部構造について十分理解していないため、事前および事後の警告を発することはできません。ただし、S 式項目と一致したそれぞれの日付に項目を表示することは行われます。

8.3.1 デッドラインおよびスケジュールの挿入

以下のコマンドにより、項目にデッドラインまたはスケジュールを瞬時に挿入[6] することができます。

C-c C-d org-deadline

タイムスタンプと 'DEADLINE'キーワードを挿入します。挿入は見出しの直下の行に対して行われます。前置引数を伴って呼ばれた場合は、エントリーから既に存在するデッドラインが消去されます。変数 org-log-redeadline[7] に対応して、既に存在するデッドラインを変更する際にノートをとることができます。

C-c C-s org-schedule

タイムスタンプと 'SCHEDULED'キーワードを挿入します。挿入は見出しの直下の行に対して行われます。CLOSED のタイムスタンプは全て消去されます。前置引数を伴って呼ばれた場合は、エントリーからスケジューリングの日付が消去されます。変数 org-log-reschedule[8] に対応して、既に存在するスケジューリングを変更する際にノートをとることができます。

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

現在の項目をアジェンダのアクションのためにマークします。このように項目をマークした後で、アジェンダまたはカレンダーを開いて適切な日を探すことができます。選択した日付の上にカーソルを置いて *k s*あるいは *k d*を入力することにより、マークされた項目にスケジュールを設定できます。

C-c / d org-check-deadlines

全てのデッドラインのうち、既に過ぎているものと org-deadline-warning-days 以内に期限となるものを抽出したツリーを作成します。前置引数 *C-u*により、ファイル内の全てのデッドラインを表示します。前置引数で数値を指定すると、指定した分だけ先のデッドラインを表示します。例えば、*C-1 C-c / d*とすると明日期限となる全てのデッドラインを表示します。

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

指定された日より前のデッドラインを抽出したツリーを作成します。

---

[6] 'SCHEDULED'あるいは 'DEADLINE'の付いた日付が見出しのすぐ下の行に挿入されます。見出しとこの行の間には文字を記入してはいけません。

[7] 対応する#+STARTUPキーワード logredeadline、lognoteredeadline、および nologredeadline

[8] 対応する#+STARTUPキーワード logredeadline、lognoteredeadline、および nologredeadline

C-c / a org-check-after-date
指定された日より後のデッドラインを抽出したツリーを作成します。

8.3.2 Repeated tasks

タスクの中には、何度も繰り返し行うものがあります。Org-mode では、そのようなタスクの管理を助けるため、通常あるいは DEADLINE、SCHEDULED のタイムスタンプに対してリピーターと呼ばれる機能を提供しています。以下の例を参照して下さい。

```
** TODO 家賃の支払い
   DEADLINE: <2005-10-01 Sat +1m>
```

この中で+1mがリピーターと呼ばれるもので、そのタスクが<2005-10-01>のデッドラインを持つと同時に、その日から一週間ごとに繰り返すことを意味します。デッドラインエントリーに対してリピーターと警告期間の両方を指定する必要がある場合には、DEADLINE: <2005-10-01 Sat +1m -3d>のようにリピータを先に書き、警告期間を後に書きます。

デッドラインおよびスケジューリングされた項目は期限を過ぎた場合にはアジェンダ上にエントリーが作成されるため、項目が終了した場合にはそのようなエントリは実施済みとマークできることが必要です。DEADLINE あるいは SCHEDULED の TODO エントリーを DONE とマークした時には、アジェンダ上でのエントリーは作成されなくなります。一方で、問題となるのは繰り返し項目の次の日時も同時にアクティブでなくなってしまうことです。Org-mode では、このような状況に対して以下のように対処します。繰り返しのエントリーを DONE に変更しようとした場合 (*C-c C-t*などにより)、繰り返しタイムスタンプの基準時刻が一つ分シフトされ、すぐにエントリーの状態が TODO に戻されます[9]。上の例では、DONE の状態にすることにより日付が以下のように変更されます。

```
** TODO 家賃の支払い
   DEADLINE: <2005-11-01 Tue +1m>
```

デッドラインの下にタイムスタンプが追加され[10]、これにより前の時刻のデッドラインについて実際に行動したことが記録されます。

日付がシフトされた結果として、このエントリーはアジェンダ上の過去の日付からは見えなくなりますが、将来の日付はアジェンダ上で確認することができます。

‘+1m’の繰り返し指定により、日付は常に 1ヶ月きっちりシフトされます。そのため、例えば家賃を 3ヶ月支払っていない場合には、一度このエントリーを DONE にしたとしても、相変わらず期限を過ぎたデッドラインと判断されます。タスクの性質によって、この方法が常に適切な処理方法とは限りません。例えば、父親と連絡をとるのを 3 週間忘れてしまった場合、それを埋め合わせるために一日に 3 回電話をすることはないでしょう。さらに、バッテリーの充電のように、最後に行ってから一定時間経過後に常に繰り返す必要がある場合もあります。このようなタスクについては Org-mode は専用の反復演算子++および.+を用意しています。例えば以下の通りです。

```
** TODO 父に電話
   DEADLINE: <2008-02-10 Sun ++1w>
   この項目を DONE とすると、日付が一週間シフトされ、同時にタスクが
   行われるまでは将来の全ての日曜日に対して項目がシフトされます。
```

---

[9] 実際には、変更される状態は REPEAT_TO_STATEプロパティ或いは変数 org-todo-repeat-to-stateによって決められます。これらが指定されていない場合は、デフォルトとして TODO 状態に戻ります。

[10] この部分の動作は、オプション org-log-repeat、あるいは#+STARTUPオプションの logrepeat、lognoterepeat、nologrepeatにより変更することができます。lognoterepeat を指定した場合には、メモを入力するように促されます。

土曜日に電話をしたとしても、次の日曜日にシフトされます。

```
** TODO 火災報知器の電池をチェックする
   DEADLINE: <2005-11-01 Tue .+1m>
```

この項目を DONE とすると、確認した日のちょうど 1ヶ月後にシフトされます。

同じタスクに対して、スケジューリングとデッドラインの情報を両方付けることができます。この場合には、両方の繰り返し間隔は同じになりますので、注意して下さい。

繰り返し演算子を使わない代替的な方法の一つとして、タスクサブツリーのコピーをいくつか作成し、それぞれのコピーに対してシフトされた時刻を指定する方法があります。そのために *C-c C-x c*コマンドがあります。この機能は 2.5 節「Structure editing」p.9 で解説されています。

8.4 Clocking work time

Org-mode では、プロジェクトの中で特定のタスクを実行するのにかかった時間を計測することができます。ある項目について取りかかる時に、計測を開始します。そのタスクを中断するときやタスクが終了した時に計測が終了し、対応する時間間隔が記録されます。同時に、あるプロジェクトの全てのサブツリーでかかった時間の合計が計算されます。さらに、最近に時間が計測されたタスクが記憶されているため、その時点で取りかかっている複数のタスク間を素早く移動することができます。

Emacs セッションでの経過時間の履歴を保存するためには、以下のコマンドを使います。

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

Emacs の再開後に新しいタスクの計測を始めると、不完全な時計[11] が表示され (8.4.3 節「Resolving idle time」p.81 を参照)、それについて何をするかを入力するように促されます。

8.4.1 Clocking commands

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

現在の項目に対して、時間の計測を開始します（クロックイン）。これにより CLOCK キーワードとともにタイムスタンプが挿入されます。最初の計測でない場合には、複数の CLOCK 行が表示され、:LOGBOOK: という引き出しに格納されます (変数 org-clock-into-drawerについても参照のこと)。前置引数 *C-u*と共に呼ばれた場合は、最近時間が計測されたタスクのリストからタスクを選択します。2 個の前置引数 *C-u C-u*が入力された場合は、現在位置のタスクの計測を開始し、そのタスクをデフォルトに指定します。デフォルトのタスクは、時間計測の選択をする場合に常にリストの中に表示され、文字 *d*が付けられます。
時間を計測している間は、現在の計測時間とタスクの名前がモード行に表示されます。表示される計測時間はこのタスクとその子タスクについての全ての時間です。タスクが工数の見積もり (8.5 節「Effort estimates」p.82 を参照) を含む場合には、モード行は見積もりに対する現在の経過時間を表示します[12]。タスクが繰り返しを含む場合には、そのタスクの最近のリセットからの経過時間[13] のみが計測されま

[11] Emacs の外でタスクに取りかかっていたという想定で時計を再開する場合は、(setq org-clock-persist t) を使用して下さい。

[12] 「その場で」工数の見積もりを追加するには、その機能を持つ関数を変数 org-clock-in-prepare-hookにフックして下さい

[13] プロパティLAST_REPEATにより記録

す。表示される時間について変更したい場合には、プロパティCLOCK_MODELINE_TOTALを変更します。この中には現在計測中の時刻のみ表示する current、今日計測された全ての時間を表示する today(変数 org-extend-today-untilについても参照)、全ての時間を表示する all、デフォルトの設定である autoなどがあります[14]。
モード行を *mouse-1*でクリックすることにより、メニューと時間計測のオプションがポップアップします。

C-c C-x C-o org-clock-out
時間の計測を終了します（クロックアウト)。これにより、時間計測が開始されたのと同じ場所にもう一つのタイムスタンプが挿入されます。同時に、計測された時間間隔が開始時刻と終了時刻の後に '=> HH:MM'の形式で挿入されます。クロックアウトのタイムスタンプ作成時にノートを追加するには、変数 org-log-note-clock-outを参照して下さい[15]。

C-c C-x C-e org-clock-modify-effort-estimate
現在時間を計測しているタスクについて、工数見積もりをアップデートします。

C-c C-c or *C-c C-y* org-evaluate-time-range
タイムスタンプの一つを変更した後で、時間間隔を更新します。これはタイムスタンプを手動で変更した場合にのみ必要です。*S-cursor*キーにより変更した場合には、自動的に再計算されます。

C-c C-t org-todo
TODO の項目を DONE に変更することにより、その項目で時間が計測されている場合には自動的に停止します。

C-c C-x C-x org-clock-cancel
現在の時間計測をキャンセルします。これは間違って時間を計り始めてしまった場合や、結果的に意図したタスク以外を行ってしまった場合に便利です。

C-c C-x C-j org-clock-goto
現在クロックイン中のタスクの見出しにジャンプします。前置引数 *C-u*により、最近時間が計測されたタスクから目的のタスクを選択します。

C-c C-x C-d org-clock-display
現在のバッファの各サブツリーからの時間のサマリーを作成します。これによりそれぞれの見出しの後ろに時間が上書きされ、その見出しの中で下位の見出しも含めて記録された時間の合計が表示されます。表示の切り替えにより、ツリーの各項目を確認できますが、バッファを変更した場合や *C-c C-c*を入力した場合は時間の上書きは消えてしまいます (変数 org-remove-highlights-with-change) を参照)。

タイムライン (10.3.4 節「Timeline」p.104 を参照) およびアジェンダ (10.3.1 節「Weekly/daily agenda」p.99 を参照) の中でキー *l*を入力することにより、その日のうちに完了したタスクがどれかを表示することができます。

[14] 変数 org-clock-modeline-totalについても参照
[15] これに対応するバッファ内の設定は#+STARTUP: lognoteclock-outです。

8.4.2 The clock table

Org-mode は、時間計測の情報をもとにかなり詳細なレポートを作成することができます。このようなレポートはクロックテーブルと呼ばれており、その名の通り Org-mode のテーブルまたはその組み合わせにより作成されます。

C-c C-x C-r `org-clock-report`
: 現在のファイルに Org-mode テーブル形式の計測時間レポートを含む動的ブロック (“char 65.6 節「Dynamic blocks」p.202 を参照) を挿入します。前置引数とともに呼ばれた場合は、現在の文書の最初のクロックテーブルに移動し、それを更新します。

C-c C-c or *C-c C-x C-u* `org-dblock-update`
: 現在位置の動的ブロックを更新します。カーソル位置は動的ブロックの`#+BEGIN`行の中に位置している必要があります。

C-u C-c C-x C-u
: 全ての動的ブロックを更新します (“char 65.6 節「Dynamic blocks」p.202 を参照)。この機能はバッファ内で複数のクロックテーブルが存在する場合に有用です。

S-left
S-right `org-clocktable-try-shift`
: 現在の`:block`の時間間隔を変更し、テーブルを更新します。このコマンドを使用するには、カーソルが`#+BEGIN: clocktable`行にある必要があります。例えば`:block`が`today`の場合、このコマンドにより`today-1`にシフトされます。

以下に、*C-c C-x C-r*コマンドによりバッファに挿入されるクロックテーブルのフレームの例を示します。

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

‘BEGIN’行には、レポートの対象範囲、構造およびフォーマットを定めるオプションを指定します。これらのデフォルト値は変数`org-clocktable-defaults`により変更することができます。

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

`:maxlevel` テーブルに表示される最大の深さレベル。
これより深いレベルの時間は上位レベルに積算されて表示される。

`:scope` 対象とするスコープ。以下のうちいずれかを指定する。

`nil`	現在のバッファ、或いはナローされた領域
`file`	現在のバッファ全体
`subtree`	クロックテーブルのあるサブツリー内
`treeN`	周囲のレベル N のツリー、例えば `tree3`
`tree`	周囲のレベル 1 のツリー
`agenda`	アジェンダファイル群の全体
`("file"..)`	指定されたファイルをスキャンする
`file-with-archives`	現在のファイルとそのアーカイブ
`agenda-with-archives`	アーカイブを含む全てのアジェンダファイル

`:block` 対象とする時間範囲、この範囲は絶対時間または現在からの相対時間で表記され、以下のいずれかのフォーマットに従う。

2007-12-31　2007 年の大晦日
2007-12　2007 年 12 月
2007-W50　2007 年の ISO 週で 50 週目
2007-Q2　2007 年の第二四半期
2007　2007 年
today, yesterday, today-*N*　相対的な日指定
thisweek, lastweek, thisweek-*N*　相対的な週指定
thismonth, lastmonth, thismonth-*N*　相対的な月指定
thisyear, lastyear, thisyear-*N*　相対的な年指定
*S-left/right*キーにより間隔をシフトすることができる。

:tstart　対象となる時間の始まりを示す文字列
:tend　対象となる時間の終わりを示す文字列
:step　テーブルをまとめる間隔。weekまたは dayを指定。
この機能を使うには、:block、:tstart、:tendのいずれかを指定する必要がある。
:stepskip0　時間間隔がゼロの項目を表示しない。
:fileskip0　時間間隔がゼロのファイルについて、テーブルに表示しない。
:tags　特定のタグがついたエントリのみ収集の対象とする。

さらに、テーブルのフォーマットを指定するためのオプションがあります。これらのオプションは関数 org-clocktable-write-defaultにより解釈されますが、パラメータ:formatterにより解釈のためのユーザ独自の関数を指定することができます。

:emphasize　値が tの場合、レベル 1 およびレベル 2 の項目を強調表示します。
:lang　項目名のセル (例えば"Task") で用いる言語[16]。
:link　テーブルの見出しの項目と元のファイルでの位置をリンクする。
:narrow　Org-mode のテーブルの見出し列の幅を上限を決める整数。
'50!'のように指定すると、エクスポート時にも見出しが短縮表示される。
:indent　各見出しフィールドをそのレベルに合わせてインデントする。
:tcolumns　時間を表示するために使われる列の数。この値が:maxlevelより小さい場合、
それより下位のレベルは一つの列に合わせて表示される。
:level　レベル番号を示す列を含めるかどうか指定する。
:compact　コンパクトに表示する。
:level nil :indent t :narrow 40! :tcolumns 1
の短縮表現で、明示的に:narrowで指定されなければ、全ての変数は上書きされる。
:timestamp　エントリのタイムスタンプが存在する場合には、それを表示する。SCHEDULED、
DEADLINE、TIMESTAMP、TIMESTAMP_IA の順に探索される。
:formula　追加的な#+TBLFMの内容。通常の形式に追加されて評価される。
特殊なケースとして、':formula %'を追加すると経過時間の割合行が追
加される。

[16] 言語に関する項目は、変数 org-clock-clocktable-language-setupにより設定することができます。

ここで形式をしていしない場合は、クロックテーブルの下に存在する形
式が
アップデートされずに評価される。

`:formatter` 時刻データをフォーマットし、バッファに表示するための関数。

現在のレベル 1 のツリーについて、当日分の時間サマリーを得る場合は以下のように指定します。

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

明示的に時間間隔を指定する場合には、以下のように記述します[17]。

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

現在のサブツリーでの経過時間のまとめを%表示するには、以下のように記述します。

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

ここ 1 週間で計測された時間をコンパクトな幅で表示するには、以下のようにします。

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

8.4.3 Resolving idle time

ある項目について作業を開始したあとで、例えば電話を取る場合などで一時的にコンピュータの前を離れると、その時間について現在の経過時間から差し引いたり、他の項目に加えたりして「解決」する必要が生じます。

変数 `org-clock-idle-time`を適当な整数値 (例えば 10 や 15) に設定することにより、設定時間を超える休止のあとで戻ってきた場合に Emacs はアラートを出し、その休止時間をどのように処理するか尋ねます[18]。休止から戻った時点で幾つか質問が表示され、実際にどの程度休止時間があったか（その時点までの計測時間が随時表示されます）を入力すると同時に、休止の扱いについて以下のような選択が可能です。

k 休止時間として計測された時間の一部または全てをタスクの計測時間として保持する場合には、*k*を押します。Org-mode は何分間保持するか尋ねます。`RET`キーを押すことにより全ての時間が保持され、タスクの計測時間は変更されません。数値を指定すると、指定した分数だけ時間が保持されます。

K シフトキーと共に *K*を押した場合、入力された分数だけ時間を保持すると同時にただちに現在のタスクの計測を中止します。全ての休止時間を保持する場合、これは単にタスクの計測を中止したのと同じことになります。

s 休止時間を保持しない場合には、*s*を押すことにより計測時間から全ての休止時間が差し引かれ、戻ってきた時点から再開されます。

---

[17] 全てのパラメータは単一行で指定する必要があるので注意して下さい。この例ではマニュアルの文字幅の制約のために改行が入っています。

[18] Mac OS X のコンピュータでは、Emacs の休止時間だけでなくユーザーが実際に休止した時間を計測します。X11 では、Org-mode の git ディストリビューションから入手できるユーティリティプログラム ‘`x11idle.c`’ をインストールすることにより、同様に全体の休止時間を計測することができます。その他のシステムでは、休止時間は Emacs が休止していた時間のみを表します。

S　休止時間を保持せず、休止開始時の時刻で時間の計測を止める場合には、シフトキーと共に *S*を押して下さい。シフトキーを使うと、いずれのオプションでも時間の計測が中止されるということを覚えておいて下さい。

C　時間計測そのものをキャンセルする場合は、*C*を押して下さい。キャンセルしない場合でも、時間が引かれた結果の残り時間が 1 分未満である場合には、中身の無いエントリでログが見づらくなるのを防ぐため、時間計測はやはりキャンセルされますので注意して下さい。

空き時間について、現在の計測時間から差し引いたあとで別の計測項目に追加したい場合にはどうすれば良いでしょうか。その場合は、差し引いたあとに単純に次のタスクの計測を開始して下さい。Org-mode は差し引かれた時間があることを記憶していて、次の時間計測を始める際にその時間を足し込むかを尋ねます。

次のようなケースでも、時間解決機能が魔法のような働きをします。あなたがタスクの時間計測をしながらご機嫌に作業をすすめていると、突然飼い猫がネズミを追いかけて、それを見たハムスターが驚いて UPS の電源装置に衝突してしまったとしましょう!あなたは全てのバッファを失うことになりますが、オートセーブ機能のおかげで Org-mode で行った最近の変更は保持され、途中であった時間計測の時間も保持されています。

Emacs を再開してタスクの計測を開始すると、Org-mode は最後のセッションで計測が終了されていない半端の時間計測があることに気がつきます。そのようなタスクについては、計測の開始時刻を不明な時刻の始点として、その間の時間をどのように解決するかについて尋ねます。その際の考え方や挙動は空き時間の処理方法と全く同じで、単に空き時間ではなくリカバリの際に発生しているだけなのです。

Org-mode のアジェンダが半端時間を絶えずチェックしているファイルのリストは、*M-x org-resolve-clocks*により確認することができます。

8.5 Effort estimates

詳細な作業計画を立てて仕事を行いたい場合や、仕事の工数の見積もりを作成する必要がある場合には、エントリに工数見積もりを割り当てたいと思うかもしれません。また、同時に時間の計測を行う場合には、あとで見積もった時間数と実際にかかった時間を比較したいと思うかもしれません。それは見積もりの精度を上げる良い方法でもあります。工数の見積もりは専用のプロパティである ‘Effort’に保存されます[19]。エントリに工数を追加するには、以下のようなコマンドを用います。

C-c C-x e　`org-set-effort`
現在のエントリについて工数の見積もりを行います。前置引数に数値を指定することにより、N 番目の数値に指定します (下記の例を参照)。このコマンドはアジェンダからも *e*キーを押すことによりアクセスできます。

C-c C-x C-e　`org-clock-modify-effort-estimate`
現在時間が計測されている項目の工数見積もりを変更します。

明らかなように、工数見積もりを行う最善の方法はカラムビュー (7.5 節「Column view」p.64 を参照) を用いることです。個別の項目についての工数見積もりから始めて、COLUMNS フォーマットによりこれらの値と実際の計測時間 (時間の計測を行いたい場合) を同時に表示します。例えば、あるバッファについて以下のように指定できます。

[19] 使用されるプロパティは、変数 `org-effort-property`で変更することができます

```
#+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
```

さらに良い方法としては、変数 org-global-propertiesあるいは org-columns-default-formatをカスタマイズすることにより、これらの数値をグローバルに指定できます。特にこの指定をアジェンダで使用したい場合には、グローバルな指定を行うことが推奨されます。

個別の項目について見積もりを割り当てるには、カラムモードに移行し、*S-right*および*S-left*を使うことにより値を変更します。入力された数値はすぐに階層構造で足し合わされます。その隣の列には計測された時間が表示されることになります。

日別あるいは週別のアジェンダでカラムビューに移行すると、工数の列は各日についての工数見積もりを足し合わせて表示され[20]、これを用いてスケジュールの空きを見つけることができます。その日の作業について全体像をつかみたい場合には、オプション org-agenda-columns-add-appointments-to-effort-sumを指定することができます。時間間隔が指定されているアポイントで、その日に発生するものについても負荷見積もりに加算して表示されます。

工数見積もりは、アジェンダ内で/を押すことによりアジェンダの 2 次的なフィルタリングに用いることができます (10.5 節「Agenda commands」p.107 を参照)。このような見積もりを確実に行えば、2、3 回キーを押すことにより空いている時間間隔に合うように項目を絞り込むことができます。

8.6 相対時間タイマーを使ったノート作成

例えば会議やビデオ閲覧時にノートをとる際など、開始時からの経過時間がわかると便利な場合があります。Org-mode はそのような場合に使える相対時間タイマー機能を持っており、時間を含むノートを簡単に作ることができます。

C-c C-x . org-timer
: バッファに相対時間タイマーを挿入します。最初に使う際にはタイマーが開始されます。前置引数と共に呼ばれた場合には、タイマーがリスタートされます。

C-c C-x - org-timer-item
: 現在の相対時刻での記述項目を挿入します。前置引数と共に呼ばれた場合は、タイマーの時刻が 0 にリセットされます。

M-RET org-insert-heading
: タイマーリストが既に開始されている場合は *M-RET*で新しいタイマー項目を追加することもできます。

C-c C-x , タイマーを一時停止します。一時停止されている場合には再開します (org-timer-pause-or-continue)。

C-u C-c C-x ,
: タイマーを停止します。これを実行した後には古いタイマーを再開することはできず、新しいタイマーの作成のみが可能です。このコマンドによりモード行からもタイマーが削除されます。

C-c C-x 0 org-timer-start
: バッファに何も挿入せずにタイマーをリセットします。デフォルトではタイマーは 0 にリセットされますが、前置引数 *C-u*と共に呼ばれた場合は、指定された時

[20] 単準なリストを階層的に足し合わせる際には落とし穴があります (10.8 節「Agenda column view」p.121 を参照)。

間からタイマーが始められます。ユーザーは開始時間を入力するよう促されます。同じ位置に既にタイマー文字列がある場合には、その時間がデフォルトとして指定されます。そのため、このコマンドは休憩時間のあとなどでノート取りを再開する場合などに用いることができます。2 つの前置引数 *C-u C-u*と共に呼ばれた場合は、アクティブなリージョンにある全てのタイマー文字列を一定の時間だけ変化させます。これはタイマーを正しい時刻に開始できなかった場合、タイマー文字列を一度に修正する場合に使用できます。

8.7 カウントダウンタイマ

Org-mode バッファから `org-timer-set-timer`を呼ぶことにより、カウントダウンタイマーが利用できます。アジェンダバッファの場合は`;`、その他は `C-c C-x ;`により実行できます。

`org-timer-set-timer`により、ユーザーに時間間隔を入力するように促し、モード行にカウントダウンタイマーを表示します。`org-timer-default-timer`によりデフォルトのカウントダウン値を設定します。前置引数で数値を指定することで、デフォルトの値が上書きされます。

9 Capture - Refile - Archive

管理システムにおける重要な点の一つとして、新しいアイディアやタスクを素早くキャプチャし、それらを参考資料と連携する能力があることです。Org-mode ではキャプチャと呼ばれるプロセスを用いて行います。また、タスクに関係するファイル (*attachments*) を特別なディレクトリに保存することがも可能です。一度システムへ取り込むと、タスクとプロジェクトを移動させる必要があります。完了したプロジェクトツリーをアーカイブファイルへ移動することで、システムをコンパクトで速く保つことが可能です。

9.1 Capture

Org-mode では、新しいアイテムをキャプチャする方法は John Wiegley による素晴らしい remember パッケージから多くのアイディアを得ています。バージョン 6.36 の Org-mode までは 'remember.el'用の特別な設定を使っていました。'org-remember.el'は従来の設定との逆互換性のため、まだ Org-mode の一部です。org-remember に関するドキュメントは http://orgmode.org/org-remember.pdf にあります。

ここで述べる新しいキャプチャのための設定が好ましく、新しいユーザーはこれを使用するべきです。あなたの org-remember-templates を変換する場合は以下のコマンドを実行します。

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

そして新しい変数を *M-x customize-variable org-capture-templates* でカスタマイズし、結果を確認してから保存してください。これにより、新しい仕組みになれるまではリメンバーとキャプチャの両方を使うことができます。

キャプチャはワークフローにおいて小さい割り込みで素早くノートを保存することができます。キャプチャの基本的なプロセスはリメンバーととても良く似ていますが、Org-mode はそれをテンプレートなどで強化しました。

9.1.1 Setting up capture

以下のカスタマイズはノートを取るデフォルトのファイルと新しい素材をキャプチャするためのグローバルなキー[1]

```
(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

org-caputre コマンドを呼びます。このキーバインドはグローバルで、デフォルトではアクティブになっておらず、インストールする必要があることに注意が必要です。もしテンプレートがあるならば defined 9.1.3 節「Capture templates」p.86 を参照, これらのテンプレートの選択か、デフォルトのテンプレートを使用した新しい Org アウトラインノードが使用されます。

C-c C-c　org-capture-finalize

キャプチャバッファに情報を入力し終わったら、*C-c C-c* はキャプチャプロセスの前に設定されたウィンドウへ戻します。これによってこれ以上気を散らさずに

[1] 自分のキーを選択して下さい。*C-c c* はただの提案です。

作業を再開することができます。前置引数と一緒に呼ばれた場合は、仕上げをしたあとキャプチャした項目へ移動します。

C-c C-w　org-capture-refile
ノートを別の場所に差し替える (9.5 節「Refiling notes」p.92 を参照) ことでキャプチャプロセスを仕上げます。これは通常の差し替えコマンドが実行されることを認識してください。したがってこのコマンドを実行するときのカーソル位置が重要です。もし親や小を持つツリーを挿入する場合、まずカーソルを親へ移動してください。このコマンドに渡された接頭辞引数はすべて org-refileコマンドに渡されます。

C-c C-k　org-capture-kill
キャプチャプロセスをアボートして前の状態へ戻ります。

org-captureをアジェンダから *k c*キーの組合せを用いた特別な方法で呼ぶこともできます。この方法では、選択されたキャプチャテンプレートに挿入されるタイムスタンプは、現在の日付ではなくアジェンダ内のカーソルがある位置の日付がデフォルトになります。

最後にキャプチャが保存された場所を探すには、org-captureをプレフィックスコマンドと一緒に使用します。

C-u C-c c
キャプチャテンプレートが対象としている場所に移動します。テンプレートの選択は通常と同じように行います。

C-u C-u C-c c
バッファ内で最後に保存したキャプチャアイテムの場所に移動します。

9.1.3 Capture templates

テンプレートは異なる種類のキャプチャアイテムや、異なる場所へ使用することができます。最も簡単にそのようなテンプレートを作る方法、カスタマイズインターフェースを通じて行うことです。

C-c c C　org-capture-templates変数のカスタマイズを行います。

テンプレート定義の正式な説明の前に、例を挙げます。一般的な TODO エントリーを作成する場合を考えます。また、これらのエントリーは‘~/org/gtd.org’の中にある見出し‘Tasks’の下に作成され、‘journal.org’内のデートツリーはジャーナルエントリーがキャプチャされます。このような場合の設定は以下のようになります。

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

ここで *C-c c t*を押すと、Org-mode はこのようなテンプレートを用意します。

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

テンプレートが展開される際、%aはキャプチャコマンドを呼んだ場所へのリンクに変換されます。これは例えばメールからタスクを登録する時にとても役立ちます。タスクの定義を埋め、C-c C-cを押すと Org-mode はキャプチャ動作を始めた場所へまた連れ戻してくれます。

特定のテンプレートを対話的な選択なしに用いてキャプチャする特別なキーを定義するには、以下のようにキーバインドを作成できます。

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

9.1.3.1 Template elements

それではテンプレートを定義するための要素について述べてます。org-capture-templates中にあるそれぞれのエントリーは以下の項目から構成されるリストです。

keys　キーは文字列で示されるテンプレートを文字だけで選択します。例えば、"a"は a キーだけで選択するテンプレートであり、"bt"は２つのキーで選択します。幾つかのキーを使う場合、同じ接頭辞キーを持つキーはリストの中で連続している必要があり、以下に例をあげるような、接頭辞キーを表す２つの要素を持つエントリーをその前に置きます。

```
("b" "買い物リストのためのテンプレート")
```

もし *C*キーにテンプレートを定義していなければ、このキーでこの複雑な変数のカスタマイズバッファを開きます。

description
　選択時に表示されるテンプレートの短い説明文。

type　エントリーの種類をシンボルで表します。正しい値は以下の通りです。

- entry　見出しのついた Org-mode のノード。対象エントリーの子かトップレベルのエントリーとして書き込まれます。対象のファイルは Org-mode のファイルでなければならなりません。
- item　対象の場所の最初にある単純なリストへ書き込まれる単純なリストの項目。対象のファイルは Org-mode のファイルでなければならなりません。
- checkitem　チェックボックス項目。これは単純なリストとデフォルトのテンプレートが異なるだけです。
- table-line　対象の場所にある最初のテーブルへ追加する新しい行。行が追加される場所は:prependと:table-line-posプロパティに依存します。(下記参照)
- plain　そのまま挿入される文章

target　キャプチャされた項目が挿入されるべき場所の指定。Org-mode ファイルでは、通常ターゲットはノードで定義されます。エントリーはこのノードの子になります。他のタイプはこのノードの本文にあるテーブルやリストに追加されます。多くのターゲットの指定にはファイル名が含まれます。もしファイル名が空の文字

列だった場合、org-default-notes-fileがデフォルトになります。ファイルは変数や関数、Emacs Lisp のフォームでも与えられます。
正しい値は以下の通りです。

(file "path/to/file")
: 文章はこのファイルの先頭か最後に挿入されます。

(id "現存する org エントリーの id")
: このエントリーの子として、もしくはこのエントリの本文として記入します。

(file+headline "path/to/file" "node headline")
: 対象の見出しがファイル中で一つしか内場合の早い設定。

(file+olp "path/to/file" "Leval 1 heading" "Leval 2" ...)
: 唯一でない見出しの場合、フルパスの方が安全です。

(file+regexp "path/to/file" "regexp to find location")
: カーソルの位置に正規表現を使った場合。

(file+datatree "path/to/file")
: 今日の日付で日付ツリーの見出しを作ります。

(file+datetree+prompt "path/to/file")
: prompt で与えられた日付で日付ツリーの見出しを作ります。

(file+function "path/to/file" function-finding-location)
: ファイルの中で正しい場所を見つける関数

(clock)　ファイル中で現在時間を測っているエントリー。

(function function-finding-location)
: 一般的な方法、ファイルと場所を見つける自分の関数を書く方法。

template　キャプチャする項目を作るテンプレート。もし空にした場合、デフォルトのテンプレートを使用します。もしエスケープコードを付加した文字列の場合、キャプチャされた時の時間と状況に依存して置換されます。エスケープされた文字列はテンプレートファイルからロードされ、特別な構文 (file "path/to/template")を使用します。詳細をいかに示します。

properties
: エントリーの残りは追加オプションのプロパティリストです。理解できるプロパティは

:prepend　通常新しくキャプチャされた情報は対象の場所 (最後の子、最後のテーブル、最後のリスト項目...) に追加されます。このプロパティを設定することで変更します。

:immediate-finish
: セットされると情報の編集を行わなず、追加だけを直ちに行います。テンプレートが自動的に追加できる情報だけを必要とする場合だけ意味を成します。

:empty-lines
: 新しい項目の前後に挿入する行の数をこれに設定します。デフォルトは 0 で、通常の他の値は 1 です。

`:clock-in`
: この項目の時計を開始します。

`:clock-keep`
: キャプチャしたエントリーを追加しても時計を動かし続けます。

`:clock-resume`
: もし時計を割り込むキャプチャを始めた場合、キャプチャ終了時に時計を再開します。`:clock-keep`は`:clock-resume`より優先されることに注意が必要です。もしどちらの設定も`t`に設定された場合、現在の時計が動き、一つ前の時計は再開されません。

`:unnarrowed`
: 対象とするバッファの幅を狭めず、単純にフルバッファで表示します。デフォルトでは幅を狭くし、新しい内容だけが表示されます。

`:kill-buffer`
: 対象のファイルがキャプチャ時にまだ読み込まれていなかった場合、キャプチャ終了時に再びバッファを閉じます。

9.1.3.2 テンプレートの拡張

テンプレート自身の中では、特別な`%`-エスケープ[2] によって動的に内容を挿入できます。

`%^{`*prompt*`}` ユーザーに文字列を入力させこの順序と置換します。
: デフォルトの値と補完テーブルは以下のように指定します。
: %^{prompt|デフォルト値|補完 2|補完 3...}
: 矢印キーで入力の履歴をたどることができます。

`%a` 注釈、通常は `org-store-link`で作成されたリンク

`%A` `%a`と同様、しかし説明部分への入力を行います。

`%i` 初期の内容、キャプチャが呼ばれた時にアクティブに
: なっているリージョン全体。
: 全体のテキストは`%i`自身と同じ様にインデントされます。

`%t` タイムスタンプ、日付のみ

`%T` 時間と日付からなるタイムスタンプ

`%u, %U` 上記と同じだが、不活性なタイムスタンプ

`%^t` `%t`と同じだが、日付の入力を行います。`%^T`、`%^u`、`%^U`と似てま
す。
: プロンプトを`%^{Birthday}t`の様に定義できます。

`%n` ユーザーネーム (`user-full-name`から取ってきます)

`%c` 現在のキルリングの先頭

`%x` X クリップボードの内容

`%^C` キルかクリップどちらを使うか対話的に選択します。

`%^L` `%^C`と同様だが、リンクとして挿入します。

`%k` 現在計時しているタスクのタイトル。

`%K` 現在計時しているタスクへのリンク。

`%f` org-capture が呼ばれた時に現在のバッファーで表示していたファイル。

`%F` `%f`と同様だが、フルパスを含んでいます。

`%^g` タグの入力を対象ファイル中のタグから補完して入力します。

`%^G` タグの入力をアジェンダファイルすべてから補完して入力します。

`%^{`*prop*`}p` ユーザーに *prop*プロパティの値の入力を行わせます。

`%:keyword` あるリンクのタイプを指示する特定の情報

`%[`*file*`]` *file*で与えられるファイルへ内容を挿入します。

`%(`*sexp*`)` *sexp*で与えられる Elisp を評価してその結果と入れ替えます。

[2] もしこれらの文字通りの並びが必要な場合は、`%`をバックスラッシュでエスケープすること

リンクのタイプを指定するため、以下のキーワードが定義[3] されています。

リンクタイプ	使用可能なキーワード
bbdb	%:name %:company
irc	%:server %:port %:nick
vm, wl, mh, mew, rmail	%:type %:subject %:message-id
	%:from %:fromname %:fromaddress
	%:to %:toname %:toaddress
	%:date (ヘッダーメッセージ中にあるメッセージの日付)
	%:date-timestamp (アクティブなタイムスタンプとしての日付)
	%:date-timestamp-inactive (アクティブでないタイムスタンプとしての日付)
	%:fromto (\"to NAME\"か\"from NAME\"のどちらか)[4]
gnus	%:group, メッセージとそれに加えてすべての email フィールドのため。
w3, w3m	%:url
info	%:file %:node
calendar	%:date

テンプレート展開後のカーソル位置のために以下を用います。

%?　テンプレートを完了したあと、カーソル位置をここに移動します。

9.2 Attachments

参照すべきものをアウトラインノートやタスクと連携させることは大抵役立ちます。小さなプレーンテキストの塊は単純にプロジェクトのサブツリーとして保存可能です。ハイパーリンク (第 4 章「Hyperlinks」p.36 を参照) はあなたのコンピュータやクラウドなどにあるファイル、例えばプロジェクトに関連する email やソースコードファイルとの連携を確立します。その他の方法として、*attachments* があります。これはアウトラインノードの属するディレクトリにあるファイルです。これらのディレクトリは 'data'ディレクトリの中に位置し、あなたの Org ファイルが保存されているディレクトリと同じディレクトリの中にある[5]。もしこのディレクトリを git initで初期化した場合、Org-mode は変更点を見つけた時に自動的にそれらをコミットします。添付システムは John Wiegley の貢献によって Org-mode へ追加されました。

もしそれがより良い方法に思える場合は、エントリーへディレクトリを添付することも自身の選択で可能です。子エントリーは親から添付ディレクトリを受け継ぐため、サブツリー全体が同じ添付ディレクトリを使うことになります。

以下のコマンドはプロパティを操作する助けとなります。

C-c C-a　org-attach

添付システムに関連するコマンドのコマンド選択画面。これらのキーのあとに、コマンドのリストが表示され、コマンドを選択するために更にキーを押す必要があります。

---

[3] もしあなたが自分のリンクタイプを定義しているならば ("char 65.3 節「Adding hyperlink types」p.195 を参照)、org-store-link-propsに保存されているいずれのプロパティもキャプチャテンプレート中に同じ方法でアクセス可能です。

[4] これはユーザーではなく常に他人になる。org-from-is-user-regexpを参照すること。

[5] もしエントリーや Org ファイルを他のディレクトリに移動した場合、org-attach-directoryが絶対パスを含むように設定する必要があります。

`a`　`org-attach-attach`
ファイルを選択してそれをタスクの添付ディレクトリへ移動します。ファイルは`org-attach-method`によって複製、移動、リンクされます。ハードリンクはすべてのシステムでサポートされていないことに注意が必要です。

`c/m/l`　コピー/移動/リンクメソッドを使ってファイルを添付します。ハードリンクはすべてのシステムでサポートされていないことに注意が必要です。

`n`　`org-attach-new`
Emacs のバッファとして新しい添付を作成します。

`z`　`org-attach-sync`
あなた自身で添付を追加した場合に、現在のタスクをその添付ディレクトリと同期します。

`o`　`org-attach-open`
現在のタスクに関する添付を開きます。もし 1 つ以上ある場合は、まずファイル名を入力させます。開き方は`org-file-apps`に従います。詳細は、ハイパーリンクを辿るための情報 (4.4 節「Handling links」p.38 を参照) を参照してください。

`O`　`org-attach-open-in-emacs`
これも添付を開きますが、強制的にファイルを Emacs を用います。

`f`　`org-attach-reveal`
現在のタスクの添付ディレクトリを開きます。

`F`　`org-attach-reveal-in-emacs`
これもディレクトリを開きますが、強制的に Emacs 内で`dired`を用います。

`d`　`org-attach-delete-one`
添付の 1 つを選択し削除します。

`D`　`org-attach-delete-all`
タスクに関連する添付をすべて削除します。安全な方法は`dired`を用いてディレクトリを開き、そこから削除する方法です。

`s`　`org-attach-set-directory`
特定のディレクトリをエントリーの添付ディレクトリーに指定します。これはディレクトリーのパスを`ATTAH_DIR`プロパティに代入することで動作します。

`i`　`org-attach-set-inherit`
`ATTACH_DIR_INHERIT`プロパティをセットします。これによって子も親と同じディレクトリを添付として使用します。

9.3 RSS フィード

Org-mode はエントリーの追加や変更を RSS フィードと Atom フィードの情報を元に行うことができます。この機能を使ってプロダクトフィードのなかからそれぞれの新しいプロダ

クトに対してタスクを作ることも可能です。もしくは、携帯電話を対象とした Web 上のノート作成サービスを Org-mode のタスクにインポートできます。フィードにアクセスするには、org-feed-alistを設定します。この変数のドキュメント文字列に詳細はあります。以下に例をしましょう。

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

この例は、以下のコマンドが使われるたびに、rss.slashdot.orgで提供されているフィードの新しいアイテムから、‘~/org/feeds.org’ファイル内に ‘Slashdot Entries’をヘッダーにもつ新しいエントリーを生成します。

C-c C-x g　org-feed-update-all
C-c C-x g　org-feed-alistに設定されたフィードからアイテムを収集し、上記のように振る舞う。

C-c C-x G　org-feed-goto-inbox
はフィード名を入力し、そのフィードに設定されたインボックスへ移動します。

幾つかの見出しでは、Org-mode は ‘FEEDSTATUS’という引き出しを作成します。これはフィード中にあるアイテムのステータス情報を保存しています。なんども同じアイテムが追加されないためには、‘FEEDSTATUS’をファイルの引き出しの中に追加することが必要です。

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

Atom フィードの読み方などのより詳しい情報は、‘org-feed.el’を御覧ください。また、org-feed-alistのドキュメント文字列もあります。

9.4 外部アクセスのためのプロトコル

Org-mode を扱うためのプロトコルを外部のアプリケーションから、‘emacsserver’を通じて Emacs に渡すことができます。例えば、ウェブブラウザーのブックマークをキャプチャ(9.1 節「Capture」p.85 を参照) を使って現在のページへのリンクとして Org-mode に送り、新しいノートを作るように設定できます。または、Emacs に現在ウェブブラウザーで開いているウェブサイトのローカルにあるソースファイルを開くようなブックマークを作ることも出来ます。詳細な説明や設定方法は http://orgmode.org/worg/org-contrib/org-protocol.phpを御覧ください。

9.5 Refiling notes

キャプチャしたデータを見なおしているとき、幾つかのエントリーをプロジェクトなどの異なるリストへ移動したい時があるでしょう。カットし、正しい位置をさがし、ノートを貼り付けるのは面倒です。これを単純にするには、以下に示す特別なコマンドを使います。

C-c C-w　org-refile
エントリーや現在のリージョンをリファイルします。このコマンドはエントリーをリファイルする場所を求め、補完とともに 1 つ選択させます。アイテム (またはリージョン内のアイテム) は対象の見出しの下にサブアイテムとして挿入されます。org-reverse-note-orderに依存して、サブアイテムの先頭か最後のどちらかに挿入されます。
デフォルトでは、現在のバッファのレベル 1 の見出しすべては対象と考えられま

すが、いくつものファイルにまたがった複雑な定義もすることが可能です。詳細は org-refile-targets変数をご覧ください。もし場所をアウトラインのパスをファイルパスのような補完で選択したければ、org-refile-use-outline-path と org-outline-path-complete-in-steps変数をご覧ください。もしリファイルするノードの親ノードをその場で作れるようにしたいときは、org-refile-allow-creating-parent-nodes変数をご覧ください。org-log-refile変数[6] がセットされている場合、タイムスタンプやノートがエントリーがリファイルされたときに記録されます。

C-u C-c C-w
リファイルのインターフェースを見出しのジャンプに使います。

C-u C-u C-c C-w　org-refile-goto-last-stored
org-refileが最後に木を移動させた場所に移動します。

C-2 C-c C-w
現在計時しているアイテムの子としてリファイルします。

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
対象のキャッシュを削除します。リファイル対象のキャッシュは org-refile-use-cacheを設定することで設定します。コマンドに新しい対象を見せるために、このコマンドでキャッシュを削除する必要があります。

9.6 Archiving

(サブ) ツリーとして表現されたプロジェクトが終わった時、ツリーを外に移動し、それがアジェンダに現れない様にしたいでしょう。アーカイブは活動中のファイルをコンパクトにし、アジェンダビューを作るようなグローバルな検索を早くするために重要です。

C-c C-x C-a　org-archive-subtree-default
現在のエントリーを org-archive-default-command変数で指定されたコマンドを使ってアーカイブします。

9.6.1 ツリーをアーカイブファイルへ移動

もっとも一般的なアーカイブアクションはプロジェクトを他のファイル、アーカイブファイル、へ移動させることです。

C-c C-x C-s or short *C-c $*　org-archive-subtree
カーソルの場所から始まるサブツリーを org-archive-locationで与えられる場所にアーカイブします。

C-u C-c C-x C-s
現在の見出しにある子がアーカイブへ移動可能か調べます。これを行うためには、それぞれのサブツリーがオープンな TODO エントリーとしてチェックされている必要がります。もし 1 つも見つからなければ、コマンドはこれをアーカイブの場所へ移動するか訪ねてきます。もしコマンドが入力されたときに、カーソルがへっどらいんでない場合、レベル 1 のツリーがチェックされます。

[6] #+STARTUPキーワード、logfile、lognoterefile、そして nologrefileに対応している。

通常のアーカイブ場所は現在のファイルと同じディレクトリにある、現在のファイル名に‘_archive’を付加した名前のファイルです。これを変更するための情報や例は、org-archive-location変数のドキュメント文字列をご覧ください。以下に示すような、バッファ内でこれを変更するためのオプションもあります[7]。

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

もしあるエントリー又は (サブ) ツリーに対して特別なアーカイブ場所を指定したいときには、エントリーに:ARCHIVE:プロパティを場所を値として与えてください (第 7 章「Properties and Columns」p.61 を参照)。

もしサブツリーが移動した場合、エントリーが移動してきたファイル、アーカイブしたときのアウトラインパスなどのコンテクスト情報が記録されます。org-archive-save-context-info変数を設定することで追加される情報の量を調整します。

9.6.2 ファイル内部でのアーカイブ

もし異なるファイルへサブツリーを移動させずに、それをアジェンダビューに表示させないようにするには、ARCHIVE tagを使うことができます。

ARCHIVE タグ (第 6 章「Tags」p.57 を参照) でマークされている見出しはアウトラインツリー内の場所に留まりますが、下記のような振る舞いをします。

- 表示を切り替えるコマンド (2.3 節「Visibility cycling」p.7 を参照) では開くことはできません。アーカイブしたサブツリーを強制的に切り替えるには *C-TAB*を使うか、org-cycle-open-archived-treesオプションを設定します。また、show-allなどの通常のアウトラインコマンドはアーカイブしたサブツリーも開きます。
- ツリーの抽出を行う過程で (2.6 節「Sparse trees」p.12 を参照)、アーカイブしたサブツリー内でマッチしたものは org-suparse-tree-open-archived-treesオプションを設定しない限り現れません。
- アジェンダビューの抽出を行う過程で (第 10 章「Agenda Views」p.96 を参照)、アーカイブしたツリーの内容は、org-agenda-skip-archived-treesを設定した場合は無視され、設定されない場合は常に表示されます。アジェンダでは、v aを押すことで一時的にアーカイブを含むことができます。
- アーカイブされたツリーは見出し以外はエクスポートされません (第 12 章「Exporting」p.132 を参照)。org-export-with-archived-trees変数を使って詳細な設定は行なってください。
- アーカイブしたツリーは org-columns-skip-archived-trees変数が nilに設定されない限りカラムビューから除外されます。

以下に示すコマンドが ARCHIVE タグの編集に役立ちます。

C-c C-x a　org-toggle-archive-tag

現在の見出しの ARCHIVE タグをトグルする。もしタグが設定されているならば、見出しは shadowed face に変更され、以下のサブツリーは隠されます。

C-u C-c C-x a

現在の見出しが持つ直接の子がアーカイブされるべきかチェックする。これをおこなうには、それぞれのサブツリーがオープンな TODO エントリーかチェックさ

[7] 後方互換性のために、もしこれらの行がファイル中にいくつか存在する場合、それ以下のテキストのアーカイブ場所をそれぞれ指定します。最初の行はその定義以前のすべてのテキストに適応されます。しかし、この方法は文書のアウトライン構造と互換性が無く、全く推奨されません。複数のアーカイブ場所をバッファ内で設定する正しい方法は、プロパティを使う方法です。

れる。もし何も見つからなかった場合、コマンドは子に ARCHIVE タグをセットする。もしカーソルがコマンド実行時に見出し上に無い場合、レベル 1 のツリーがチェックされる。

C-TAB　`org-force-cycle-archived`
もし ARCHIVE タグがついていてもツリーの切り替えを行う

C-c C-x A　`org-archive-to-archive-sibling`
現在のエントリーをアーカイブ兄弟に移動する。これは見出しが ‘Archive’か ‘ARCHIVE’タグの付いたエントリーの兄弟である。このエントリーはこの兄弟の子になる。そのため、継承したタグやアウトライン内のだいたいな位置など、従来の状況は保ち続けている。

10 アジェンダビュー

Org-mode で作業した結果、TODO アイテム、タイムスタンプのついたアイテム、タグの付いた見出しなどが、1 つのファイル、あるいはいくつものファイルにまたがって、撒き散らされることとなります。ある特定の日に重要な、実際に動いているアイテムやイベントの全体像を把握するためには、ひとつの管理された方法で、これらの情報を集めたり、並び替えたりしながら、表示することが必要です。

Org-mode では、いろいろな基準によってアイテムを選択することが可能であり、独立したバッファにそれらのアイテムを表示させることができます。7 つの異なるビューのタイプが用意されています。:

- アジェンダ　カレンダーのように指定した日付の情報を表示します、
- *TODO* リスト　未完了のアクションアイテムをカバーします、
- マッチビュー　関連づけられているタグやプロパティ、TODO の状態に基づいて見出しを表示します、
- タイムラインビュー　1 つの Org-mode のファイルの中に含まれている全てのイベントを時間順のビューに表示します、
- a テキストの検索ビュー　複数のファイルの中かから、指定したキーワードを含んでいるすべてのエントリーを表示します、
- a 詳細が未決定のプロジェクトビュー　現在作業が進んでいないプロジェクトを表示します。そして、
- カスタムビュー　特別な検索や異なるビューの組合せによるビューです。

抽出された情報は特別なアジェンダバッファに表示されます。このバッファはリードオンリーですが、オリジナルの Org-mode ファイルにジャンプしたり、オリジナルのファイルを間接的に編集することができます。

2 つの変数によって、アジェンダバッファをどのように表示するか、アジェンダが存在したときに、ウインドウの設定を元に戻すかどうかをコントロールします。;`org-agenda-window-setup`と `org-agenda-restore-windows-after-quit`.

10.1 Agenda files

表示される情報は、通常すべてのアジェンダファイルから収集されます。アジェンダファイルは `org-agenda-files`[1] 変数にリスト化されたファイルが対象となります。もしもこのリストの中にディレクトリ名が記載されていたら、そのディレクトリの中にある‘`.org`’という拡張子がついた全てのファイルが、アジェンダファイルの対象となります。

したがって、たとえあなたが 1 つの Org-mode ファイルでしか作業をしていなくても、このファイルをそのリスト[2] に記載したことになるでしょう。`org-agenda-files`をカスタマイズすることが可能で、しかも以下に述べるコマンドを通して簡単な方法で維持することができます。

---

[1] もしもその変数の値がリストではなく、単独のファイル名の場合には、その外部ファイルの中に記載されているアジェンダファイルの名前となります。

[2] コマンド選択画面を使用しているときに、コマンドを選択する前に、<を押すと、編集中のファイルに対するコマンドが制限されて、次のコマンド選択画面でコマンドが入力されるまで、`org-agenda-files`は無視されます。

C-c [　　org-agenda-file-to-front
　アジェンダファイルのリストに編集中のファイルを追加する。そのファイルは、リストの先頭に追加される。もしも既にリストに存在していたら、先頭に移動する。前置引数をつけることで、リストの最後に追加／移動する。

C-c]　　org-remove-file
　編集中のファイルをアジェンダファイルのリストから削除する。

C-'　　org-cycle-agenda-files
C-,　　アジェンダファイルのリストに従って、1 つのファイルから次のファイルへと切り替える。

M-x org-iswitchb
　iswitchbと似たようなインターフェースで Org-mode のバッファの間を切り替えるコマンド。

Org-mode メニューには、現時点のファイルのリストが含まれており、その中のファイルに移動するのに役立ちます。

もしもこのリストに載っているファイルではなく、作業中のアジェンダファイルに焦点をあてたかったり、リストにあるファイルのまさにひとつのファイルに焦点をあてたかったり、はたまたあるファイルの中のあるサブツリーに焦点をあてたかったりしたいときは、いくつかの方法が用意されています。単一のアジェンダコマンドとして、コマンド選択画面上 (10.2 節「Agenda dispatcher」p.97 を参照) で<を 1 回ないし数回押すとよいのです。アジェンダの対象をある限定した期間に絞り込むために以下のコマンドが用意されています。:

C-c C-x <　　org-agenda-set-restriction-lock
　アジェンダの対象を現在カーソルが置かれているサブツリーに固定的に制限します。前置引数をつけたり、ファイルの最初の見出しよりも前にカーソルが置かれているときには、アジェンダの対象範囲はファイル全体になります。この制約は *C-c C-x >*を実行して取り除くか、<または>をアジェンダのコマンド選択画面上で入力するまでは維持します。もしもウインドウ上にアジェンダビューが表示されているならば、あたらしい制約が即座に効果を及ぼします。

C-c C-x >　　org-agenda-remove-restriction-lock
　*C-c C-x <*で作成された固定する制限を削除します。

'speedbar.el'を併用しているときは、Speedbar のフレームの中で以下のコマンドを使用することができます。

< in the speedbar frame　　org-speedbar-set-agenda-restriction
　Speedbar のフレームの中で、1 つの Org-mode ファイルか、そのファイルのサブツリーの一つか、カーソルの置かれているアイテムに対応してアジェンダを恒久的に限定します。もしもアジェンダビューが表示されているウインドウがあるならば、限定箇所が変更されると即座に反映する。

> in the speedbar frame　　org-agenda-remove-restriction-lock
　制限をふたたび解除する。

10.2 アジェンダのコマンド選択画面

グローバルなキーと結びついている、コマンド選択画面を通してそのビューは作成されます。—例えば、*C-c a* (1.2 節「Installation」p.3 を参照) 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。(10.3.1 節「Weekly/daily agenda」p.99 を参照)

`t` / `T`　すべての TODO アイテムのリストを作成します。(10.3.2 節「Global TODO list」p.100 を参照)

`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。(10.3.3 節「Matching tags and properties」p.101 を参照)

`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。(10.3.4 節「Timeline」p.104 を参照)

`s`　そのエントリーに存在するしないにかかわらず、and/or という正規表現によるキーワードの論理式で選択したエントリのリストを作成します。

`/`　すべてのアジェンダファイルと `org-agenda-text-search-extra-files`の中でリスト化かれているファイルの中から正規表現を用いて検索します。これは Emacs の `multi-occur`というコマンドを使用します。前置引数をつけると、それぞれのマッチした行の状況の数をしていすることができます。デフォルトは 1 となっています。

`#` / `!`　詳細が未決定のプロジェクトのリストを作成します。(10.3.6 節「Stuck projects」p.104 を参照)

`<`　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。10.6 節「Custom agenda views」p.116 を参照してください.

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。

[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。.

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。

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、10.5 節「Agenda commands」p.107 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)
```

その後、すべてが自動的に行われます。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
```

[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 (10.6.2 節「Block agenda」p.117 を参照). をその代わりに利用することが多いです。

```
#+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
```

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 (第 10 章「Agenda Views」p.96 を参照) から TODO アイテムを 1 つのバッファに

[6] Note that the order of the arguments (month, day, year) depends on the setting of `calendar-date-style`.

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 (10.5 節「Agenda commands」p.107 を参照)。上と似ていますが、指定した TODO キーワードと合致したものを表示します。同じことを前置引数をつけて *C-c a t* を実行することでも指定できます。キーワードの入力を促す指示が表示され、そして複数のキーワードを論理式 OR という意味で '|' で区切って指定することができます。数字付きの前置引数をつけると org-todo-keywords の中の N 番目のキーワードを選択することができます。*r* キーをアジェンダバッファで使用するとバッファの再構成が行われます。たとえば *3 r* というように、前置引数をつけてこのコマンドを実行すると選択した TODO キーワードが変更することができます。もしも特定のキーワードを使って検索することが多い場合は、カスタムコマンドを定義することもできます (10.2 節「Agenda dispatcher」p.97 を参照)。
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 (6.3 節「Tag searches」p.59 を参照)。

リモートで TODO アイテムを編集するということの意味は、1 つのキーを入力することで TODO エントリーの状態を変更できるということです。TODO リストの中で利用できるコマンドは 10.5 節「Agenda commands」p.107 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*(8.1 節「Timestamps」p.70 を参照) 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*(第 6 章「Tags」p.57 を参照) がついていたり、あるいは属性 (第 7 章「Properties and Columns」p.61 を参照) 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 (第 6 章「Tags」p.57 を参照)。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 (10.2 節「Agenda dispatcher」p.97 を参照)。

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。6.3 節「Tag searches」p.59 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 10.5 節「Agenda commands」p.107 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 (第 7 章「Properties and Columns」p.61 を参照) 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 (7.2 節「Special properties」p.62 を参照) 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 7.4 節「Property inheritance」p.63 参照。

逆互換として、さらにまたタイプのスピードを上げるために、検索において TODO の状態をテストする別の方法があります。このためには、検索文字列（それは ‘|’で結合された複数の用語が含まれていると思いますが）のタグ・属性検索の部分を ‘/’を使って終了させ、TODO キーワードを論理式で結んで指定します。その構文はタグの検索で使用したのと似ていますが、よく考えて適用する必要があります。例えば、複数の TODO キーワードが存在することを検索するには論理式の AND で結びつけても意味がありません。しかしながら、*negative selection*（存在しないことを選択する場合）では「AND」で結合することは意味を持ちます。これを確かめるには、実際にいくつかの TODO キーワードで、*C-c a M*を用いて確認するだけです（そのほうがスピードアップできます）。あるいはスラッシュのあとに ‘!’を記入して同時に TODO の部分を開始します。*C-c a M*または ‘/!’を使用したときは、DONE の状態にある TODO キーワードを検索することはできません。例えば、

‘work/WAITING’
　　‘work+TODO="WAITING"’と同じ

‘work/!-WAITING-NEXT’
‘:work:’を選択。ただし TODO 行では‘WAITING’と‘NEXT’のどちらのタグもついていないもの

‘work/!+WAITING|+NEXT’
‘:work:’を選択。TODO 行に‘WAITING’か‘NEXT’かどちらかのタグがついているもの。

10.3.4 Timeline for a single file

タイムラインはひとつの Org-mode ファイルの中から *time-sorted view*（時間順のビュー）ですべてのタイムスタンプのついたアイテムをまとめてて表示します。このコマンドの主な目的は、あるプロジェクトに含まれているイベント全体の概要をつかむためにあります。

C-c a L org-timeline
すべてのタイムスタンプの付いたアイテムについて、Org-mode ファイルの中で時間順のビューを提供します。*C-u*という前置引数をつけて呼び出したときは、現在の日付の時点で、すべての未完了の TODO エントリー（予約されているものも、そうでないものも）を一覧にします。

タイムラインのバッファで利用できるコマンドは、10.5 節「Agenda commands」p.107 にリスト化されています。

10.3.5 Search view

アジェンダのビューでは Org-mode のエントリーに対する一般的なテキスト検索機能を持っています。これはノートを探すのに特に役に立ちます。

C-c a s org-search-view
このコマンドは特別な検索のためのもので、論理式を使って、文字列または特定の単語に合致するエントリーを選択します。

例えば、‘computer equipment’という検索文字列は、‘computer equipment’という 1 つの文字列が含まれているエントリーを検索するでしょう。もしも、２つの単語が、1 つ以上のスペースまたは改行で分かれていても、依然として一致するものを検索するでしょう。検索ビューでは、エントリーの中にある特別なキーワードについて論理式を使って検索することもできます。‘+computer +wifi -ethernet -{8\.11[bg]}’ という検索文字列では、次のようなノートエントリーを検索します。computerと wifiというキーワードを含んでおり、ethernetというキーワードは含まれておらず、8\.11[bg]という正規表現を含んでいない、すなわち 8.11b および 8.11g ともに含まれていないという意味ですが、エントリーを検索します。最初の‘+’は単語検索を開始するために必要ですが、ほかの‘+’はオプションです。詳しく知りたい場合は、org-search-viewというコマンドのドキュメント文字列を参照してください。

アジェンダファイルに加えて、このコマンドは org-agenda-text-search-extra-files の中で一覧になっているファイルもまた検索するということに注意してください。

10.3.6 Stuck projects

もしもあなたが、以下に述べるような David Allen 氏の GTD のようなシステムであなたの仕事を管理しているならば、あなたが抱えている「義務」のひとつは、すべてのプロジェクトが進んでいるかを明確にするために、レビューを定期的に行うことです。詳細が未決定のプロジェクトは、次の行動が何も定義がされていないため、Org-mode が提示する TODO リストに、全く何も表示されることがないのです。レビューをする際に、そういったプロジェクトを明確にし、それらのプロジェクトための次の行動を定義することが必要です。

C-c a # org-agenda-list-stuck-projects
詳細が未決定のプロジェクトリスト

C-c a ! org-stuck-projectsの変数をカスタマイズすることで何が詳細が未決定のプロジェクトで、どうやったらそういうプロジェクトを発見できるかを定義することができます。

あなたは九分九厘このコマンドが機能するために、このビューを定義する必要があります。あらかじめビルトインされているデフォルトの設定では、すべてのあなたのプロジェクトは第 2 階層の見出しに記述されており、あるプロジェクトが未決定であるとはいえない状況とは、すくなくとも 1 つのエントリーに TODO または NEXT または NEXTACTION という印がつけられている場合です。

Org-mode を使う際に、あなた自身の方法でアプローチするとして、PROJECT というタグがあるものをプロジェクトと定義し、プロジェクトがまだ検討する段階にないということを示すために TODO キーワードで MAYBE と書いているものと仮定しましょう。さらに TODO キーワードで DONE という印の付いたものは完了したプロジェクトであると仮定しましょう。そしてまた NEXT もしくは TODO と書かれたものは NextAction であると仮定しましょう。@SHOP というタグがついたときは NEXT というタグが付いていなくても、ショッピングに行くという次の行動を示しているとします。最終的に、もしもプロジェクトに IGNORE（無視）という特別なキーワードがどこかについていたら、それはリストに表示されないものとします。このようなケースの場合、タグ・TODO[7] が '+PROJECT/-MAYBE-DONE' とマッチし、さらにサブツリーに TODO、NEXT、@SHOP、および IGNORE というタグが付いているようなプロジェクトは、詳細が未決定のプロジェクトではないといえます。このようなカスタマイズを正しく定義するには、

```
(setq org-stuck-projects
      '("+PROJECT/-MAYBE-DONE" ("NEXT" "TODO") ("@SHOP")
                               "\\<IGNORE\\>"))
```

もしもあるプロジェクトが詳細が未決定のプロジェクトではないと定義されたならば、そのエントリーのサブツリーは依然として詳細が未決定のプロジェクトとして検索されるということに注意してください。

10.4 Presentation and sorting

アジェンダビューにアイテムが表示される前に、Org-mode ではそのアイテムを表示し並び替える準備を行っています。それぞれのアイテムは 1 行を占めます。その行にはその項目の *category* (10.4.1 節「Categories」p.105 を参照) 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]

[7] 6.3 節「Tag searches」p.59 を参照してください.

[8] 逆に言うと、以下のような動作も生じます。もしも 1 つのファイルの中に、いくつかののそういう行が存在するならば、それよりも下の行にあるテキストに、そのカテゴリーをそれぞれ指定することになります。最初の

```
#+CATEGORY: Thesis
```

もしもあなたが、1 つのエントリーもしくは 1 つの（サブ）ツリーに特別な CATEGORY を持たせたいと望むのならば、そのエントリーに、値として適用したいと思っている特別なカテゴリーを:CATEGORY:という属性に設定しなさい。

アジェンダバッファの表示は、そのカテゴリーが 10 文字以上長くしない方が見栄えが良いです。

あなたは org-agenda-category-icon-alist変数をカスタマイズすることで、カテゴリーにアイコンを設定することができます。

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 (10.3.1 節「Weekly/daily agenda」p.99 を参照) と一体化されていたときは、ダイアリーのエントリーの中で指定した時間は、同様に認識されます。

アジェンダの表示のために、Org-mode は時間を引き出し、前置引数の一部として標準的な 24 時間のフォーマットでそれを表示します。前の段落に書かれた時間の例は、アジェンダの中で結局以下のように表示されます。

```
 8:30-13:00 Arthur Dent lies in front of the bulldozer
12:45...... Ford Prefect arrives and takes Arthur to the pub
19:00...... The Vogon reads his poem
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

もしもアジェンダが一日モードであるならば、あるいは今日を表示しているならば、時間設定されたエントリーは、次のような時間のグリッドに埋め込まれます。

```
 8:00...... ------------------
 8:30-13:00 Arthur Dent lies in front of the bulldozer
10:00...... ------------------
12:00...... ------------------
12:45...... Ford Prefect arrives and takes Arthur to the pub
14:00...... ------------------
16:00...... ------------------
18:00...... ------------------
19:00...... The Vogon reads his poem
20:00...... ------------------
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

時間のグリッドは、org-agenda-use-time-grid変数で表示したりしなかったさせることができます。そしてまた org-agenda-time-gridで設定をすることができます。

カテゴリーは、その最初の CATEGORY の行はよりも前にあるどのテキストにも適用されます。しかしながら、*strongly* という手法を使うことは、文書のアウトライン構造と非互換であることを、強く非難することになります。複数のカテゴリーをバッファの中で設定する正しい方法は属性を使用することです。.

10.4.3 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 (5.4 節「Priorities」p.53 を参照) に従って並び替えられます。優先順位は基本的な優先順位で構成されます（優先順位 '`A`'ならば 2000、'`B`'ならば 1000、'`C`'ならば 0 として)。さらに、予定あるいはデッドラインを過ぎているアイテムのウエイトが追加されます。
- TODO リストでは、各アイテムはカテゴリーの順番に並び替えられますが、各カテゴリーの中では、優先順位 (5.4 節「Priorities」p.53 を参照) によって並び替えられます。優先順位は、優先順位の記号に従って並べ替えられます。さらに、アイテムが実行する日あるいは予約した日にどれだけ近いかということも考慮されます。
- タグでの一致については、項目は並び替えは行われず、アジェンダファイルの中で一致した項目が発見された順番に従って表示されるのみです。

並び替えは、`org-agenda-sorting-strategy`変数でカスタマイズすることができます。そして、並び替えはそのエントリーの工数の見積りに基づく評価も含まれます。

10.5 Commands in the agenda buffer

アジェンダバッファでのエントリーは、その項目が作成された Org-mode ファイルと日記ファイルの間でリンクされます。アジェンダバッファでは編集することはできませんが、コマンドを使って、そのエントリーがある場所を表示したり、ジャンプして、アジェンダバッファから「遠隔的に」Org-mode ファイルを編集することができます。この方法で、すべての情報は 1 度書き込めばよく、あなたがアジェンダとノートのファイルが別の情報になるというリスクを避けることができます。

いくつかのコマンドはアジェンダの行上でマウスをクリックすることで実行されます。それ以外のコマンドは、必要とされる行の中にカーソルが置かれている必要があります。

Motion

n　`org-agenda-next-line`
次の行へ (`up`及び *C-p*と同じ)。

p　`org-agenda-previous-line`
次の行へ (`down`及び *C-n*と同じ)。

View/Go to Org file

SPC or *mouse-3*　`org-agenda-show-and-scroll-up`
そのアイテムのオリジナルの場所を別のウインドウで表示する。前置引数を使うことで、見出しだけでなく、アウトライン上にエントリー全体を明確に表示する。

L　`org-agenda-recenter`
オリジナルの場所を表示し、ウインドウのセンターに再配置する。

TAB or *mouse-2*　org-agenda-goto
別のウインドウでそのアイテムのオリジナルの場所に移動する。

RET　org-agenda-switch-to
そのアイテムのオリジナルの場所に移動し、他のウインドウは削除する。

F　org-agenda-follow-mode
Follow モードをトグルする。Follow モードではアジェンダバッファ上でカーソルを動かすと、Org-mode ファイルの中で、別のウインドウ上で対応する場所を表示する。新しいアジェンダバッファの中でこのモードの初期設定値は、org-agenda-start-with-follow-mode変数で設定することができる。

C-c C-x b　org-agenda-tree-to-indirect-buffer
間接的なバッファの中で可憐とアイテムのサブツリー全体を表示する。数値付きの前置引数 N をつけると、第 N 階層まで階層を上がり、そのツリーを取得する。もしも N がマイナスならば、多くの階層まで上がる。*C-u*という前置引数を付けた場合は、既に使われた間接的バッファは消去されない。

C-c C-o　org-agenda-open-link
エントリーの中にあるリンクをフォローする。この機能は、参照されている Org-mode のノードに属しているテキストの中に含まれているいくつかのリンクの中から選択するという機能を提供する。もしもリンクが 1 つしかない場合は、選択画面を表示せずに、そこにリンクを貼る。

Change display

o　他のウインドウを削除します。

v d or short *d*　org-aganda-day-view
v w or short *w*　org-aganda-day-view
v m　org-agenda-month-view
v y　org-agenda-month-year
v SPC　org-agenda-reset-view
日／週／月／年のビューを切り替えます。日または週にビューを切り替えたときは、この設定は、それに続くアジェンダの更新ついてのデフォルトの設定となります。月および年のビューは、作成するために時間を要するので、デフォルトとはしていません。数字の付いた前置引数をつけると、その年、ISO の週、月、年の指定した日に直接ジャンプします。例えば *32 d*と書いたときは 2 月 1 日、*9 w* と書いたら ISO の週番号が 9 を指します。日、週あるいは月のビューを設定したときは、1 年は同様に前置引数の中でコード化されます。例えば、*200712 w*と書いたときは 2007 年の第 12 週にジャンプするでしょう。もしもそのような年の指定を、1 桁もしくは 2 桁の数字で行いたいたときは、1938 年から 2037 年の間に位置づけられます。*v SPC*によって、org-agenda-spanでの設定をリセットすることができます。

f　org-agenda-later
時間を前の日付の表示へと遡ります。

.　org-agenda-goto-today
今日へ移動します。

j　org-agenda-goto-date
日付の選択画面でその日に移動します。

J　org-agenda-clock-goto
アジェンダバッファの中で現在時間を計測中のタスクに移動します。

D　org-agenda-toggle-diary
日記のエントリーに含めるかどうかトグルします。参照 10.3.1 節「Weekly/daily agenda」p.99.

v l or short *l*　org-agenda-log-mode
Logbook mode にするかどうかをトグルします。Logbook mode の中では、ログの取得中に (変数 org-log-done)DONE と印が付けられたエントリーが、その日の時刻を持っているエントリーとして、アジェンダの中に表示されます。org-agenda-log-mode-items変数を用いて log モードに含まれるエントリーのタイプを設定することができます。*C-u*という前置引数をつけて呼び出すと、状態の変化を含め、すべてのおこりうる logbook のエントリーを表示できるでしょう。*C-u C-u*という 2 つの前置引数をつけて呼び出すと、ログの情報のみが表示され、それ以外は表示されません。*v L*は、*C-u v l*と等価です。

v [or short *[*　org-agenda-manipulate-query-add
現在のビューに、不活性のタイムスタンプを含めます。週／日のアジェンダとタイムラインビューのみです。

v a　org-agenda-archives-mode
v A　org-agenda-archives-mode 'files
Archives モードをトグルします。Archives モードでは、ARCHIVEDと印されたツリーもまたアジェンダを作成するときにスキャンされます。大文字の *A*を使用したときは、全てのアーカイブファイルを含みます。archives mode から出るためには、再度 *v a*を押してください。

v R or short *R*　org-agenda-clockreport-mode
Clockreport モードをトグルします。Clockreport モードでは、日／週のアジェンダは、時間軸のための時刻のついた表を表示し、カレントのアジェンダビューでカバーされる範囲をファイルします。新しいアジェンダバッファの中で、このモードの初期設定は、org-agenda-start-with-clockreport-mode変数で設定することができます。このモードをトグル (すなわち *C-u R*) している時に、前置引数を使用することで、アジェンダフィルター[9] によって隠されているエントリーからの情報を表示しないでしょう。

v E or short *E*　org-agenda-entry-text-mode
entry text mode をトグルします。entry text mode では、アジェンダ行によって参照されている Org-mode のアウトラインのノードから、多数の行が、その行の下に表示されるでしょう。最大の行数は、org-agenda-entry-text-maxlines変数で指定します。数値付きの前置引数を付けて、このコマンドを呼び出すと、前置引数の値の数によって、即座に修正されます。

G　org-agenda-toggle-time-grid
時間のグリッドの表示をトグルします。org-agenda-use-time-gridと org-agenda-time-grid変数を参照してください。

r　org-agenda-rodo
アジェンダバッファを再構築する。例えば、*S-left*と *S-right*を使って、アイテムのタイムスタンプを改修したあと、その変更を反映するために。そのバッ

[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。

ファがグローバルな TODO リストの場合は、指定した TODO キーワードを選択できるリストを作成するために、前置引数を解釈します。

g org-agenda-rodo

カレントの Emacs のセッションにおいて、すべての Org-mode のバッファを保存します。あわせて ID の場所も。

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

アジェンダバッファの中でカラムビュー (7.5 節「Column view」p.64 を参照) 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。

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。(10.1 節「Agenda files」p.96 を参照).

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

[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。

ストの中で、工数見積りのインデックスの数字を入力します。そこでは *0*は１０番目の値を意味します。フィルターは選択された値よりも、以下、イコール、以上であるかによって限定されます。もしも 0-9 のキーがタグへのアクセスキーとして使用されていないならば、単純にあなたは操作コマンドを利用することなく、直接インデックスとなる数字を入力するだけです。この場合<が仮定されます。操作のアプリケーションのために、定義された工数がないエントリーでは、org-sort-agenda-noeffort-is-high変数の値に従って取り扱われます。工数の定義のないタスクにフィルターをかけるには、? を操作の値として入力します。

Org-mode はまた、コンテクストに対応したタグのフィルターを自動的にサポートしています。もしも、org-agenda-auto-exclude-function変数の値が、ユーザが定義した機能に設定されているときは、その機能によって、どのようなタグがアジェンダから自動的に排除されるかを決定します。一度この機能が設定されると、それによって、/コマンドは、*RET* をサブのオプションキーとして受け付け、自動的に排除ロジックを走らせます。例えば、いってみれば、ネットワークへのアクセスを必要とするタスクを定義するために Netというタグ、街での用事のために Errandというタグ、電話を掛けなければならないときに Callというタグを使用しているとします。あなたは、インターネッ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る。

[ ] { }

in *search view*

新しい検索の単語 (*[*と*]*)、あるいは新しい正規表現 (*{*と*}*) をクエリー文字列に追加する。開いた角括弧／大括弧は、‘+’という接頭辞のついたポジティブな検索用語を追加する。この検索用語は、必ずそのエントリーに発生／合致しなければならないことを示す。閉じた角括弧／大括弧は、ネガティブな検索用語を追加し、それは、選択されているエントリーの中で、絶対に発生／合致しないということである。

Remote editing

0-9　Digit argument.

C-_　org-agenda-undo
外部の編集コマンドでの変更を元に戻す。この変更はアジェンダバッファと外部のバッファの両方を元に戻す。

t　org-agenda-todo
アイテムの TODO のステータスを変更する。アジェンダファイルでもオリジナルの Org ファイルでも有効である。

C-S-right　org-agenda-todo-nextset
C-S-left　org-agenda-todo-previousset
次／前の TODO キーワードのセットへと切り替える。

C-k　org-agenda-kill
オリジナルの Org ファイルの中で、そのアイテムが属しているサブツリー全体と共に、カレントのアジェンダアイテムを削除する。もしも外部ファイルの削除するテキストが 1 行以上ならば、削除を行うには、ユーザーが指定する必要がある。org-agenda-confirm-kill変数を参照のこと。

C-c C-w　org-agenda-refile
その時点でそのエントリーを差し替える。

C-c C-x C-a or short *a*　org-agenda-archive-default-with-confirmation
org-archive-default-commandに設定されたデフォルトのアーカイブコマンドを使用して、その時点でエントリーに対応したサブツリーをアーカイブする。a キーを使用したときは、承認が必要である。

C-c C-x a　org-agenda-toggle-archive-tag
カレントの見出しのための ARCHIVE タグをトグルする。

C-c C-x A　org-agenda-archive-to-archive-sibling
カレントエントリーに対応したサブツリーを、アーカイブファイルに移動する。

C-c C-x C-s or short *$*　org-agenda-archive
カレントの見出しに対応したサブツリーをアーカイブする。これは、設定されたアーカイブの場所に、多くの場合それは異なるファイルであるが、エントリーを移動することを意味している。

T　org-agenda-show-tags
カレントアイテムと関連づけられたすべてのタグを表示する。もしも、あなたたが org-agenda-show-inherited-tags機能を停止しているにもかかわらず、依然として、たびたび見出しのすべてのタグを確認したいというときに役に立つ。

:　org-agenda-set-tags
カレントの見出しにタグを設定する。もしもアジェンダの中にアクティブなリージョンがあるときは、そのリージョンの中ですべての見出し用としてタグを変更する。

,　カレントアイテムに優先順位を設定する。(org-agenda-priority)　Org-mode は優先順位を表す文字を指示します。もしも、SPCを使って返答すると、優先順位のクッキーがそのエントリーから取り除かれる。

P org-agenda-show-priority
カレントアイテムの優先順位の重み付けを表示する。

+ or *S-up* org-agenda-priority-up
カレントアイテムの優先順位を高くする。優先順位はオリジナルのバッファで変更される。しかしアジェンダ上では並び替えの更新は行われない。このためには、*r* キーを使用する。

- or *S-down* org-agenda-priority-down
カレントアイテムの優先順位を低くする。

z or *C-c C-z* org-agenda-add-note
そのエントリーにのノートを追加する。このノートは記録され、ノートが置かれている状態を変更した同じ場所にファイルされる。org-log-into-drawerによって、これは引き出しの中に入る。

C-c C-a org-attach
すべてのコマンドの選択画面は、付属するものに関連づけられる。

C-c C-s org-agenda-schedule
このアイテムを予約する。前置引数をつけると、予約のタイムスタンプが削除される。

C-c C-d org-agenda-deadline
このアイテムにデッドラインを設定する。前置引数をつけるとデッドラインが削除される。

k org-agenda-action
カーソルの置かれた日付に選択されたアイテムの日付を設定するための、アジェンダのアクション。このコマンドはカレンダーでも動作する！コマンドは追加されたキーで入力する。

m　その地点でアクションのためにエントリーにマークする。複数のエントリーに対
しても可能である。
Org-mode では次を伴う *C-c C-x C-k*.

d　その時点の日付でマークされたエントリーのデッドラインを設定する。

s　その時点の日付でマークされたエントリーを予約する。

r　デフォルトの日付としてカーソルの日付とともに org-captureを呼び出す。

アジェンダを更新した後に、*r*を押すと、コマンドの効果を確認できる。

S-right org-agenda-do-date-later
カレント行に関連づけられたタイムスタンプを 1 日先に変更する。数値付きの前置引数をつけると、その数字の日数分だけ先に変更する。例えば、*3 6 5 S-right*と入力すると 1 年先に変更される。*C-u*という前置引数をつけると、1 時間ずつ時間を変更する。もしもあなたが、同じコマンドを即座に繰り返したいときは、前置変数を付けなくても 1 時間単位で変化し続けるでしょう。二重の *C-u C-u*という前置引数をつけると、同様に分単位で変更される。オリジナルの Org-mode ファイルの中でタイムスタンプは変更されるが、その変更はアジェンダバッファには直接は反映されない。バッファを更新するには、*r*または *g*を使用する。

S-left　org-agenda-do-date-earlier
カレント行のに関連づけられたタイムスタンプを 1 日過去に変更する。

>　org-agenda-date-prompt
カレント行に関連づけられたタイムスタンプを変更する。>キーが選択される。というのは、私のキーボード上では *S-.* と同じだからである。

I　org-agenda-clock-in
カレントアイテムの時計をスタートする。もしもすでに時計が動いているのならば、まずそれが停止する。

O　org-agenda-clock-out
すでにスタートした時計を停止する。

X　org-agenda-clock-cancel
カレントで動いている時計をキャンセルする。

J　org-agenda-clock-goto
別のウインドウの中の動いている時計にジャンプする。

Bulk remote editing selected entries

m　org-agenda-bulk-mark
大量のアクションについて、その時点でエントリーにマークをつける。前置引数を付けると、多くの連続したエントリーにマークをつける。

U　org-agenda-bulk-remove-all-marks
大量のアクションのマークを取り除く。

U　org-agenda-bulk-remove-all-marks
大量のアクションのためにマークがつけられたエントリーのマークを取り除く。

B　org-agenda-bulk-action
大量のアクション。アジェンダの中ですべてのマークをつかられたエントリーについて実行する。この機能では、適用されるアクションを選択するために、別のキーを入力する。*B*に前置引数をつけると、*s*や *d*のコマンドをパスして、これらの特別なタイムスタンプをまとめて取り除く。

r　1 つのリフィル上のターゲットに入力しすべてのエントリーを移動する。そのエン
トリーは
　アジェンダ上には表示されなくなる。再表示 (*g*) によって再度表示される。
$　選択されているエントリーをすべてアーカイブする。
A　エントリーをアーカイブし、それぞれを所定のアーカイブ先に移動する。
t　TODO の状態を変更する。これは TODO キーワード 1 文字を入力し、そして
　選択されたエントリーすべての状態を変更する。それはブロックしているのを無視
し
　ログのノートを抑え込んで（タイムスタンプは別です)。
+　選択されたエントリーのすべてにタグを付加する。

- 選択されたエントリーのすべてから、タグのひとつを削除する。
s すべてのアイテムに新しい日付で予約する。すでに予約がついていれば、日数分だ
け
日付を更新する。入力欄でプラスを 2 つつけて何かの数字を最初に打つことで。
例えば、‘++8d’とか ‘++2w’のように。
S N 日を指定して、それぞれをリスケジュールする。N は入力欄で指定する。前置引数

(*C-u B S*) をつけることで、平日のみに指定できる。
d 指定した日をデッドラインとして設定する。

Calendar commands

c org-agenda-goto-calendar

Emacs のカレンダーを開き、アジェンダのカーソルの置かれている日付に移動します。

c org-calendar-goto-agenda

すでにカレンダーの中にあるときは、カーソルの置かれている日付で計算し、Org-mode のアジェンダを表示します。

i org-agenda-diary-entry

カーソルの置かれている日付および（ブロックエントリーでは）マークされた日付を使って、新しいエントリーを日記に書き込みます。この機能では Emacs の日記ファイル[11] に追加することになります。ある意味では、カレンダーの *i* コマンドと似た機能です。日記ファイルは別のウインドウにポップアップし、そこでエントリーを書き加えることができます。

もしも Org-mode ファイルに `org-agenda-diary-file` を指定したならば、Org-mode ではそのファイルの中に（Org-mode の構文を使って）日記の代わりに、エントリーを作成することができます。ほとんどのエントリーは、日付を元にしたアウトラインのツリーの中に記述されており、あとで過去の月／年の中から予定をアーカイブするのを簡単にします。そのツリーは、`DATE_TREE`属性か、最上位のエントリーとして、年という属性を持ったエントリーのもとに構築されています。Emacs でエントリーのテキストを入力するようプロンプトが表示されるでしょう。もしもあなたがそれを指示するならば、さらなる連携なく、`org-agenda-diary-file`にそのエントリーを作成することになるでしょう。テキストを入力することなく、その入力欄で直接 `RET`を入力したら、そのターゲットとなるファイルがその場でのエントリーを終了させ、別のウインドウが表示されるでしょう。*k r* コマンドを参照してください。

M org-agenda-phases-of-moon

その日を中心として 3ヶ月間の月齢を表示する。

S org-agenda-sunrise-sunset

日の出と日の入りを表示する。地理上の場所によって、カレンダーの変数が設定される。Emacs の calendar の章を参照のこと。

[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。

C　org-agenda-convert-date
カーソルの置かれている日付によって、多くの他の文化的・歴史的なカレンダーに変換する。

H　org-agenda-holidays
カーソルのある日付を中心に 3ヶ月間の祝祭日を表示する。

M-x org-export-icalendar-combine-agenda-files
すべてのアジェンダファイルからエントリーを含んだ iCalendar 形式のファイルにエクスポートする。これはグローバルに利用できるコマンドで、そしてまたアジェンダメニューの中で利用できるコマンドです。

Exporting to a file

C-x C-w　org-write-agenda
アジェンダビューを 1 つのファイルに書き出します。選択したファイル名の拡張子に従って、そのビューは、HTML (拡張子が‘.html’または‘.htm’)、Postscript (拡張子‘.ps’)、PDF (拡張子‘.pdf’)、そしてプレーンテキスト (その他の拡張子) などにエクスポートされます。*C-u*という前置引数を用いてコマンドを呼び出したならば、即座に新しく作成されたファイルが開きます。エクスポートの間に使用されている‘ps-print’および‘htmlize’のためのオプションを設定するために、org-agenda-exporter-settings変数を使用します。

Quit and Exit

q　org-agenda-quit
アジェンダを終了し、アジェンダバッファを削除します s 。

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 (10.2 節「Agenda dispatcher」p.97 を参照) を通して利用できる。

10.6.1 Storing searches

カスタム検索の最初のアプリケーションは、よく使われる検索式のためのキーボードショートカットを定義することです。それはアジェンダバッファの作成、またはツリーの抽出（後者は言うまでもなくカレントバッファのみをカバーする）のどちらに対してでも。カスタムコマンドは、org-agenda-custom-commands変数で設定されます。あなたはこの変数をカスタマイズできます。例えば、*C-c a C*というように。またあなたは‘.emacs’に Emacs の Lisp を記述して直接設定することもできます。以下に述べる例はすべての適正な検索タイプを含んでいます。

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING")
        ("W" todo-tree "WAITING")
        ("u" tags "+boss-urgent")
        ("v" tags-todo "+boss-urgent")
        ("U" tags-tree "+boss-urgent")
        ("f" occur-tree "\\<FIXME\\>")
        ("h" . "HOME+Name tags searches") ; description for "h" prefix
        ("hl" tags "+home+Lisa")
        ("hp" tags "+home+Peter")
        ("hk" tags "+home+Kim")))
```

それぞれのエントリーの頭文字は、コマンドにアクセスするために、コマンド選択画面を呼び出す *C-c a*というコマンドの後に、入力しなければならないキーを定義します。通常、これは 1 文字をあてますが、もしもあなたが似たようなコマンドをたくさん持っていたら、あなたは 2 文字の組合せで定義することができます。その場合、いくつかの組合せでは最初の文字が同じものとなり、前置引数[12]. と同じように提供されます。2 番目のパラメーターは検索の種類を示し、マッチさせるために使われる文字列や正規表現がそれに続きます。上の例ではそれゆえ以下のように定義します。

C-c a w　TODO のキーワードとして、‘WAITING’となっている TODO エントリーのためのグローバルな検索として。す。

C-c a W　同じような検索であるが、カレントバッファにのみ適用され、ツリーの抽出として検索結果を表示する。

C-c a u　‘:urgent:’ではなく‘:boss:’というタグがつけられた見出しのための、グローバルなタグ検索を行う。

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。

[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。

```
(setq org-agenda-custom-commands
      '(("h" "Agenda and Home-related tasks"
         ((agenda "")
          (tags-todo "home")
          (tags "garden")))
        ("o" "Agenda and Office-related tasks"
         ((agenda "")
          (tags-todo "work")
          (tags "office")))))
```

これによって、家で精を出さなければならない用事に対するマルチブロックのビューを作成するために、*C-c a h*を定義します。アジェンダバッファには結果として、その週の、'home'というタグが含まれているすべての TODO アイテムと、'garden'というタグがついたすべての行のためのアジェンダを含むことになります。最後に、*C-c a o*というコマンドで、同様に、オフィスの作業についてのビューを得ることができます。

10.6.3 Setting options for custom commands

Org-mode はたくさんのアジェンダの構築や表示について調整する変数を含んでいます。グローバルな変数では、カスタムコマンドも含めて、アジェンダの全てのコマンドの動作を定義することができます。しかしながら、もしもあるひとつのカスタムビューについて、いくつかの設定を変更したいならば、それも可能です。オプションの設定は変数名のリストに書き込むことが必要で、org-agenda-custom-commandsの中に、正しい位置に値を書き込む必要があります。例えば。

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING"
         ((org-agenda-sorting-strategy '(priority-down))
          (org-agenda-prefix-format "  Mixed: ")))
        ("U" tags-tree "+boss-urgent"
         ((org-show-following-heading nil)
          (org-show-hierarchy-above nil)))
        ("N" search ""
         ((org-agenda-files '("~org/notes.org"))
          (org-agenda-text-search-extra-files nil)))))
```

こう書き込むことによって、*C-c a w*というコマンドは、優先順位によってのみ収集したエントリーを並べ替えるでしょう。そのエントリーのカテゴリを設定する変わりに、例えば' Mixed: 'という文字を prefix の形で書くことで変更することができます。*C-c a U*というタグでツリーを抽出するコマンドは、この結果、超コンパクトとなるでしょう。なぜならば、検索に合致した項目の上の階層の見出しも、合致した項目の見出しもどちらも表示されないからです。*C-c a N*というコマンドは、1 つのファイルに制限されたテキスト検索を実行します。

ブロックアジェンダを作成するコマンドセットのために、org-agenda-custom-commandsではオプションの設定用に 2 つの別の場所を用意しています。その設定の中にたったひとつのコマンドに有効なオプションを付け加えることも、その設定の中にすべてのコマンドに有効なオプションを付け加えることもできます。前者のオプションは 1 つのコマンドエントリーを付け加える。後者のオプションは、コマンドエントリーのリストを書き込むことが必要です。ブロックアジェンダの例に戻ると (10.6.2 節「Block agenda」p.117 を参照)、*C-c a h*というコマンドで、並べ替えの順序を優先順位の降順 priority-downに変更することができますし、その中で「GARDEN」というタグのついたものについては反対の順序、すなわち優先

順位の昇順priority-upに並べ替えることができるでしょう。このことは以下のように記述できます。

```
(setq org-agenda-custom-commands
      '(("h" "Agenda and Home-related tasks"
         ((agenda)
          (tags-todo "home")
          (tags "garden"
                ((org-agenda-sorting-strategy '(priority-up)))))
         ((org-agenda-sorting-strategy '(priority-down))))
        ("o" "Agenda and Office-related tasks"
         ((agenda)
          (tags-todo "work")
          (tags "office")))))
```

おわかりだと思いますが、変数とカッコで囲んでいる設定はやや複雑なところがあります。わかりにくいときは、カスタマイズのインターフェースとしてこの変数を設定してください。これはカスタマイズの構造を完全にサポートしています。注意しなければならないのは、このインターフェースでオプションを設定するときに、変数は、Lisp による表現をとっているということです。そのため、もしもその変数が 1 つの文字ならば、あなた自身でその変数の値に「”(ダブルクォート)」で囲む必要があるということです。

10.7 Exporting Agenda Views

もしもあなたが自分のコンピュータから離れているときは、いくつかのアジェンダのバージョンを印刷して持ち歩くことは大変役に立ちます。Org-mode はカスタムアジェンダビューをプレーンなテキスト、HTML[13]、Postscript、PDF[14]、iCalender ファイルとしてエキスポートすることができます。もしも、ときどきこのようなことを実行するのならばコマンドを使用しましょう。

C-x C-w org-write-agenda

アジェンダビューを 1 つのファイルに書き出します。選択したファイル名の拡張子により、そのビューは HTML（拡張子が ‘.html’または ‘.htm’）、Postscript（拡張子が ‘.ps’）、iCalendar（拡張子が ‘.ics’）、あるいはプレーンなテキスト（何かほかの拡張子）としてエクスポートされます。エクスポートの間に、‘ps-print’のため、および ‘htmlize’のためにオプションを設定するには、org-agenda-exporter-settings変数を使用します。例えば

```
(setq org-agenda-exporter-settings
      '((ps-number-of-columns 2)
        (ps-landscape-mode t)
        (org-agenda-add-entry-text-maxlines 5)
        (htmlize-output-type 'css)))
```

[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。

もしも、あなたがアジェンダビューをたびたびエクスポートする必要があるのならば、アウトプットのファイルの名前[15] のリストに、いくつかのカスタムなアジェンダのコマンドを関連づけることができます。ここに一つの例があります。最初のものはアジェンダとグローバルな TODO リストに対するカスタムなコマンドを定義しており、それらをエクスポートするたくさんのファイルと一緒になっています。それから 2 つのブロックアジェンダコマンドを定義し、同様にそれらのためのファイル名を指定しています。ファイル名は、現在作業しているディレクトリに対して相対パスにすることも絶対パスにすることもできます。

```
(setq org-agenda-custom-commands
      '(("X" agenda "" nil ("agenda.html" "agenda.ps"))
        ("Y" alltodo "" nil ("todo.html" "todo.txt" "todo.ps"))
        ("h" "Agenda and Home-related tasks"
         ((agenda "")
          (tags-todo "home")
          (tags "garden"))
         nil
         ("~/views/home.html"))
        ("o" "Agenda and Office-related tasks"
         ((agenda)
          (tags-todo "work")
          (tags "office"))
         nil
         ("~/views/office.ps" "~/calendars/office.ics"))))
```

ファイル名の拡張子がエクスポートのタイプを決定します。もしも拡張子が‘.html’ならば、Org-mode は‘htmlize.el’パッケージを使用し、バッファを HTML に変換し、そのファイル名で保存します。もしも拡張子が‘.ps’ならば、ps-print-buffer-with-facesが Postscript の出力をするために使用されます。もしも拡張子が‘.ics’ならば、iCalendar のエクスポートは、アジェンダを構成しているすべてのファイルにわたってエクスポートを実行し、現在アジェンダの中ではリスト化されたエントリーのエクスポートに限定されます。ほかの拡張子がついた場合は、プレーンな ASCII テキストファイルが作成されます。

エクスポートファイルは、非常に負荷が高いので、これらのコマンドの一つを相互に影響するように使用している時は、出力されません。そのかわり、1 ステップですべての指定されたファイルを出力する特別なコマンドが用意されています。

C-c a e org-store-agenda-views

アジェンダに関連するエクスポートファイル名を持つすべてのアジェンダビューをエクスポートします。

あなたは、エクスポートコマンドのためのオプションの設定をするために。カスタムアジェンダコマンドのオプションのセクションを使用することができます。例えば、

```
(setq org-agenda-custom-commands
      '(("X" agenda ""
         ((ps-number-of-columns 2)
          (ps-landscape-mode t)
          (org-agenda-prefix-format " [ ] ")
```

[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。

```
          (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。

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“char 65.8 節「Extracting agenda information」p.204, 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 (7.5 節「Column view」p.64 を参照) 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

[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。

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 (7.5.1.2 節「Column attributes」p.64 を参照) が定義されているならば、アジェンダでカラムビューに切り替えるときに、すべての関連するアジェンダファイルを確認して、この属性の計算の更新を確実に行います。このことは、特別な `CLOCKSUM`の属性が真であると設定されているということです。Org-mode はアジェンダの中で表示された値を合計するでしょう。一日／週間アジェンダの中で、合計は 1 日をカバーしています。他のビューでは、ブロック全体をカバーするのです。アジェンダでは同じエントリーを２度表示したり（例えばスケジュールと期限というように）、同じ階層（例えば親と子）から２つのエントリーを表示したりするかもしれない、ということを理解することは重要なことです。これらの場合、アジェンダの中での要約は、いくつかの値が二重にカウントされるという間違った結果を導く可能性があります。
3. アジェンダの中のカラムビューが、`CLOCKSUM`を表示するときは、このアイテムのためにいつでも時間計測全体に対応します。そのため 1 日/週間アジェンダにおいて、カラムビューでリスト化された時間合計は、カレントのビューの外側の時間から発生することになるかもしれません。この機能によって、あるタスクについて、計画された総工数を 1 つのカラムにリストにして、その値を比較することができるので、優位性を持ちます。この機能はアジェンダのカラムビューにおける重要なアプリケーションのひとつです。もしもあなたが表示されている期間の中の作業時間についての情報を得たいならば、clock table mode（`R`をアジェンダの中で入力する）を使用してください。

11 Markup for rich export

Org-mode の文書をエクスポートする時、エクスポート機能は文書の構造をできるだけ正確に反映しようとします。HTML や LaTeX、DocBook、その他のリッチフォーマット等のエクスポートの対象について、Org-mode は文書をリッチエクスポートに変換するルールを持ちます。このセクションは Org-mode 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

第 2 章「Document Structure」p.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<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 org-export-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

2.7 節「Plain lists」p.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。多くのバックケンドがサポートしているのは記号付きリスト、番号付きリスト、見出し付きリストです。

段落、改行、引用

段落は最低 1 つの空白行で区切られます。もしあなたが強制的に段落の中で改行しないなら、'\\'を行の末尾に書いてください。

リージョンで改行を保つためには、しかしそうでなければ通常のフォーマットが使われるなら、あなたはフォーマット技法として使われるこの構文を使うことができます。

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

     -- AlexSchroeder
#+END_VERSE
```

別の文書から一節を引用する時、段落の左右の余白を空けることが慣習となっています。あなたは以下を用いることで引用を Org-mode の文書に含めることができます:

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

もしあなたがテキストを中央寄せにしたいなら、以下を使うことができます:

```
#+BEGIN_CENTER
Everything should be made as simple as possible, \\
but not any simpler
#+END_CENTER
```

Footnote markup

脚注は 2.10 節「Footnotes」p.17 で説明されたように定義されていて、全てのバックエンドにエクスポートされます。Org-mode は同じノートに対しての複数の参照と異なるバックエンドをサポートします。

Emphasis and monospace

あなたは*bold*と/italic/、_underlined_、=code=、~verbatim~、そして必要なら‘+strile-through+’を単語に適用することができます。code と verbatim 文字列の中のテキストは Org-mode の明確な構文ではありません; それは verbatim にエクスポートされます。

Horizontal rules

少なくとも 5 文字のダッシュ文字のみで行成される線は水平線 (HTML では‘<hr/>’、LaTeX では\hrule) にエクスポートされます。

コメント行

行頭の文字が‘#’から始まる行はコメントとして扱われ、エクスポートされません。もしあなたがコメント行をインデントしたいのであれば、‘#+’から行を開始してください。‘COMMENT’ワードを持つサブツリーは、サブツリー全体がエクスポートされません。最後に、‘#+BEGIN_COMMENT’から‘END_COMMENT’で囲まれた範囲はエクスポートされません。

C-c ;　　エントリー先頭の COMMENT キーワードをトグルします。

11.2 画像と表

Org-mode ネイティブなテーブル (第 3 章「Tables」p.20 を参照) 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。

4.4 節「Handling links」p.38 を参照してください.

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 (2.7 節「Plain lists」p.13 を参照してください.) のためにインデントされるでしょう。小さな例を使う時、それを簡単にするために、あなたはコロンとそれに続く空白からなる例の行を使うことができる。それらはコロンの前に空白を追加することもできます。

```
ここに例を書きます
   : テキストファイルからの例
```

もし例がソースコードなら、もしくは Emacs でフォントロックによりマークアップされたテキストなら、あなたは Emacs バッファ [1] を要塞化するように要請することができます。あなたが例に色付けするために使うメジャーモードの名前を指定することが必要な時、‘src’ブロックを使います:

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

exampleと srcスニペットでは、あなたは BEGINの行の最後に-nを追加することで、例の行番号を表示することができます。もしあなたが+nとすると、前のスニペットから現在のものに番号が引き継がれます。リテラルの例で、Org-mode は‘(ref:name)’をラベルとして解釈し、[[(name)]]のような特別なリンクによりそこを参照することができます (i.e. 参照名は 1 つの括弧に囲まれています)。HTML では、対応するコード行をマウスオーバーすると自動的にハイライト表示になり、少しクールです。

また、ソースコード [2] からラベルを消去するかどうかの切り替えのために-rを追加することもできます。-nで切り替えると、リンクされるそれらのリファレンスはコードリスティングの行番号によってラベルを付けられ、そうでなければ括弧無しのラベルにリンクされます。

```
#+BEGIN_SRC emacs-lisp -n -r
(save-excursion                  (ref:sc)
   (goto-char (point-min))       (ref:jump)
#+END_SRC
In line [[(sc)]] we remember the current position.  [[(jump)][Line (jump)]]
jumps to point-min.
```

もし、ラベルの構文が言語の構文と衝突した場合、-lを使うことで‘#+BEGIN_SRC pascal -n -r -l "((%s))"’のようにフォーマットを変更できます。org-coderef-label-format変数を見てください。

---

[1] HTML バックエンドに対しては、この作業は自動的に行われます (Org-mode と一緒に配布されている‘htmlize.el’のバージョン 1.34 が必要です)。LaTeX の要塞化されたコードの塊はリスティングか、minted (http://coe.google.com/p/minted) パッケージによってアーカイブされます。リスティングを使うには、org-export-latex-listings変数をオンにし、LaTeX のヘッダにリスティングパッケージが含まれているようにします (例: org-export-latex-packages-alistの設定とを使います)。色付きの出力を含む設定のオプションについて、リスティングのドキュメントを見てください。minted を使うには、pygemnts (http://pygemnts.org) プログタムをインストールする必要があり、org-export-latex-mintedを追加で設定し、LaTeX のヘッダに mited パッケージが含まれていることと-shell-escapeオプションが‘pdflatex’に引き継がれている (org-latex-to-pdf-processを見てください) ことを確認します。

[2] Org-mode の例で説明するのに便利なリンクに行番号を使う間、-kを-n -rに追加することでソースコードのラベルを維持します。

HTML はエクスポート時にテキストエリア,12.5.7 節「Text areas in HTML export」p.138 を参照してください. 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 つ目のオプションは (e.g. '`quote`'や '`example`'、'`src`') のようなマークアップで、3 つ目はマークアップが '`src`' ならコンテンツの言語を表します。マークアップはオプションです; もし与えられなければ、Org-mode フォーマットのテキストと仮定される。インクルードの行は最初の行とそれに続く行のプレフィクスの指定のための追加キーワードパラメータの`:prefix1` と`:prefix`を、Org-mode のコンテンツを指定したレベル下げるための`:minilevel`を、同様に選択したマークアップ固有のオプションを持ちます。例えば、ファイルをインクルードするには:

```
#+INCLUDE: "~/snippets/xx" :prefix1 "   + " :prefix "     "
```

`:line`パラメータを使うことで、ファイルの指定した範囲の行のみをインクルードすることができます。範囲外の行はインクルードされません。範囲の開始と、または終了は明らかにデフォルトを使いません。

```
#+INCLUDE: "~/.emacs" :lines "5-10"    Include lines 5 to 10, 10 excluded
#+INCLUDE: "~/.emacs" :lines "-10"     Include lines 1 to 10, 10 excluded
#+INCLUDE: "~/.emacs" :lines "10-"     Include lines from 10 to EOF
```

C-c '　ポイント位置のインクルードされたファイルに移動します。

11.5 Index entries

あなたは公開した文書のインデックスに用いるエントリーを規定することができます。これは`#+INDEX`から始まる行により設定します。感嘆符を含むエントリーはサブアイテムを作るでしょう。さらなる情報を見るには 13.1.8 節「Generating an index」p.156 を参照してください。

---

[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。

[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。

```
* Curriculum Vitae
#+INDEX: CV
#+INDEX: Application!CV
```

11.6 Macro replacement

あなたは次のようにしてテキストスニペットを定義することができます。

```
#+MACRO: name   replacement text $1, $2 are arguments
```

これは`{{{name(arg1,arg2)}}}`と書くことでドキュメントのどこからも (コードの例からも) 参照されます。マクロの定義に加えて、`{{{title}}}`、`{{{author}}}`などは`#+TITLE:`や`#+AUTHOR:`や他の似たような行によりセットされる情報を参照します。また、`{{{date(`*FORMAT*`)}}}`と`{{{modification-time(`*FORMAT*`)}}}`は現在の日付とファイルがエクスポートされて変更された時刻をそれぞれ参照します。*FORMAT*は`format-time-string`で認識した文字列をフォーマットします。

マクロ展開はエクスポート中に行われ、一部の人々は複雑な HTML コードの構築に用いる。

11.7 Embedded LaTeX

プレーンな ASCII はほとんどの場合ノートをとるのに十分です。例外は数学の記号や時々出てくる数式を必要とする科学に関するノートのようなものです。LaTeX[5] は科学に関する文書の組版に広く使われています。多くの academics は LaTeX のソースコードの読み書きに使われていて、すぐに多くのエクスポートバックエンドに対応できるため、Org-mode は LaTeX コードのファイルへの組込みをサポートしています。

11.7.1 Special symbols

あなたは LaTeX マクロをギリシャ文字を表す‘`\alpha`’や矢印を表す‘`\to`’のような特殊記号の挿入に使うことができます。これらのマクロは補完が可能で、‘\’まで入力し、その後に何文字か入力して*M-TAB*を押すことで補完が可能です。LaTeX のコードとは違い、Org-mode は数学の区切り文字を囲まないようなマクロも使うことができます。以下に例を挙げます:

```
Angles are written as Geek letters \alpha, \beta and \gamma.
```

エクスポート時、これらのシンボルはエクスポート先のネイティブフォーマットに変換されます。HTML では`\alpha`のような文字列は`&alpha;`にエクスポートされ、LaTeX では`$\alpha$`となります。同様に、`\nbsp`は HTML では` `に、LaTeX では`~`となります。もしあなたが記号を単語の中に含めたいのであれば、次のようにします: ‘`\Aacute{}stor`’。

非常に多くのエンティティが提供されていて、HTML と LaTeX からその名前を引き継いでいます; 完全なリストは`org-entities`変数を見てください。‘`\-`’はシャイなハイフンとして扱われていて、‘`--`’や‘`---`’、‘`...`’は異なる長さのハイフンかドットの集合を作成するための全て特殊コマンドに変換されます。

もしあなたが UTF-8 文字でエンティティを表示したいのなら、以下のコマンド[6]:

*C-c C-x *　エンティティの UTF-8 での表示をトグルします。これはバッファの内容を変更せず、UTF-8 の文字を表示するためにオーバレイを用いています。

[5] LaTeX はドナルド・クヌースの TeX システムを基としたマクロシステムです。“LaTeX” で説明される多くの機能は TeX からのものですが、違いはそれほどありません

[6] あなたは`org-pretty-entities`変数または`#+STARTUP`オプション`entitiespretty`にデフォルトを設定することができます

11.7.2 Subscripts and superscripts

LaTeX と同じように、‘^’と‘_’が下付き文字と上付き文字を示しています。さらに、それらは math-mode にの区切り文字に組込まずに使うことができます。ASCII テキストの可読性の向上のため、複数文字の下付き文字と上付き文字を波括弧で囲む必要はありません (囲んでもかまいませんが)。例

```
The mass of the sun is M_sun = 1.989 x 10^30 kg.  The radius of
the sun is R_{sun} = 6.96 x 10^8 m.
```

上付きテキスト、下付きテキストの説明を避けるため、あなたはバックスラッシュをつけた‘^’と‘_’を引用できます: ‘\^’と‘_’です。異なる文脈でしばしば使われるアンダーラインのテキストを書くなら、常にこれらの下付き文字として解釈する Org-mode の慣習はあなたのやり方で得ることができます。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: ^:{}
```

この設定を有効にした場合、‘a_b’は下付き文字として解釈されず、‘a_{b}’とすることで解釈されます。

*C-c C-x *　さらに UTF-8 のエンティティを見るため、このコマンドは下付き文字と上付き文字を WYSISYM で形成する。

11.7.3 LaTeX の断片的なコード

シンボルと上付き、下付き、完全な式を越えることが必要です。Org-mode は LaTeX の数式を含むことができ、各エクスポート先への変換もサポートしています。LaTeX にエクスポートするとき、コードは明らかに残っています。HTML へエクスポートするとき、Org-mode は数式[7] の処理と描画のために MathJax library (`http://www.mathjax.org`) (12.5.6 節「Math formatting in HTML export」p.138 を参照) を呼び出します。最後に、数式表現はブラウザか DocBook 文書で描画可能な画像[8] へと処理されます。

LaTeX のコード片は、特別なマークは全く必要ありません。以下のコード片は LaTeX のソースコードとして知られています:

- あらゆる種類の環境[9]。唯一必要なことは`\begin`文は空白のみがある行に表示されることです。
- 通常の LaTeX の数学の区切り文字内部のテキスト。流通仕様との衝突を避けるために、囲まれたテキストに最大 2 つの改行が含まれている場合、‘$’文字は数学区切り文字のみとして認識され、‘$’文字がの間に空白がない、そして

例：

```
\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
```

[7]

[8] これを行うには、あなたのシステムに LaTeX をインストールする必要があります。そしてまた、`http://sourceforge.net/projects/dvipng/`で入手できる‘dvipng’プログラムも必要です。

[9] ‘MathJax’が使われている時、‘MathJax’によって認識されている環境が処理されます。‘dvipng’を画像の生成に用いる時、LaTeX 環境が扱われます。

```
either $$ a=+\sqrt{2} $$ or \[ a=-\sqrt{2} \].
```

もしあなたが他の目的に ASCII の区切り文字が必要なら、LaTeX コンバータに邪魔されえることを望まない文字を除外するために `org-format-latex-options`オプションを設定することができます。

LaTeX の処理は `org-export-with-LaTeX-fragments`変数を設定することができます。デフォルトの設定は `t`で、HTML には‘`MathJax`’を用い、DocBook と ASCII、LaTeX では処理しません。あなたはこの変数をファイルの冒頭部分に書くことで設定することもできます:

`#+OPTIONS: LaTeX:t`	Do the right thing automatically (MathJax)
`#+OPTIONS: LaTeX:dvipng`	Force using dvipng images
`#+OPTIONS: LaTeX:nil`	Do not process LaTeX fragments at all
`#+OPTIONS: LaTeX:verbatim`	Verbatim export, for jsMath or so

11.7.4 Previewing LaTeX fragments

もしあなたが‘`dvipng`’をインストールしているのであれば、LaTeX のコード片は出力された組版において画像として処理されます:

C-c C-x C-l
: ポイント位置の LaTeX コード片の画像プレビューの提供とソースコード上のオーバレイ。もしポイント位置にコード片がないのであれば、現在のエントリ (2 つの見出しの間) の全てのコード片を処理します。前置引数を付けて呼ばれた時は、サブツリー全体を処理します。前置引数を 2 つ付けて呼ばれた時、またはカーソルが最初の見出しの前にある時は、バッファ全体を処理します。

C-c C-c　オーバレイされたプレビュー画像を消去します。

プレビューの外観を変更するために、あなたは `org-format-latex-optins`変数をカスタマイズすることができます。とりわけ、`:scale`(そして HTML へのエクスポートでは`:html-scale`) プロパティは画像のプレビューサイズの調整に使われます。

11.7.5 CDLaTeX を数学の入力に使う

CDLaTeX モードは環境や数学テンプレートの挿入をスピードアップするために AUCTeX に似たメジャーモードである LaTeX モードと併用して通常使われるマイナーモードです。Org-mode では、あなたは CDLaTeX モードのいくつかの機能を使用できます。あなたは `http://www.astro.uva.nl/~dominik/Tools/cdlatex`から‘`cdlatex.el`’と‘`texmathp.el`’(最近 AUCTeX に追加されました) をインストールする必要があります。Org-mode 中では CDLaTeX モード自身は使わないでください、代わりに Org-mode に一部である、より軽量なバージョンの `org-cdlatex-mode`を使ってください。`M-x org-cdlatex-mode`をカレントバッファで実行して有効にするか、全ての Org-mode ファイルで有効するために次の設定を行います:

```
(add-hook 'org-mode-hook 'turn-on-org-cdlatex)
```

このモードが有効である時、以下の機能が提供されます (詳細は CDLaTeX モードのドキュメントを参照してください)::

- *C-c {*による環境テンプレートの挿入。
- カーソルが LaTeX のコード片[10] の中にある場合、`TAB`キーはテンプレートの展開を行います。例えば、`TAB`は `fr`を`\frac{}{}`に展開しカーソルを最初の括弧に移動します。も

[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。

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)。
- LaTeX のコード片以外の文字に続いて *TAB*を押すと、数学のマクロが挿入されます。もしあなたがバッククォートを押して 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。
- 別の文字に続いてシングルクォート ’を押すと、強調やフォントでポイント前のシンボルが変更されます。もしシングルクォートを入力した後 1.5 秒以上待つと、ヘルプウィンドウがポップアップします。文字の変更は LaTeX コード片の中でのみ働きます; それ以外ではクォートは通常通りの働きをします。

12 Exporting

org-mode のドキュメントは様々なフォーマットにエクスポートすることができます。ノートを共有し印刷するには ASCII 形式でエクスポートすることで Org ファイルの読みやすく、シンプルなものが得られます。HTML のエクスポートではノートをウェブに公開できるようになりますし, XOXO フォーマットは他の様々なアプリケーションでやりとりするうえで確かな基礎となります。 LaTeX のエクスポートでは、org-mode とその構造化された編集機能を使って、容易に LaTeX のファイルを出力することができます。DocBook のエクスポートでは、Org ファイルを DocBook のツールを使った様々なフォーマットに変換することが可能です。プロジェクトの管理では、TaskJuggler 形式のエクスポートを使って、ガントリソースチャートを作成することができます。デッドラインや予約のような時間と関連のあるエントリーを iCal のようなデスクトップカレンダーに取り込むために org-mode は iCalendar 形式で抽出することもできます。現在、Org-mode はエクスポートのみをサポートしており、他の異なるフォーマットからインポートすることはできません。

org-mode は、 `transient-mark-mode` がオンの時 (Emacs 23 ではデフォルト)、は選択したリージョンをエクスポートをすることができます。

12.1 Selective export

エクスポートしたいドキュメントのある部分を選択、または除外する時にタグを使うことができます。 その挙動は、`org-export-select-tags` と `org-export-exclude-tags` の二つの変数により決まります。

org-mode はまず最初に *select* タグがバッファにないかチェックします。 あった場合は, タグがない全てのツリーは除外されます。もし選択したツリーがサブツリーだった場合、それより上の階層はエクスポートされるものとして選択されますが, それより下の階層は選択されません。

もし、選択されたタグがなかった場合, バッファにある全ての内容がエクスポートされるものとして選択されるでしょう。

最後に, *exclude* タグでマークされていない全てのサブツリーはエクスポートバッファから除かれるでしょう。

12.2 Export options

エクスポートする際にはバッファにある特別な行が読みこまれます。 その行には追加的な情報が含まれており、ファイルの中でどこにでも書くことができます。*C-c C-e t* と入力することで、バッファにそのような行をセットで挿入することができます。それぞれの行で '`#+`' と入力した後に *M-TAB* による補完を行ない、 (15.1 節「Completion」p.182 を参照) 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 15.6 節「In-buffer settings」p.184 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

[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。

`d:`　　drawer を出力するかどうか

これらのオプションは HTML、 LaTeX の両方のエクスポートに影響します。 `TeX` と `LaTeX` のオプションを除き LaTeX のエクスポートをするのに、それぞれ `t` 、または `nil` となります。

`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。(13.1.1 節「Project alist」p.152 を参照), `org-export-plist-vars` の定数を見てください。

エクスポートのコマンドを呼びだす前に、 *C-c @* で選択した単一のサブツリーをエクスポートする時、そのサブツリーは、`EXPORT_FILE_NAME` 、 `EXPORT_TITLE` 、 `EXPORT_TEXT` 、 `EXPORT_AUTHOR` 、 `EXPORT_DATE` 、 そして `EXPORT_OPTIONS` プロパティでエクスポートの設定を無視することができます。

12.3 The export dispatcher

全てのエクスポートコマンドはエクスポートコマンド選択画面から選ぶことができます。コマンド選択画面では、コマンドを特定するための追加的なキーの入力を促されます。通常、ファイルの全ての内容がエクスポートされますが、もしアクティブなリージョンに一つのアウトラインツリーが含まれていた場合, まず、見出しがドキュメントのタイトルとして扱われ、サブツリーがエクスポートされます。

C-c C-e　　`org-export`

エクスポート、または公開のコマンド選択画面です。エクスポート、または公開のコマンドを起動するのに必要なキーがヘルプウィンドウに表示されます。前置引数として、入力すると、直接エクスポート機能となります。二重の前置引数 *C-u C-u* を入力することで、コマンドは別の Emacs プロセスにおいてバックグラウンドで実行されます。[2].

C-c C-e v　　`org-export-visible`

C-c C-e のように動作しますが、今見えている文章だけがエクスポートされます。(i.e. アウトライン表示により、隠されていない文章).

C-u C-u C-c C-e　　`org-export`

エクスポート機能が呼ばれますが、 `org-export-run-in-background` の設定と逆の挙動となります。 i.e. 動いていないバックグラウンドプロセスを呼びだしたり, 現在の Emacs のプロセスで強制的に実行したりします。

12.4 ASCII/Latin-1/UTF-8 export

ASCII 形式へのエクスポートは、org-mode のファイルを ASCII のみが含まれる、シンプルで読みやすい形に書き出します。Latin-1 及び UTF-8 でのエクスポートでは特殊な文字やシンボルをそれらのエンコードで出力します。

C-c C-e a　　`org-export-as-ascii`

ASCII 形式のファイルをエクスポートします。 Org ファイルを ‘`myfile.org`’ だとすると、ASCII 形式のファイルは ‘`myfile.txt`’ となるでしょう。 そのファイ

[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.

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、そのツリーの見出しがドキュメントのタイトルとなるでしょう. 見出しがあるか、または EXPORT_FILE_NAME プロパティを継承していた場合、 エクスポートする際にはその名前が使われるでしょう。

C-c C-e A　org-export-as-ascii-to-buffer
一時的なバッファに出力し、ファイルを作成しません。

C-c C-e n　org-export-as-latin1
C-c C-e N　org-export-as-latin1-to-buffer
上に示したコマンドのような動作をしますが, Latin-1 でエンコーディングされたものが出力されます。

C-c C-e u　org-export-as-utf8
C-c C-e U　org-export-as-utf8-to-buffer
上に示したコマンドのような動作をしますが, UTF-8 でエンコーディングされたものが出力されます。

C-c C-e v a/n/u
文書の中で、バッファで表示されている部分だけを出力する。

エクスポートされたものでは、 最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層はアイテムのリストとしてエクスポートされます。 この違いを異なる階層に変えたい場合は、 前置引数で、その階層を指定します。例えば、

```
C-1 C-c C-e a
```

は 最初の階層のみを見出しとし、それ以外はアイテムとなります。見出しがアイテムに変更された時、見出し後の文章のインデントは、アイテムの下にうまく調和するように変更されます。この変更は、最初の本文が全体のインデントを示しているという仮定のもとで実行されます。これよりも大きなインデントは、最初の文章との相対的なレイアウトを維持するように調整されます。最初の行より少ないインデントであれば, 左寄せします。

次の見出しの前にあるリンクは脚注のような形でエクスポートされます。その脚注は、次の見出しの前に項目名とリンクがエクスポートされます。詳しい内容と他のオプションについては、変数 org-export-ascii-links-to-notes を見てください。

12.5 HTML export

org-mode には多くの HTML のフォーマットに対応した HTML (XHTML 1.0 準拠) エクスポート機能があります。それは、John Gruber が開発した *markdown* 言語に似ていますが、org-mode ではさらにテーブルもサポートしています。

12.5.1 HTML エクスポートのコマンド

C-c C-e h　org-export-as-html
HTML ファイル ‘myfile.html’をエクスポートします。Org ファイル ‘myfile.org’ をエクスポートすると、ASCII 形式のファイルは ‘myfile.html’

[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。

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。

---

[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。

12.5.3 Links in HTML export

内部リンク (4.2 節「Internal links」p.36 を参照) 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 (4.2.1 節「Radio targets」p.37 を参照) 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、13.1.6 節「Publishing links」p.155 参照してください。

リンクの属性を記述したい時は、特別な#+ATTR_HTML行を用いることができます。この行は、<a>タグや<img>タグを追加する属性を定義するために使われます。以下の例では、リンクに titleと styleの属性を設定しています。

```
#+ATTR_HTML: title="The Org-mode homepage" style="color:red;"
[[http://orgmode.org]]
```

12.5.4 Tables

org-mode の表は、org-export-html-table-tagで定義されているテーブルのタグを使って HTML にエクスポートされます。デフォルトの設定では、セルの罫線とフレームがない状態でテーブルが出力されます。 個々のテーブルでその設定を変えたい場合は, 次のような行をテーブルの前に記述してください。

```
#+CAPTION: これはセルの周囲に線が引かれた表です。
#+ATTR_HTML: border="2" rules="all" frame="all"
```

12.5.5 Images in HTML export

HTML のエクスポートでは Org ファイルにリンクがある画像をインライン表示することができます。その画像はリンクされているクリック可能な部分として扱われます。デフォルトでは、[7], リンクに description がなければ、画像はインライン表示されます。 つまり、‘[[file:myimg.jpg]]’はインライン表示されますが、‘[[file:myimg.jpg][the image]]’はが画像にリンクされる‘the image’というテキストリンクが作られます。description の部分が file:リンクか画像を示す http:の URL の場合, 画像はインラインに表示され、画像がクリックされると活性化されます。例えば、リンク先に高解像度の画像があるサムネイルを追加したい場合、次のように書くと良いでしょう。

```
[[file:highres.jpg][file:thumb.jpg]]
```

インライン画像に属性を追加したい場合は、#+ATTR_HTMLを使います。次の例では、テキストでの見やすさとアクセスのしやすさを考慮して altの属性と titleの属性を指定して、align を右にしています。

```
#+CAPTION: A black cat stalking a spider
#+ATTR_HTML: alt="cat/spider image" title="Action!" align="right"
[[./img/a.jpg]]
```

httpのアドレスも使うことができます。

[7] ただし、org-export-html-inline-imagesを確認してください。

12.5.6 Math formatting in HTML export

LaTeX の数学用スニペット (11.7.3 節「LaTeX fragments」p.129 を参照) 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"
```

See the docstring of the variable `org-export-html-mathjax-options` for the meaning of the parameters in this line.

望むのであれば、LaTeX を小さな画像に変換してブラウザ上のページに挿入することもできます。MathJax が有用である前には、これが org-mode でのデフォルトの方法でした。この方法を用いるには、あなたのシステムで'`dvipng`'プログラムが利用できる状態である必要があります。この方法は以下のような行を追加することでも有効になります。

```
#+OPTIONS: LaTeX:dvipng
```

12.5.7 Text areas in HTML export

コードサンプルを HTML にして公開する方法として、テキストエリアを使う方法があります。何かのアプリケーションに貼りつける前であれば、そのコードサンプルは編集することができます。`example`ブロックか`src`ブロックに`-t`スイッチが付加されることでテキストエリアに変換されます。このスイッチを使うことで、シンタックス、ラベルのハイライト、行番号に関するオプションが無効になります。`-h`と`-w`を使うことがあるかもしれません。それらのスイッチはテキストエリアの高さと幅を特定するもので、デフォルトでは高さが example ブロックの行数で幅は 80 となります。設定は、例えば以下のようになります。

```
#+BEGIN_EXAMPLE -t -w 40
  (defun org-xor (a b)
     "Exclusive or."
     (if a (not b) b))
#+END_EXAMPLE
```

12.5.8 CSS support

エクスポートするファイルには、スタイルに関する情報を含めることができます。HTML エクスポート機能には、文章のパーツを適切に表示するために次に示す特別な CSS クラス[9] があります。見出しやテーブルなどの標準的なクラスに加えて、それら特別な CSS クラスも変更することができます。

`p.author`	著者の情報、email 含む
`p.date`	公開日
`p.creator`	作成情報, org-mode のバージョン
`.title`	文章のタイトル

[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/?installation` を参照してください。

[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。

.todo	DONE となっていない TODO キーワード
.done	DONE キーワード、DONE と扱われる全てのキーワードが対象
.WAITING	各 TODO キーワードはその名前のクラス名も用いることができる
.timestamp	タイムスタンプ
.timestamp-kwd	SCHEDULED 等のタイムスタンプに関連するキーワード
.timestamp-wrapper	SCHEDULED 等のキーワードとタイムスタンプ全体
.tag	見出し中のタグ
._HOME	各タグはその名前のクラス名も用いることができる ("@"は"_" に置き換えられる)
.target	リンクのターゲット
.linenr	コード中の行番号
.code-highlighted	参照されコード行のハイライト
div.outline-N	深さレベル N の div 要素 (見出しとテキスト)
div.outline-text-N	深さレベル N のテキスト部分の div 要素
.section-number-N	深さレベル N の見出しの番号。各レベルで異なる
div.figure	インライン画像のフォーマット方法
pre.src	ソースコードブロックのフォーマット方法
pre.example	例示ブロック
p.verse	verse ブロック
div.footnotes	脚注の見出し
p.footnote	脚注定義の文章、脚注を含む
.footref	脚注の参照番号 (常に<sup>となる)
.footnum	脚注定義中の番号 (常に<sup>となる)

エクスポートされたファイルは、基礎的な方法で定義されたコンパクトなスタイル[10] が含まれています。この設定は上書きされるかもしれませんし、org-export-html-style (Org-wide の設定に使われます) や org-export-html-style-extra (ファイルごとの設定のような詳細な設定に使われます。) を使って追加されるかもしれません。後者の変数をファイルごとに設定するには、次のように行ないます。

```
#+STYLE: <link rel="stylesheet" type="text/css" href="stylesheet.css" />
```

長いスタイルの定義には複数行で記述することもできます。外部ファイルを参照せずに<style></style>セクションに直接記述してください。

サブツリーにスタイルを追加するには, ツリーにクラスを適用する:HTML_CONTAINER_CLASS: プロパティを使います。個々の見出しに CSS スタイルを適用するには、:CUSTOM_ID: プロパティで指定される ID を使うことができます。

12.5.9 ウェブページの表示に関する JavaScript のサポート

Sebastian Rose は、org-mode が生成した HTML ファイルに関するウェブエクスペリエンスを拡張するためにデザインされた Javascript プログラムを書きました。このプログラムを使うことで、異なる二つの方法で大きなファイルを見ることができます。一つめは *Info* のようなモードで、それぞれの章は別々に表示され、*n*キーと *p*キーで操作できます。(他のキーでも操作できます。利用できるキーの概要を知るには、?を入力してください。)。二つめは、org-mode が Emacs で提供するような折りたたまれたスタイルです。このスクリプトは、http://

[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。

orgmode.org/org-info.jsで利用できます。ドキュメントについては、http://orgmode.org/worg/code/org-info-js/にあります。このスクリプトは私達のサイトでホスティングしていますが、何度も使う場合は、orgmode.orgにあるものを使わずにあなたのサーバにコピーしたものを使う方を選択するかもしれません。

このスクリプトを使うには、‘org-jsinfo.el’がロードされているか、確認する必要があります。デフォルトでは、ロードされるようになっていますが、*M-x customize-variable RET org-modules RET*と入力して、確かにロードされている確認してください。このプログラムを使えるようにするには、次のような行を Org ファイルに追加するだけです。

```
#+INFOJS_OPT: view:info toc:nil
```

ファイル中にこの行が見つかると、HTML のヘッダーは自動的にこのスクリプトを起動させるのに必要なコードを自動的に追加します。以上のような行を使うと、次のようなオプションを設定できます。

path:　スクリプトのパス。デフォルトでは、http://orgmode.org/org-info.jsを使うようになっていますが、ローカルにコピーしたものを使いたい場合は

‘../scripts/org-info.js’のようなパスを使ってください。

view:　ウェブサイトを最初に開いた時の表示。可能な値は次のとおり:

info　一つのページに一つのセクションが表示される Info のようなインターフェイス

overview　最初はトップレベルのみが表示される折りたたみインターフェイス

content　全ての見出しが見える状態の折りたたみインターフェイス

showall　全ての見出しと文章が見える状態の折りたたみインターフェイス

sdepth:　info や折りたたみモードで独立して表示されるセクションの最大の見出しレベル。デフォルトでは org-export-headline-levels (= #+OPTIONSの中の Hスイッチ) の値が使われる。もし、org-export-headline-levelsの値より小さかった場合、info/折りたたみ のセクションは小見出しまで含まれます。

toc:　目次表示の有無

nilとしても、*i*を入力することで目次は表示されます。

tdepth:　目次の深さ。デフォルトでは、org-export-headline-levels org-export-with-tocの値が用いられます。

ftoc:　CSS によって、目次の場所を指定するかどうか。「yes」の場合は、セクションとして表示されなくなります。

ltoc:　それぞれのセクションにショートコンテンツを設置するかどうか。セクションの冒頭にショートコンテンツを設置する場合は値を aboveとします。

mouse:　マウスを見出しの上に移動させた時にハイライトさせます。‘underline’ (default) か、‘#cccccc’のように背景色が指定できます。

buttons:　ビューの変更をトグルさせるボタンを様々なところに設置するかどうか。nilの場合は、(デフォルト)、ボタンが一つだけ表示されます。

org-infojs-optionsを変更することで、これらのオプションの初期値を変更することができます。このスクリプトを常にページに適用させたい場合は、org-export-html-use-infojsを変更してください。

12.6 LaTeX と PDF のエクスポート

org-mode には、Bastien Guerry によって書かれた LaTeX のエクスポート機能があります。追加的な処理と合わせて、[11], このバックエンドは PDF の出力にも使われています。LaTeX の出力は、リンクと相互参照の実装に 'hyperref' を使っているので、出力された PDF ファイルは完全にリンクされているでしょう。セクションの階層に合わせて正しく出力されるためには、org ファイルは適切に構造化されていないといけないので注意してください。

12.6.1 LaTeX エクスポートのコマンド

C-c C-e l　org-export-as-latex
: LaTeX ファイル 'myfile.tex'を出力します。Org ファイルに対して 'myfile.org' 、ASCII ファイルは 'myfile.tex'となるでしょう。そのファイルは警告なしに上書きされます。アクティブなリージョン[12] があれば、そのリージョンのみが出力されるでしょう。選択したリージョンが一つのツリー[13] であった場合、ツリーの見出しがタイトルになります。ツリーの見出しのエントリーが EXPORT_FILE_NAME プロパティを継承、または持っている場合、エクスポートされる際には、その名前が使われるでしょう。

C-c C-e L　org-export-as-latex-to-buffer
: 一時バッファに出力します。ファイルを作りません。

C-c C-e v l/L
: 文書の中で、バッファで表示されている部分だけを出力する。

M-x org-export-region-as-latex
: Org-mode の記法が使われているという前提でリージョンを LaTeX に変換します。これはどのバッファでも起動するグローバルなコマンドです。

M-x org-replace-region-by-latex
: アクティブなリージョンを (Org-mode の記法が使われている前提で)LaTeX コードに置き変えます。

C-c C-e p　org-export-as-pdf
: LaTeX に出力し、PDF にも変換します。

C-c C-e d　org-export-as-pdf-and-open
: LaTeX に出力し、PDF にも変換します。その際出力された PDF ファイルを開きます。

エクスポートされたものでは、 最初の 3 つのアウトラインの階層が一般的な文書の構造と見なされて、見出しとなります。それ以外の階層は概要のリストとしてエクスポートされ

[11] デフォルトの LaTeX 出力は、pdftex または latex により出力されるよう設計されています。それには、xetex や恐らく luatex と互換性のないパッケージが含まれています。org-export-latex-default-packages-alist や org-export-latex-packages-alistを参照してください。

[12] 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。

ます。エクスポート機能では、org-latex-low-levelsを変更することで、この設定を無視、または変更することができます。

この違いを異なる階層で変えたい場合は、 前置引数で、その階層を指定します。 例えば、

C-2 C-c C-e l

この場合２番目のレベルまでを見出しとして取り扱い、それ以外は項目として取り扱います。

12.6.2 見出しと構造の分割

デフォルトでは、LaTeX の出力には articleクラスが使われます。

クラスは org-export-latex-default-classの値を変更することで、全体的に変更することもできますし、ファイル中に org-export-latex-default-classのようなオプションを追加することで、局所的に変更することもできます。:LaTeX_CLASS:プロパティを使えば、エクスポートするリージョンにそのツリー (サブツリー) のみが含まれていた場合にクラスを指定できます。クラスは、org-export-latex-classesにリストアップされてます。この変数は、各クラス[14] の見出しテンプレートを定義し、各クラスの構造の分割について定義します。クラス自体についても定義されます。#+LaTeX_CLASS_OPTIONS、または LaTeX_CLASS_OPTIONS プロパティは\documentclassマクロのオプションを指定します。見出しに#+LATEX_HEADER: \usepackage{xyz}を追加して同様のことをすることもできます。詳しい情報については、org-export-latex-classesのドキュメント文字列を参照してください。

12.6.3 LaTeX コードの引用

11.7 節「Embedded LaTeX」p.128 で記述された埋め込まれた LaTeX は、LaTeX に正しく挿入されます。図の相互参照を生成するために、‘\ref{LABEL}’のようなシンプルなマクロが含まれます。さらに、次のような行を追加することで、LaTeX エクスポートの際に表示だけしてほしい特別なコードを追加することができます。

```
#+LaTeX: エクスポートする際に文字のまま、出力される LaTeX code
```

or

```
#+BEGIN_LaTeX
マーカの間にある全ての行は文字がそのまま出力されます。
#+END_LaTeX
```

12.6.4 LaTeX エクスポートにおけるテーブル

LaTeX で表を出力する際に、番号と表題を指定することができます (11.2 節「Images and tables」p.125 を参照)、ATTR_LaTeX行を使うことで、表に関する longtable環境を呼び出すこともできます。複数のページにまたがる表や、デフォルトの表の環境を tableから table*にするため、またはデフォルトの内部 tabular 環境を tabularxや tabularyにしたい時にも ATTR_LaTeX行は使われます。つまり、文字の配置や (tabularxや tabularyを使って) 幅を次のようにして設定できます。:

```
#+CAPTION: A long table
#+LABEL: tbl:long
#+ATTR_LaTeX: longtable align=l|lp{3cm}r|l
| ..... | ..... |
| ..... | ..... |
```

[14] org-export-latex-default-packages-alistと org-export-latex-packages-alistが接合されたものです。

tabularyを使って、複数のセルにまたがる表を指定することもできます。

```
#+CAPTION: A wide table with tabulary
#+LABEL: tbl:wide
#+ATTR_LaTeX: table* tabulary width=\textwidth
| ..... | ..... |
| ..... | ..... |
```

12.6.5 LaTeX エクスポートにおける画像

‘[[file:img.jpg]]’ や ‘[[./img.jpg]]’ のように説明文にリンクされていない画像は、LaTeX の処理により PDF の中に挿入されます。 Org-mode は、画像を挿入するのに \includegraphics マクロを使います。もし、11.2 節「Images and tables」p.125 で説明されているように図の表題や番号を特定したいのならば、その図を figure 環境で囲むと float 要素になります。 \includegraphics マクロのオプション引数として使われている様々なオプションを特定するには#+ATTR_LaTeX: を使います。figure 環境のオプションの配置を変更するには、‘placement=[h!]’ のように属性に追加します。

画像のまわりに文字を回りこませたいのであれば、#+ATTR_LaTeX: の行に ‘wrap’ を追加すると、画像がページの左半分にきます。微調整するには、wrapfigure 環境に引数として、 placement フィールドを追加します。画像のサイズを変更する時は、互換性のある \includegraphics と wrapfigure を使わなければいけないので注意してください。

```
#+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]]
```

もし、このような番号を参照する必要があれば, LaTeX に ‘\ref{fig:SED-HR4049}’ と記述してください。

12.6.6 Beamer クラスのエクスポート

LaTeX の一種である ‘beamer’ は、LaTeX と pdf 処理により高品質なプレゼンテーション資料を提供します。Org-mode は Org-mode のファイルやツリーを ‘beamer’ のプレゼンテーション資料に変換するのに特別なサポートをします。

カレントバッファ (#+LaTeX_CLASS: beamer がセットされている) かサブツリー (LaTeX_CLASS 属性がセットされている) の LaTeX クラスが beamer ならば、特別なエクスポートモードがファイルやツリーを beamer のプレゼンテーション資料にします。原則的にはあまり深くないネストのツリーなら何でもプレゼンテーション資料にします。デフォルトでは、トップレベルのエントリー (または、選択したサブツリーの最初のレベル) がフレームに変換され、そのレベルの下のアウトライン構造が箇条書きされたリストになります。 変数 org-beamer-frame-level は異なるレベルに設定でき、その時フレームより上の構造はプレゼンテーションの構造の区切りになります。

バッファでの便利なテンプレートに関する設定や属性は *M-x org-insert-beamer-options-template* によってバッファに挿入されます。その他については、カラムビューのフォーマットにインストールされます。カラムビューは beamer で使う特別な属性を編集するのに便利だからです。

次のような属性を使ってプレゼンテーションの構造を変えることができます。:

BEAMER_env
: このエントリーをフォーマットする環境を指定します。有効な環境が変数 org-beamer-environments-default に定義されていて、さらに org-beamer-environments-extra に追加して定義することができます。もし、この属性がセットされていれば、それを可視化するため、そのエントリーには :B_environment: タグがあるはずです。このタグは字句的な意味はなく、視覚的に補助するためにあります。

BEAMER_envargs
: [t] 、 [<+->] や <2-3> のような beamer に特有な引数は、この環境で使われます。もし BEAMER_col 属性がセットされていると、 columns 環境のオプション引数として、 C[t] が追加されたことを意味します。 c[t] や c<2-> は column 環境にオプションとして設定されたことを意味します。

BEAMER_col
: この列が始まるときの列の幅です。もしこの属性がセットされていると、そのエントリーには :BMCOL: 属性が現れます。このタグも視覚的な補助のためにあります。もし、これが整数だった場合、 \textwidth の割合とみなされます。もしくは、'3cm' のような場合、特定の単位を使ったとみなされます。まず、フレームの中のそのような属性は列に囲まれた columns 環境で始まります。 BEAMER_col 属性が 0 か 1、または自動的にフレームの最後となるエントリーでは、その環境は閉じられます。

BEAMER_extra
: その環境が始まった後に挿入される追加的なコマンドです。例えば、フレームを作った時に変遷を特定します。

もし、verbatim 環境を使ったソースコードが含まれていると、フレームは自動的に fragile を受けとります。'beamer' 特有のコードは #+BEAMER: を使うことで挿入され、#+BEGIN_beamer...#+end_beamer ブロックは, 他のエクスポートのものと似ていますが、#+LaTeX: はプレゼンテーション資料にも含まれるという点で異なります。

BEAMER_env 属性があるノードの 'note' や 'noteNH' の値は beamer の notes として処理されます。例えば、 \note{...} のように囲まれます。前者は、ノートのテキストの一部分として見出しが含まれ、後者は、ノードの見出しは無視されます。ノートの生成を簡単にするには、実は BEAMER_env 属性を作るかわりに、 *tag* (または :B_note: や:B_noteNH:) でマークするだけで十分です。

編集作業のサポートを得るには次のオプションを追加して、マイナーモードの org-beamer-mode を有効にします。

```
#+STARTUP: beamer
```

C-c C-b　　org-beamer-select-environment
: org-beamer-mode でこのキーバインドを使うことで beamer の環境や BEAMER_col 属性を素早く選択することができます。

カラムビューはノードにおける環境及び重要なパラメータをセットするうえで、優れたやり方です。カラムのフォーマットがこの特別な目的のためにセットされている確認してください。 コマンド *M-x org-insert-beamer-options-template* はそのようなフォーマットを定義します。

次の例は、beamer へのエクスポートを意図した簡単な Org-mode の文書の例です。

```
#+LaTeX_CLASS: beamer
#+TITLE: プレゼンテーション資料の例
#+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)

* これは最初の構造的な章です。

** フレーム 1 \\ サブタイトル
***  Eric Fraga へありがとう
                                    :BMCOL:B_block:
    :PROPERTIES:
    :BEAMER_env: block
    :BEAMER_envargs: C[t]
    :BEAMER_col: 0.5
    :END:
    Org-mode での最初の beamer の設定
*** みんなへありがとう
                                         :BMCOL:B_block:
    :PROPERTIES:
    :BEAMER_col: 0.5
    :BEAMER_env: block
    :BEAMER_envargs: <2->
    :END:
    議論への寄与してくれたみんなへ
**** これは、beamer の note として処理される。                    :B_note:
** Frame 2 \\ 使わないカラム
*** リクエスト                                                  :B_block:
    この部分を試してみてください!
    :PROPERTIES:
    :BEAMER_env: block
    :END:
```

さらに詳しく知りたい場合は、Worg の文書を見てください。

12.7 DocBook export

Org-mode は、Baoqiu Cui によって作成された DocBook へのエクスポート機能があります。Org-mode のファイルは DocBook のフォーマットで出力され、さらに DocBook のツールやスタイルシートを使って PDF、HTML や man など他のフォーマットに出力することができます。

現在、DocBook のエクスポート機能は DocBook V5.0 をサポートしています。

12.7.1 DocBook export commands

C-c C-e D　　org-export-as-docbook

DocBook ファイルを出力します。Org-mode のファイル 'myfile.org' は DocBook XML ファイルの 'myfile.xml' となります。ファイルは警告なしに上書きされます。アクティブリージョン[15] がある場合は、リージョンのみが出力されます。選択したリージョンが単一のツリー [16] だった場合は, そのツリーの見出しが

[15] transient-mark-mode を有効にしている必要があります。

[16] 現在のツリーを選択するには *C-c @* を使ってください。

文書のタイトルになります。そのツリーの見出しがある、または継承されている場合や、EXPORT_FILE_NAME 属性がある場合は、その名前がエクスポートに使われます。

C-c C-e V　org-export-as-docbook-pdf-and-open

DocBook ファイルが出力され、PDF 処理を経て出力された PDF ファイルが開きます。

DovBook ファイルにエクスポートして PDF に出力するには、XSLT 処理系と XSL-FO 処理系を環境にインストールしておく必要があるので注意してください。変数 org-export-docbook-xslt-proc-command と org-export-docbook-xsl-fo-proc-commandを確認してください。

変数 org-export-docbook-xslt-proc-command でスタイルシートを表わす引数 %s はユーザによってセットされる変数 org-export-docbook-xslt-stylesheet の値で置き換えられます。Org-mode ファイルに #+XSLT: を追加することで、グローバルな設定を封じることができます。

C-c C-e v D

文書の見えている部分だけを出力します。

12.7.2 Quoting DocBook code

次のようなブロックを使えば、Org-mode のファイルで DocBook のコードを引用したり、文章をそのまま DocBook のファイルにコピーすることができます。

```
#+DOCBOOK: エクスポートする DocBook コードの文字列
```

or

```
#+BEGIN_DOCBOOK
これらのマーカの間の行は DocBook エクスポート機能により文字がそのまま出力されます。
#+END_DOCBOOK
```

例えば、DocBook の警告文を含めるには次のような文章を使います。この警告文により、Org-mode のファイルに DocBook コードをのせる時に、文章の文脈に注意を払うでしょう。DocBook コードを正しく引用しないと、DocBook XML ファイルを正確に出力できないかもしれません。

```
#+BEGIN_DOCBOOK
<warning>
  <para>Org-mode ファイルの中で DocBook XML コードを引用する際、何をしているか
  知っておく必要があります。注意が足りないと、DocBook のエクスポート機能により
  正しくない DocBook XML が出力されるかもしれません。</para>
</warning>
#+END_DOCBOOK
```

12.7.3 Recursive sections

DocBook のエクスポート機能は、DocBook の article 要素を使って Org-mode のファイルを articles として出力します。再帰的な sections、例えば section 要素が出力された articles の中で使われます。Org-mode のファイルのトップレベルの見出しは、トップレベルの sections

として出力され、低いレベルの見出しはネストした sections として出力されます。Org-mode のファイルの全体構造は、完全に出力されます。見出しにネストされたレベルはどれだけあっても構いません。

再帰的なセクションを使えば、 出力された DocBook コードを他の `book` や `set` のようなドキュメントタイプに移植したり、再利用したりするのが用意になります。

12.7.4 Tables in DocBook export

Org-mode の表は HTML の表を出力します。HTML の表は DocBook V4.3 からサポートされています。

テーブルに表題がなかった場合、 `informaltable` 要素によって informal table が出力されます。表題があれば、`table` 要素により、テーブルが出力されます。

12.7.5 Images in DocBook export

‘`[[file:img.jpg]]`’ や ‘`[[./img.jpg]]`’ のように説明文にリンクされていない画像は、`mediaobject` タグが使われて DocBook に出力されます。 各 `mediaobject` 要素には、`imagedata` 要素を囲む `imageobject`要素が含まれます。もし、11.2 節「Images and tables」p.125 で説明されているように図の表題を特定するならば、`caption` 要素を `mediaobject` の中に追加します。番号も特定する場合は、 `mediaobject` 要素の中に `xml:id` 属性が出力されます。`figure` 環境のオプションの配置を変更するには、‘`placement=[h!]`’ のように属性に追加します。

画像の属性には `imagedata` 要素がサポートされ、 `align` や `width` のような属性が二つの方法で特定されます。一つ目は変数 `org-export-docbook-default-image-attributes` を設定する方法です。二つ目は`#+ATTR_DOCBOOK:` 行を使う方法です。変数 `org-export-docbook-default-image-attributes` で特定される属性は出力元の Org-mode ファイルに含まれる全ての画像に適用されます (ただし、 `#+ATTR_DOCBOOK:` 行で画像の属性が上書きされている場合は除きます。)。

`#+ATTR_DOCBOOK:` 行は、追加的な画像の属性の指定や個々の画像にデフォルトの画像の属性を上書きするのに使います。もし、`#+ATTR_DOCBOOK:` と 変数 `org-export-docbook-default-image-attributes` に同じ属性が現れた場合、前者の値が優先的に使われます。次の例は画像の属性に関する設定例です。

```
#+CAPTION:    Org-mode のロゴ
#+LABEL:      unicorn-svg
#+ATTR_DOCBOOK: scalefit="1" width="100%" depth="100%"
[[./img/org-mode-unicorn.svg]]
```

デフォルトで DocBook のエクスポート機能は、‘`jpeg`’, ‘`jpg`’, ‘`png`’, ‘`gif`’, そして ‘`svg`’ のような画像のタイプを認識します。変数 `org-export-docbook-inline-image-extensions` を設定することで、DocBook がサポートしている画像のタイプを追加することができます。

12.7.6 DocBook 出力における特殊文字

`\alpha`, `\Gamma`, そして `\Zeta` のような TEX ライクなシンタックスで記述された特殊文字列は、DocBook のエクスポート機能でサポートされています。 そのような文字列は、変数 `org-entities` に格納されているリストにもとづき、`&alpha;`, `&Gamma;`, そして`&Zeta;` のように XML エンティティとして記述されます。対応するエンティティが含まれる DocBook ファイルが出力されると、特殊文字列が認識されます。

変数 org-export-docbook-doctype を設定することで必要なエンティティを含めることができます。例えば、 変数 org-export-docbook-doctype に次のような値を設定することで、XHTML エンティティに含まれる全ての特殊文字列が認識されます。

```
"<!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/) はプロジェクト管理ツールです。プロジェクトのアウトラインと設定して制限をもとにプロジェクトのタイムラインとリソースの割り当てを計算して最適化されたスケジュールを提供します。

TaskJuggler のエクスポート機能は、例えば HTML や LaTeX のような他のエクスポート機能とは少し違い、ドキュメントのノード全てを出力しませんし、ドキュメントのノードの序列に従って出力することもしません。

代わりに、TaskJuggler のエクスポート機能はタスクが定義されているツリーか、このプロジェクトのリソースとして任意に定義されたツリーを探します。そして、それらツリーと全てのノードの中で定義された属性をもとに TaskJuggler ファイルを作成します。

12.8.1 TaskJuggler のエクスポートコマンド

C-c C-e j　　org-export-as-taskjuggler

TaskJuggler ファイルを出力します。

C-c C-e J　　org-export-as-taskjuggler-and-open

TaskJuggler ファイルを出力し、TaskJugglerUI でファイルを開きます。

12.8.2 タスク

いつものように Org-mode でタスクを作ります。各タスクにプロパティを使ってエフォートを指定します (カラムビューを使うと簡単です。)。最終的には、Peter Jones が作成した http://www.contextualdevelopment.com/static/artifacts/articles/2008/project-planning/project-planning.orgの例と似たものになっているはずです。次に、タスクのトップノードを:taskjuggler_project:というタグでマークします (または変数 org-export-taskjuggler-project-tagをカスタマイズします)。これでプロジェクトの計画を *C-c C-e J*で出力する準備ができました。出力されれば、TaskJugglerUI でガントチャートが開くはずです。

12.8.3 リソース

次に特定のタスクにリソースを割り当てることができます。階層的にリソースをまとめることもできます。トップノードのタグは:taskjuggler_resource:になります (または org-export-taskjuggler-resource-tagをカスタマイズします。)。識別子 (‘resource_id’) をリソースに割り当てることもできます (標準的な Org-mode のプロパティについては, 7.1 節「Property syntax」p.61 を参照)。また、エクスポート機能は自動で識別子を生成することが

できます。(識別子はユニークであればよいので、エクスポート機能は見出しの最初の単語を抽出します。詳しくは org-taskjuggler-get-unique-idのドキュメントを読んでください)。識別子を使ってリソースをタスクに再配置することができます。'allocate'属性がタスクで再び実行されます。カラムビューか、タスク上で *C-c C-x p allocate RET <resource_id> RET*と入力することで実行されます。

再配置が実行されると、再び TaskJuggler に出力してリソースの再配置グラフを確認することができます。そのグラフでは、各人がいつ何のタスクをこなしているかがわかります。

12.8.4 属性の出力

エクスポート機能は TODO の状態に関する情報も考慮されています。例えば、タスクが「DONE」とマークされると、それに対応して、TaskJuggler の属性も ('complete 100') となります。タスクリソースやタスクノード上のう TaskJuggler で使われる 'limits', 'vacation', 'shift', 'booking', 'efficiency', 'journalentry', 'rate'のようなリソースの属性や 'account', 'start', 'note', 'duration', 'end', 'journalentry', 'milestone', 'reference', 'responsible', 'scheduling'などのタスクの属性も、出力されます。

12.8.5 依存状態

エクスポート機能はタスクで 'ORDERED' 属性 (5.2.7 節「TODO dependencies」p.48 を参照)、'BLOCKER' 属性 (see 'org-depend.el')、そして選択的に 'depends'属性で表される依存状態を操作することができます。'BLOCKER' 属性も 'depends'属性も 'previous-sibling'のように扱えますし、プロジェクトの他のタスクで定義された識別子への参照 ('task_id') として扱うこともできます。'BLOCKER'属性と 'depends'属性は、カンマやスペースで分けることで複数の依存状態として定義できます。依存状態の属性は、単純に追加することで任意の属性を追加することができます。これまでの例は次のように記述できます。

```
* Preparation
  :PROPERTIES:
  :task_id:  準備
  :ORDERED:  t
  :END:
* 練習の材料
  :PROPERTIES:
  :task_id:  training_material
  :ORDERED:  t
  :END:
** マークアップのガイドライン
   :PROPERTIES:
   :Effort:   2.0
   :END:
** ワークフローのガイドライン
   :PROPERTIES:
   :Effort:   2.0
   :END:
* プレゼンテーション
  :PROPERTIES:
  :Effort:   2.0
  :BLOCKER:  training_material { gapduration 1d } preparation
```

```
:END:
```

12.8.6 レポート

TaskJuggler は多くのレポートを作成できます。(例えば gantt chart, resource allocation など)。TaskJuggler ファイルにあるプロジェクトでどのレポートを作成するか定義することができます。エクスポート機能は自動的にデフォルトのレポートをファイルに挿入します。それらは、`org-export-taskjuggler-default-reports`で定義されています。カスタマイズ機能を使って、他の様々なオプションを変更することができます。ほかのオプションについて知りたい時は *M-x customize-group RET org-export-taskjuggler RET*と入力した確認してください。

さらに詳しい情報や例を見たい時は、`http://orgmode.org/worg/org-tutorials/org-taskjuggler.html`で Org-taskjuggler チュートリアルを見てください。

12.9 Freemind export

Freemind のエクスポート機能は Lennart Borgman によって作成されました。

C-c C-e m　`org-export-as-freemind`

Freemind のマインドマップファイル '`myfile.mm`' を出力します。

12.10 XOXO export

Org-mode には XOXO スタイルで出力するエクスポート機能があります。現在、このエクスポート機能は一般的なアウトライン構造を扱うだけで、Org-mode の特徴を解釈しません。

C-c C-e x　`org-export-as-xoxo`

XOXO ファイルを '`myfile.html`'として出力します。

C-c C-e v x

見えている部分だけを出力します。

12.11 iCalendar エクスポート

Org-mode ユーザーには、プロジェクトの進行を記録している人もいますが、中にはまだ誕生日や予約のために標準的なカレンダーアプリケーションを好む人もいます。このような場合、カレンダーアプリケーションで Org-mode のファイルにあるデッドラインとタイムスタンプのある項目が表示されると便利です。Org-mode はカレンダーの情報を標準的な iCalendar に出力することができます。もし TODO アイテムも出力したいならば、変数`org-icalendar-include-todo`を調整しま。タイムスタンプは VEVENT として出力され、TODO アイテムは VTODO として出力されます。TODO アイテムでないデッドラインもイベントが生成されます。デッドラインと TODO アイテムの予定日は TODO アイテム[17] の開始日と期日として扱われます。カテゴリーには、見出しで定義されたタグやファイルやツリーのカテゴリー[18] が使われます。アラームを設定する方法については、変数`org-icalendar-alarm-time`を参照してください。

標準の iCalendar 形式はそれぞれのエントリーにグローバルでユニークな識別子 (UID) が必要となります。Org-mode ではエクスポートする際にこの識別子を作成します。変数`org-icalendar-store-UID`が設定されていると、UID はエントリーの`:ID:`属性に保存され、次に

[17] 詳しくは変数`org-icalendar-use-deadline`と`org-icalendar-use-scheduled`を見てください。

[18] 継承したタグや TODO の状態を追加するには変数`org-icalendar-categories`をカスタマイズしてください。

このエントリーを使うときに再度利用されます。一つの項目が複数の iCalendar 形式の項目 (タイムスタンプ、デッドライン、スケジューリングされたアイテムや TODO アイテム) となるので, エントリーの中のトリガーによって、Org-mode は UID にプレフィックスをつけます。このようにして、UID はユニークな値となりますが、同期処理では全て異なるエントリーから作られたエントリーとみなされるでしょう。

C-c C-e i　`org-export-icalendar-this-file`
: 現在のファイルから iCalendar 形式のエントリーを作成し、拡張子 '`.ics`'をつけて同じディレクトリに保存します。

C-c C-e I　`org-export-icalendar-all-agenda-files`
: *C-c C-e i*に似ていますが、`org-agenda-files`で指定された全てのファイルで実行されます。それぞれのファイルで iCalendar 形式のファイルが作成されます。

C-c C-e c　`org-export-icalendar-combine-agenda-files`
: `org-agenda-files`で指定されたファイルから単一の iCalendar 形式のファイルを生成し、`org-combined-agenda-icalendar-file`で指定されたファイルに出力します。

エクスポート機能はエントリーが SUMMARY、DESCRIPTION そして LOCATION 属性[19] を持っていた場合、それを継承します。なかった場合は SUMMARY 属性は見出しから抽出され、DESCRIPTION 属性はその内容 (`org-icalendar-include-body`で制限されます。) から抽出されます。

このカレンダーを読んだり更新したりするのにベストな方法は、使うカレンダーアプリケーション次第です。FAQ ではこの問題についてカバーしています。

[19] `org-use-property-inheritance`を設定した場合、それに応じて LOCATION 属性は高い階層からその値が継承されます。

13 Publishing

Org-mode は連結された Org-mode のファイルから成る *projects* の自動的な HTML への変換設定をあなたに許可します。あなたはまた、エクスポートした HTML ページと画像やソースコードのような関係する添付ファイルの自動的なアップロードを Org-mode に設定できます。

あなたはまた、Org-mode から PDF へ変換することもでき、さらに、サーバ上で両方のファイルを利用できるように HTML と PDF への変換を組み合わせることも可能です。

公開は David O'Toole によって Org-mode に寄与されました。

13.1 Configuration

公開はファイルの指定、公開先、プロジェクトのその他多くのプロパティの重要な設定を必要とします。

13.1.1 `org-publishing-alist`変数

公開は`org-publish-project-alist`というある変数の値により設定されます。リストの各要素が 1 つのプロジェクトの設定で、2 種類の設定方法について以下に示します:

```
("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" ...))
```

どちらの場合もプロパティと値の指定によって設定されます。プロジェクトは公開されるファイルのセットと、それらが公開される時に使用する設定を定義します。上記の 2 番目の形式を取る場合、`:components`プロパティの個々のメンバは異なる公開設定を持つファイルをまとめたサブプロジェクトとして扱われます。あなたが “メタプロジェクト” を公開する時、全てのコンポーネントは特定の順序で公開されます。

13.1.2 ファイルの送り元と送り先

多くの<プロパティは任意に選択でき、しかし多くは常にセットされているべきです。特に、Org-mode はソースファイルをどこで探すか、またファイルの公開先を知る必要があります。

`:base-directory`
: 公開するソースファイルを含むディレクトリ

`:publishing-directory`
: アウトプットファイルのあるディレクトリは公開されます。あなたは Emacs‘`tramp`’ パッケージの構文のファイル名を使ってウェブサーバに直接公開することができます。もしくは、ローカルディレクトリに公開したり、あなたのウェブサイト (13.2 節「Uploading files」p.157 を参照) にアップロードするための拡張ツールを使うことができます。

`:preparation-function`
: 公開プロセスの開始前に呼ばれる関数または関数のリストです (例えば、公開のためのファイルの更新に使う `make`を実行します)。`project-plist`変数であるプロジェクトのプロパティリストはこの呼び出しにスコープされます。

`:completion-function`　公開プロセスが完了した後に呼ばれる関数または関数のリストです (例えば、出力したファイルの権限を変更します)。`project-plist`変数であるプロジェクトのプロパティリストはこの呼び出しにスコープされます。

13.1.3 Selecting files

デフォルトでは、ベースディレクトリの中の拡張子が‘`.org`’である全てのファイルはプロジェクトの一部であるとみなされます。これはプロパティの設定により変更することができます:

`:base-extension`　ソースファイルの拡張子 (. を除きます!) です。これは実際は正規表現です。もしあなたが`:base-directory`の中の全てのファイルを取得したいのであれば、`any`シンボルをセットします。

`:exclude`　そのファイルの拡張子が選択されていたとしても、公開したくないファイル名にマッチする正規表現。

`:include`　`:base-extension`と`:exclude`にかかわらずインクルードするファイルのリスト。

`:recursive`　nil でなければ、ベースディレクトリから再帰的に公開するファイルをチェックします。

13.1.4 Publishing action

公開は、ファイルを宛先ディレクトリにコピーし、場合によっては変換することです。デフォルトの変換は Org-mode ファイルから HTML ファイルへのエクスポートで、これは HTML エクスポーター (12.5 節「HTML export」p.135 を参照) を呼び出す`org-publish-org-to-html`によって行われます。しかしあなたはまた、`org-publish-org-to-pdf`を用いた PDF ファイルのようなコンテンツも公開することができます。もしあなたがアーカイブされた、コメントされた、除外タグがついたツリーを消去した Org-mode ファイル自身を公開したいのであれば、`org-publish-org-to-org`を使い、`:plain-source`パラメラと/または`:htmlized-source`をセットします。これは公開ディレクトリ[1]に‘`file.org`’と‘`file.org.html`’を作ります。画像のようなその他のファイル公開ディレクトリにコピーされる必要があります; このために、あなたは`org-publish-attachment`を使うことができます。Org-mode でないファイルのために、あなたは常に公開用の関数を指定する必要があります:

`:publishing-function`　ファイルの公開を実行する関数です。これは、順番に呼び出される関数のリストにすることもできます。

`:plain-source`　nil でなければ、プレーンソースを公開します。

`:htmlized-source`　nil でなければ、HTML に変換されたソースを公開します。

関数には 3 つの引数を渡す必要があります: 最低でも`:publishing-directory`プロパティを含むプロパティリスト、公開されるファイル名、出力ファイルの公開ディレクトリへのパスです。それは指定されたファイルを取り、(もしあれば) 必要な変換を行い、出力をコピー先ディレクトリに置きます。

[1] もしソースと公開ディレクトリが同じであれば、‘`file-source.org`’と‘`file-source.org.html`’です。このようなセットアップでは、プロジェクトが公開される次回、新しい Org-mode ファイルとみなされるソースファイルが公開されるのを防ぐためにあなたは`:exclude \"-source\\\\.org\"`を`org-publish-alist`中のプロジェクトの定義に追加する必要があることに注意しください。

13.1.5 HTML/LaTeX エクスポート機能のオプション

プロパティリストは HTML と LaTeX エクスポート機能の多くのエクスポートオプションをセットするために使うことができます。多くの場合、これらのプロパティは Org-mode 中のユーザ変数と対応します。下のテーブルはそれらが所属する変数と同様なプロパティのリストです。各変数の詳細については、ドキュメントを参照してください。

:link-up	org-export-html-link-up
:link-home	org-export-html-link-home
:language	org-export-default-language
:customtime	org-display-custom-times
:headline-levels	org-export-headline-levels
:section-numbers	org-export-with-section-numbers
:section-number-format	org-export-section-number-format
:table-of-contents	org-export-with-toc
:preserve-breaks	org-export-preserve-breaks
:archived-trees	org-export-with-archived-trees
:emphasize	org-export-with-emphasize
:sub-superscript	org-export-with-sub-superscripts
:special-strings	org-export-with-special-strings
:footnotes	org-export-with-footnotes
:drawers	org-export-with-drawers
:tags	org-export-with-tags
:todo-keywords	org-export-with-todo-keywords
:priority	org-export-with-priority
:TeX-macros	org-export-with-TeX-macros
:LaTeX-fragments	org-export-with-LaTeX-fragments
:latex-listings	org-export-latex-listings
:skip-before-1st-heading	org-export-skip-text-before-1st-heading
:fixed-width	org-export-with-fixed-width
:timestamps	org-export-with-timestamps
:author	user-full-name
:email	user-mail-address : addr;addr;..
:author-info	org-export-author-info
:email-info	org-export-email-info
:creator-info	org-export-creator-info
:tables	org-export-with-tables
:table-auto-headline	org-export-highlight-first-table-line
:style-include-default	org-export-html-style-include-default
:style-include-scripts	org-export-html-style-include-scripts
:style	org-export-html-style
:style-extra	org-export-html-style-extra
:convert-org-links	org-export-html-link-org-files-as-html
:inline-images	org-export-html-inline-images
:html-extension	org-export-html-extension
:html-preamble	org-export-html-preamble
:html-postamble	org-export-html-postamble
:xml-declaration	org-export-html-xml-declaration

```
:html-table-tag           org-export-html-table-tag
:expand-quoted-html       org-export-html-expand
:timestamp                org-export-html-with-timestamp
:publishing-directory     org-export-publishing-directory
:select-tags              org-export-select-tags
:exclude-tags             org-export-exclude-tags
:latex-image-options      org-export-latex-image-default-option
```

org-export-with-*変数の大部分は LaTeX エクスポートで nilと tのそれぞれで:TeX-macrosと:LaTeX-fragmentsオプションを除き HTML と LaTeX エクスポート機能の両方で同じ効果を持ちます。このオプションのリストをチェックするには org-export-plist-varsを見てください。

プロパティが org-publish-project-alist中の値で与えられた時、その設定は公開時に対応するユーザ変数を上書きします。オプションをファイル (12.2 節「Export options」p.132 を参照) 内でセットしても、全て上書きされます。

13.1.6 公開ファイル間のリンク

ある Org-mode ファイルから別のファイルへのリンクを作成する方法として、あなたは‘[[file:foo.org][The foo]]’、もしくはより単純な‘file:foo.org.’(第 4 章「Hyperlinks」p.36 を参照) のような形式を使うことができます。公開するとき、このリンクは‘foo.html’へのリンクとなります。この方法では、あなたはあなたの\"Org-mode ウェブ\"プロジェクトのページと HTML へ公開するときにリンクを連結することができます。もしあなたが Org-mode のソースファイルを公開したりそこへのリンクを作成したいのであれば、file:リンクは対応する‘html’ファイルにリンクが変換されるため、file:リンクの代わりに http:リンクを使います。

あなたはまた、画像のような関連するファイルへのリンクを作成することができます。あなたが関連するファイル名に慎重になっているとき、そしてあなたが関連するファイルをアップロードする Org-mode の設定をするとき、これらのリンクも働きます。使い方の例は、13.3.2 節「Complex example」p.158 を参照してください。

時々、公開される Org-mode ファイルはあなたの製品環境内でのみ有効で、公開する場所では有効でないなリンクを含むことができます。この場合、次のプロパティを使います。

`:link-validation-function`　有効にする関数

このプロパティはリンクの有効性をチェックする関数を定義します。この関数はファイル名と製品環境内で解釈できるファイル名に関連するディレクトリ 2 つの引数を持ちます。もしこの関数が nilを返すなら、HTML ジェネレータは HTML ファイルにリンクを除き説明のみ挿入します。この関数へのオプションとして、org-publish-project-alist中のプロジェクトの一部であるファイルが与えられているなチェックする org-publish-validate-linkがあります。

13.1.7 サイトマップの生成

以下のプロパティはプロジェクトのファイルのマップの公開を制御するために使うことができます。

`:auto-sitemap`　nil でないとき、org-publish-current-projectもしくは org-publish-allの実行中にサイトマップを公開します。

:sitemap-filename	出力するサイトマップのファイル名です。デフォルトでは‘sitemap.org’(‘sitemap.html’になります) です。
:sitemap-title	サイトマップページのタイトルです。デフォルトはファイル名です。
:sitemap-function	サイトマップの生成に使うプラグイン関数です。デフォルトでは org-publish-org-sitemapで、プロジェクト中の全てのファイルへのリンクのプレーンリストを生成します。
:sitemap-sort-folders	フォルダがサイトマップに表示される位置です。先頭か最後にそれぞれフォルダを表示するため、これに first(デフォルト) か lastをセットしてください。
:sitemap-sort-files	サイトマップ中でのファイルのソート方法です。alphabetically(デフォルト)、chronologicallyまたは anti-chronologicallyをセットします。chronologicallyはファイルを日付の古い順番にソートし、anti-chronologicallyは新しい順番にソートします。alphabeticallyはファイルをアルファベット順にソートします。ファイルの日付は org-publish-find-datにより検索します。
:sitemap-ignore-case	ソート時に大文字と小文字を区別するかどうかです。デフォルトでは nilです。
:sitemap-file-entry-format	このオプションでサイトマップのエントリがサイトマップでフォーマットされているか知ることができます。これはエスケープシークエンスを含むフォーマット文字列です: %tはファイルのタイトル、%aはファイルの著者、%dはファイルの日付を表します。日付は org-publish-find-date関数により検索され、org-publish-sitemap-date-formatによりフォーマットされます。デフォルトは%tです。
:sitemap-date-format	エントリの日付をどのようにフォーマットするかを決める format-time-string関数用のフォーマット文字列です。このプロパティはデフォルトでは%Y-%m-%dである rog-publish-sitemap-date-formatを迂回します.

13.1.8 Generating an index

Org-mode は公開するプロジェクトのファイルを跨ぐインデックスを生成することができます。

:makeindex	nil でないとき、‘theindex.org’ファイルにインデックスを作成し、‘theindex.html’に公開します。

:makeindexを設定したプロジェクトの最初の公開のとき、ファイルは作成されます。ファイルは#+include: \"theindex.inc\"ステートメントのみを含みます。あなたはタイトルやスタイル情報等をステートメントの周囲に追加することができます。

13.2 Uploading files

Tramp に深く頼っている Org-mode の組込みの公開機能を使うことは望ましくなく、rsyncや unisonのようなサードパーティの同期ツールを人々は活用しています。Tramp は非常に便利で協力ですが、複数のファイルを送る場合効率的でない場合があり、ヘビーに使う環境下での問題の原因として知られています。

専用の同期ユーティリティはいくつかのアドバンテージを提供します。タイムスタンプの比較に加えて、コンテンツや権限、属性のチェックも行います。そういうわけで、あなたの WEB ページをローカルディレクトリ (場合によると Org-mode ファイルの場所で) に公開し、そして ‘unison’や ‘rsync’をリモートホストとの同期に使うために選ぶことができます。

Unison はファイルを信頼できるリモートの公開先に転送する設定を行うことができるので、プロジェクトの公開についての定義を非常にシンプルにすることができます。org-publishを用いるあなたの Org-mode ファイルの処理と同期ツールの設定により、正しい場所で全てのファイルをシンプルに保ちます。このシナリオでは、‘jpg’や ‘css’、‘gif’のようなファイルの添付については、サードパーティの同期ツールがそれらを同期するため、設定する必要がありません。

ローカルディレクトリの公開はリモートへの公開に比べると高速で、全てのプロジェクトをより簡単に再公開できます。もしあなたが org-publish-use-timestamp-flagを nilに設定しているのであれば、変更が加えられた#+INCLUDEでインクルードしたたソースファイルのような外部ファイルの再公開で恩恵を受けます。Org-mode におけるタイムスタンプのメカニズムはファイルに変更が加えられたかどうか検出できるほどスマートではありません。

13.3 Sample configuration

私達は 2 つの設定の例を以下に示します。1 つ目の設定は Org-mode ファイルのみのシンプルなプロジェクトの公開です。2 つ目の設定は複数のコンポーネントを持つプロジェクトで、少し複雑な例です。

13.3.1 例：シンプルな公開用の設定

この例は Org-mode ファイルのセットをローカルマシンの ‘public_html’にディレクトリ公開します。

```
(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 例：複雑な公開用の設定

この少し複雑な例は HTML に変換された org-mode ファイル、画像ファイル、Emacs Lisp のソースコード、スタイルシートなどを含む全てのウェブサイトを公開します。公開ディレクトリはリモートと除外されたプライベートなファイルです。

リンクが保持されていることを保証するには、ウェブサーバ上のディレクトリ構造の再現、ファイルの相対パスの使用を慎重にするべきです。例えば、もしあなたの Org-mode ファイルが‘~/org’に、公開する画像が‘~/image’にあるとすると、あなたは画像のリンクを以下のようにします。

```
file:../images/myimage.png
```

ウェブサーバ上では、画像への相対パスは同じにするべきです。あなたは\"images\" フォルダをウェブサーバ上の正しい一に置き、画像をそこに公開することでこれを達成することができます。

```
(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

一度適切に設定すると、Org-mode は以下のコマンドを使うことで公開することができます。

C-c C-e X　org-publish

プロンプトでプロジェクトを指定し、プロジェクトの全てのファイルを公開します。

C-c C-e P　org-publish-current-project
: 現在のファイルを含むプロジェクトを公開します。

C-c C-e F　org-publish-current-file
: 現在のファイルのみ公開します。

c-c C-e E　org-publish-all
: 全てのプロジェクトを公開します。

Org-mode はファイルが変更された時を記録するためにタイムスタンプを使います。上記の関数は通常変更されたファイルのみを公開します。あなたはこれを上書きし、上記コマンドに前置引数を付けて実行するか org-publish-use-timestamps-flag変数をカスタマイズすることで全てのファイルを強制的に公開できます。もし、ファイルが#+SETUPFILE:や#+INCLUDE:を用いて他のファイルをインクルードしているなら特別必要です。

14 ソースコードとの連携

‘src’ブロックを使う事で Org-mode の文書にソースコードを含めることができます。例えば

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

Org-mode では生のソースコードと連携する幾つかの機能を用意してます。その機能には、コードブロックをそのメジャーモードで編集する機能、コードブロックを評価する機能、コードブロックをソースコードへと変換する機能 (文芸的プログラミングでは『tangling』と知られている機能)、そしてコードブロックとその結果を幾つかのフォーマットに沿ってエクスポートする機能が含まれます。この機能はもともと Org-bablel と名付けられ、Eric Schult と Dan Davidson によって開発されました。

以下のセクションでは、Org-mode でコードブロックを取り扱う機能について説明します。

14.1 Structure of code blocks

コードブロックの構造は以下の通りです:

```
#+srcname: <name>
#+begin_src <language> <switches> <header arguments>
  <body>
#+end_src
```

スイッチとヘッダー引数は省略可能です。次のように、コードを文章にインラインで埋め込むこともできます。

```
src_<language>{<body>}
```

もしくは

```
src_<language>[<header arguments>]{<body>}
```

`<name>`
: この「name」はコードブロックと関連付けられています。これは、Org-mode のファイル内でテーブルに名前を付ける ‘`#+tblname`’行と似ています。コードブロックの名前を参照することで、そのファイル内の別の場所にあるコードブロックや別のファイルの別の場所にあるコードブロック、または Org-mode のテーブルの計算式 (3.5 節「The spreadsheet」p.25 を見て下さい) にあるコードブロックから評価することが可能となります。

`<language>`
: ブロック内にあるコードの言語です。

`<switches>`
: コードブロックのエクスポートを管理する省略可能なスイッチです (11.3 節「Literal examples」p.125 を参照してください)

`<header arguments>`
: コードブロックの評価、コードブロックのエクスポート、そしてコードブロックからのソースコードの抽出において、省略可能なヘッダー引数が多くの側面を管理します。14.8 節「Header arguments」p.165 セクションを見て下さい。プロパ

ティを使うことで、バッファー毎もしくはサブツリー毎にヘッダー引数を設定できます。

`<body>`　ソースコードです。

14.2 Editing source code

現在のコードブロックを編集するには *C-c '*を使ってください。コードブロックの本文を含んだ編集バッファが、その言語のメジャーモードで新たに開かれます。このバッファを保存すると、Org-mode のバッファへ新しい内容を上書きします。終了させるには再度 *C-c '*を使って下さい。

編集バッファにて `org-src-mode`のマイナーモードが起動します。編集バッファの振る舞いを設定するために、以下の変数が使えます。さらなる設定オプションについては、カスタマイズのグループ `org-edit-structure`も見て下さい。

`org-src-lang-modes`
: `<lang>-mode`と名付けられたメジャーモードが Emacs に存在する場合、そのメジャーモードで編集バッファを開きます。ここで`<lang>`とは、コードブロックのヘッダー行で指名された言語のことです。存在するメジャーモードに任意の言語名を割り当てるために、この変数を使えます。

`org-src-window-setup`
: 編集バッファを新規作成する時に、Emacs のウィンドウが再配置される方法を管理します。

`org-src-preserve-indentation`
: ホワイトスペース (空白) のインデントが非常に重要である Python などの言語を tangling(コード本文を抽出) するときに、この変数は特に役に立ちます。

`org-src-ask-before-returning-to-edit-buffer`
: デフォルトでは、既に開いてある編集バッファへと戻る前に、Org-mode が確認のプロンプトを出します。確認のプロンプトを出さずにバッファを変更するには、この変数を nil に設定してください。

Org-mode バッファ内にもかかわらず、記述した言語をメジャーモードの配色で表示したい場合には、変数 `org-src-fontify-natively`を設定してください。

14.3 Exporting code blocks

コードブロックの内容、コードブロックを評価した結果、いずれでもないもの、もしくは内容と結果両方をエクスポートできます。ほとんどの言語では、デフォルトの設定ではコードブロックの内容をエクスポートします。しかし、ある言語 (例えば `ditaa`) では、デフォルトの設定でコードブロックを評価した結果をエクスポートします。コードブロックの本文をエクスポートするには、11.3 節「Literal examples」p.125 を参照してください。

ヘッダー引数`:exports`を設定することで、エクスポート時の振る舞いを指定できます:

ヘッダー引数:

`:exports code`
: ほとんどの言語でのデフォルトの設定です。11.3 節「Literal examples」p.125 で述べたように、コードブロックの本文が出力されます。

`:exports results`
: エクスポートするために、コードブロックが評価され、その結果が Org-mode のバッファに出力されます。バッファ内に以前評価した結果があればその結果を更新し、もし以前に評価した結果が無ければ直ちに今回の評価結果をコードブロックの後に追記します。コードブロックの本文はエクスポートされません。

`:exports both`
: コードブロックとその結果の両方がエクスポートされます。

`:exports none`
: コードブロックとその結果のいずれもエクスポートされません。

エクスポートの間、コードブロックの評価を抑止することも可能です。変数 `org-export-babel-evaluate`を `nil`と設定することで、エクスポートのプロセスの一部としてのコードブロックの評価を抑止できます。例えば wiki のマークアップ言語として Org-mode を使う場合など、信頼できない可能性がある Org-mode が自動的にエクスポートされる場合には、この設定が役立ちます。

14.4 Extracting source code

ソースブロックからコードを抽出して純粋なソースコードファイルを作成することを、“tangling” といいます。この用語は文芸的プログラミングのコミュニティで使われている専門用語です。コードブロックを “tangling” する間、変数と “noweb” スタイルの参照点 (14.10 節「Noweb reference syntax」p.179 を参照) との両方を展開できる `org-babel-expand-src-block`を使ってコードブロックの本文を展開します。

Header arguments

`:tangle no`
: デフォルトの設定です。コードブロックは、tangle されたアウトプットには含まれません。

`:tangle yes`
: tangle されたアウトプットにコードブロックが含まれます。アウトプットされるファイルの名前は、org ファイルの拡張子 ‘`.org`’をブロックに記述した言語の拡張子名に置換したものです。

`:tangle filename`
: ‘`filename`’への tangle されたアウトプットにはコードブロックを含みます。

関数

`org-babel-tangle`
: 現在のファイルを tangle します。キーバインドは *C-c C-v t*です。

`org-babel-tangle-file`
: tangle するファイルを選びます。キーバインドは *C-c C-v f*です。

Hooks

`org-babel-post-tangle-hook`
: 関数 `org-babel-tangle`によって tangle されたコードファイルの中から、この hook が実行されます。tangle されたコードファイルの後処理、編集もしくは評価を、実用例として含められます。

14.5 Evaluating code blocks

コードブロックを評価し[1]、その結果を Org-mode バッファに表示することが可能です。デフォルトでは、emacs-lispのコードブロックだけを評価しますが、多くの言語での評価のサポートがあります。サポートされている言語のリストは、14.7 節「Languages」p.164 で見ることができます。コードブロックを定義する構文について 14.1 節「Structure of code blocks」p.160 を参照してください。

コードブロックを評価する方法は多くあります。一番簡単な方法は、コードブロック[2] にて *C-c C-c*もしくは *C-c C-v e*をと入力することです。これが関数 org-babel-execute-src-blockをコール、コードブロックを評価し、その結果を Org-mode バッファに挿入します。

Org-mode のバッファ内、もしくは Org-mode のテーブル内のどこからでも、名前付きのコードブロックを評価することも可能です。現在の Org-mode バッファや “Library of Babel”(14.6 節「Library of Babel」p.164 にあるコードブロックを離れた場所から実行するには、#+call 行 (もしくは同義語として#+function行や#+lob行) が使えます。これらの行は、以下の構文を使います。

```
#+call: <name>(<arguments>) <header arguments>
#+function: <name>(<arguments>) <header arguments>
#+lob: <name>(<arguments>) <header arguments>
```

<name>　評価されるコードブロックの名前です。

<arguments>
: この場所に記述された引数をコードブロックへと渡します。通常の関数呼び出しの構文を使って呼び出されたコードブロック内では、これらの引数はヘッダー引数:varと関係しています。例えば、doubleと名付けられたもともとのコードブロックではヘッダー引数が:var n=2と書かれている場合、そのコードブロックへ数値の 4 を渡すには、call 行を#+call: double(n=4)と記述します。

<header arguments>
: 関数呼び出しの後にヘッダー引数をセットできます。ヘッダー引数の詳細については 14.8 節「Header arguments」p.165 を参照してください。

上の<header arguments>という部分に記述された全てのヘッダー引数は、#+call:行の評価に適用されます。しかし、時には、コードブロックへ渡すヘッダー引数を指定することが好ましい場合もあります。

これには、以下の選択可能な拡張された構文を使えます。

```
#+call: <name>[<block header arguments>](<arguments>) <header arguments>
```

角括弧 ([]) で囲まれた<block header arguments>部分に置かれたヘッダー引数がどのようなものであっても、名前付きコードブロックの評価に適用されます。#+call:行へとヘッダー引数を渡す例については、「Header arguments in function calls」p.167 を参照してください。

---

[1] コードを評価するときはいつでも、そのコードが害をなす可能性があります。Org-mode は、ユーザーから明示的な確認をできたときのみにコードを評価する安全装置をいくつも用意しています。これらの安全装置については (加えて、それらを無効化する方法についても)、15.4 節「Code evaluation security」p.183 を参照してください。

[2] 変数 org-babel-no-eval-on-ctrl-c-ctrl-cを設定することで、キーバインド *C-c C-c*からコードの評価を除去できます。

14.6 Library of Babel

“Library of Babel” は、どの Org-mode のファイルからも呼び出せるコードブロックのライブラリです。そのライブラリは、Org-mode の ‘contrib’ディレクトリの Org-mode ファイルに格納されています。Org-mode のユーザーは、一般的に有用だと考えられる関数をそのライブラリ内に置くことができます。

“Library of Babel” で定義されたコードブロックは、あたかも現在の Org-mode バッファで処理されるかのように、リモートで呼び出せます (リモートでコードブロックを評価する構文についての情報は、14.5 節「Evaluating code blocks」p.163 を参照してください)。

どんな Org-mode ファイルに記述されたコードブロックでも、`org-babel-lob-ingest` を使って “Library of Babel” の関数へロードできます。キーバインドは *C-c C-v i*です。

14.7 Languages

コードブロックでは、以下の言語を使えます。

言語	識別子	言語	識別子
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

言語特有の文書が存在する場合があります。文書が存在する場合、それは`http://orgmode.org/worg/org-contrib/babel/languages`にて見つけられます。

どの言語を評価するかの管理には、関数`org-babel-load-languages`を使います (デフォルトでは`emacs-lisp`のみを評価可能です)。この変数を設定するには、カスタマイズのインターフェースを使うか、以下のコードを emacs の設定ファイルに追加します。

次のコードでは、`emacs-lisp`を評価させないように設定し、`R`のコードブロックを評価させるように設定しています。

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((emacs-lisp . nil)
   (R . t)))
```

上記方法に加えて、`require`関数を使って関連する elisp ファイルをロードすることでも、言語のサポートを有効にできます.

以下は、`clojure`コードブロックの評価を有効にしています。

```
(require 'ob-clojure)
```

14.8 Header arguments

ヘッダー引数を通してコードブロックの機能を設定できます。このセクションではヘッダー引数の使い方を概説した後に、各ヘッダー引数の詳細について説明します。

14.8.1 Using header arguments

ヘッダー引数の値は 6 通りの方法で設定できます。以下の説明で、後に説明されるものほど、個別の設定として優先されます。

System-wide header arguments

変数 org-babel-default-header-argsをカスタマイズすることで、システム全体にわたるヘッダー引数の値を指定できます。

```
:session    => "none"
:results    => "replace"
:exports    => "code"
:cache      => "no"
:noweb      => "no"
```

例えば、次の例ではヘッダー引数:nowebの初期値を yesと設定できます。この設定は、ソースコードブロックを評価する時にデフォルトの設定として:nowebの参照記号を展開します。

```
(setq org-babel-default-header-args
(cons '(:noweb . "yes")
(assq-delete-all :noweb org-babel-default-header-args)))
```

Language-specific header arguments

それぞれの言語用にデフォルトのヘッダー引数を定義できます。オンライン http://orgmode.org/worg/org-contrib/babelで閲覧可能な言語特有の文書を見て下さい。

Buffer-wide header arguments

Org-mode ファイルにある特別な行を使って、バッファー全体にわたるヘッダー引数を設定できます。その行では、キーワード#BABEL:に続いて、普通のヘッダー引数の構文を使って指定された一連のヘッダー引数があります。

次の例は、sessionを*R*と設定し、バッファ内にあるコードブロック毎の resultsを silentと指定しています。この設定により、全ての (コードブロックが) 同一のセッションで実行されること、そして、結果はそのバッファに挿入されないことを確かにしています。

```
#+BABEL: :session *R* :results silent
```

Header arguments in Org-mode properties

ヘッダー引数は Org-mode のプロパティから読み出すこともできます (7.1 節「Property syntax」p.61 を見て下さい)。プロパティはバッファ全体もしくは見出しごとの単位で設定できます。例えば、バッファ内の全コードブロックのヘッダー引数を設定するには、以下のようにします。

```
#+property: tangle yes
```

デフォルトのヘッダー引数を設定するためにプロパティを使う場合、プロパティは継承 (関係) に沿って参照されます。したがって、以下の見出しに属するサブツリー内の全てのコードブロックでは、ヘッダー引数:cacheの値はデフォルトで yesとなります:

```
* アウトラインの見出し
:PROPERTIES:
:cache:    yes
:END:
```

このように設定されたプロパティは、org-babel-default-header-argsに設定されたプロパティを上書きします。Org-mode ドキュメント内のプロパティを設定する関数 org-set-propertyを使うのが便利です。キーバインドは *C-c C-x p*です。

Code block specific header arguments

コードブロックのレベルでヘッダー引数に値を割り当てる方法が最も一般的です。#+begin_src行の一部として、ヘッダー引数とその値とを並べることによって設定されます。このように設定されたプロパティは変数 org-babel-default-header-argsとプロパティとして設定されたヘッダー引数との両方を上書きします。以下に示す例では、実行結果をバッファに挿入しないように、ヘッダー引数:resultsを silentと設定し、加えて HTML や LaTeX へのエクスポート時にコードブロックの本文だけが保存されるように、ヘッダー引数:exportsを codeと設定しています。

```
#+source: factorial
#+begin_src haskell :results silent :exports code :var n=0
fac 0 = 1
fac n = n * fac (n-1)
#+end_src
```

同様に、インラインのコードブロックにヘッダー引数を設定することも可能です:

```
src_haskell[:exports both]{fac 5}
```

コードブロックのヘッダー引数を複数行にわたって記述できます。=#+header:=もしくは =#+headers:=の行を、コードブロックの前に記述するか、名前付きコードブロックの名前とコード本文との間に記述します。

名前を付けていないコードブロックに複数行のヘッダー引数を設定します:

```
#+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
```

名前を付けたコードブロックに複数行のヘッダー引数を設定します:

```
#+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

一番個別のレベルにおいて、Babel ライブラリのためのヘッダー引数もしくは関数の呼び出し行は、下に示す 2 つの例のように設定できます。`#call:`行の構造についての更なる情報は、14.5 節「Evaluating code blocks」p.163 を参照してください。

以下は、ヘッダー引数`:exports results`を`#+call:`行の評価に適用します。

```
#+call: factorial(n=5) :exports results
```

以下は、ヘッダー引数`:session special`をコードブロック`factorial`の評価に適用します。

```
#+call: factorial[:session special](n=5)
```

14.8.2 Specific header arguments

以下のヘッダー引数が定義されています:

14.8.2.1 `:var`

ヘッダー引数`:var`はコードブロックへ引数を渡すために使われます。コードブロックに含まれている引数を受け取る方法は、言語によって異なります; これらは、言語固有の文書にて説明されています。しかし、引数を指定する構文はすべての言語で共通です。引数に渡される値は、定数 (リテラル値)、Org-mode のテーブルと literal example ブロックからの値、他のコードブロックの結果、あるいは Emacs Lisp のコードです。—Emacs_Lisp については、以下の見出し “Emacs Lisp evaluation of variables” を見て下さい。

これらの値は配列のようにインデックス化できます—以下の見出し “indexable variable values” を見て下さい。

ヘッダー引数`:var`を使ってコードブロックへ引数を渡すには、以下の構文を使います。

```
:var name=assign
```

ここで、`assign`は以下のフォームのどれかをとることができます。

- 定数 (リテラル値) 文字列 (例:`"string"`)、もしくは数字 (例:`9`) のどちらかです。
- 参照テーブルの名前:

```
#+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
```

`#+srcname:`(訳者注:`#+source:`と同一のもの) で割り当てられたコードブロックの名前で、後ろに括弧「()」が続きます。

```
#+begin_src emacs-lisp :var length=table-length()
(* 2 length)
#+end_src

#+results:
: 8
```

加えて、:varにより参照されるコードブロックへと引数を渡すことも可能です。コードブロック名に続く括弧「()」内の引数が渡されます:

```
#+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(引数を指定する別の構文)

コードブロックの#+source:行を利用するという方法でも引数を指定できます。もしかすると、こちらの方法がより自然かもしれません。以下の例のように、引数はソースの名前に続く括弧「()」の内に、カンマ「,」で分離された形でひとまとめにされています。

```
#+source: double(input=0, x=2)
#+begin_src emacs-lisp
(* 2 (+ input x))
#+end_src
```

Indexable variable values(インデックス化可能な変数の値)

変数の “indexing” インデックス化によって割り当てられた変数の一部分を参照できます。インデックス (添え字) は 0 から始まります。インデックスが負の値の場合、それは要素の最後から逆順に数え上げたものです。インデックスが「,」で分割されているとき、後続する部分のそれぞれが、次に深いネスティング (入れ子構造) にインデックス化するか、値の次元にインデックス化します。:hlines、:colnames、そして:rownamesのようなヘッダー引数と関連する他のテーブルが変更される前に、このインデックス化が行われることに注意して下さい。次の例では、テーブル example-table1 行目の最後のセルを変数 dataに割り当てています:

```
#+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
```

:で区切られた 2 つの整数を使うことで、変数の範囲 (行や列?) を参照できます。この場合、包括的な範囲を参照します。例えば、以下では example-tableの中央 3 行を dataに割り当てています。

```
#+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 |
```

加えて、インデックスが空の場合もしくは単一文字*の場合のどちらでも、全ての範囲すなわち 0:-1を意味していると解釈されます。次の例では、1 列目すべてが参照されます。

```
#+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 |
```

コードブロックの結果をテーブルの様にインデックス化できます。どんな次元数でもインデックス化できます。以下の例に示すように、次元はそれぞれカンマで分割されます。

```
#+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 による変数の評価)

変数の値を初期化するときに Emacs Lisp のコードを使えます。変数の値が (, [, ' もしくは `で始まる場合、変数は Emacs Lisp として評価され、その評価結果を変数の値として代入されます。次の例では、この評価を使って Org-mod のファイル名を確かにコードブロックへ渡しています—元の Org-file 内でヘッダー引数が間違いなく評価されるということに注意して下さい。コードブロックで評価する場合そのような保証はありません。

```
#+begin_src sh :var file-name=(buffer-file-name) :exports both
  wc -w $file
#+end_src
```

テーブルやリストから読み出された値は Emacs Lisp として評価されないことに注意して下さい。以下はその例です。

```
#+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

ヘッダー引数:resultsには 3 つのクラスがあります。コードブロックでは、クラス毎に 1 個のオプションだけを付与できます。

- ヘッダー引数 **collection** は、コードブロックからどのように結果を集めかを指定します。
- ヘッダー引数 **type** はコードブロックがどのタイプの結果を返すかを指定します—これは、Org-mode のバッファーへ結果を挿入する方法に影響します。
- ヘッダー引数 **handling** はコードブロックの評価の結果をどのように扱うかを指定します。

Collection

以下のオプションは、コードブロックから結果を収める方法を指定します。どちらかのオプションしか指定できません。

- value デフォルトの設定です。コードブロックの最終行の値が結果になります。このヘッダー引数は functional mode で評価します。このタイプの返り値を使う場合、Python な

どの言語では、ソースコードブロックの本文で `return`文が必要なことに注意して下さい。例えば、`:results value`のように記述します。

- `output` コードブロックで実行される間に標準出力 (STDOUT) へと出力されたもの全てを収集して結果とします。このヘッダー引数は scripting mode で評価します。例えば、`:results output`のように記述します。

Type

以下のオプションは相互に排他的で、コードブロックの結果のタイプを返します。デフォルトでは、結果の値によって、テーブルかスカラーのどちらかが結果として挿入されます。

- `table`, `vector` 結果は、Org-mode のテーブルとして解釈されます。単一の値が返された場合、結果は 1 行 1 列のテーブルへと変換されます。例えば、`:results value table`のように記述します。
- `list` 結果は、Org-mode のリストとして解釈されます。単一のスカラー値が返された場合、結果は 1 要素からなるリストへと変換されます。
- `scalar`, `verbatim` 結果は文字通りに評価されます—結果はテーブルへと変換されません。結果は引用されたテキストとして Org-mode のバッファに挿入されます。例えば、`:results value verbatim`と記述します。
- `file` 結果はファイルへのパスとして解釈され、ファイルのリンクとして Org-mode のバッファに挿入されます。例えば、`:results value file`のように記述します。
- `raw`, `org` 結果は加工されていない Org-mode のコードとして解釈され、そのバッファに直接挿入されます。結果がテーブルのように見えるときは、Org-mode のように整形します。例えば、`:results value raw`のように記述します。
- `html` 結果は HTML であると見なされて、`begin_html`ブロックに囲まれます。例えば、`:results value html`のように記述します。
- `latex` 結果は LaTeX と見なされて、`begin_latex`ブロックに囲まれます。例えば、`:results value latex`のように記述します。
- `code` 結果は構文解析可能なコードと見なされて、コードブロックに囲まれます。例えば、`:results value code`のように記述します。
- `pp` 結果は装飾されたコードに変換されて、コードブロックに囲まれます。このオプションは、現在、Emacs Lisp, Python と Ruby をサポートしています。例えば、`:results value pp`のように記述します。
- `wrap` 結果は `begin_result`ブロックに囲まれます。結果の展開方法 (extend?名詞形を見つけられず) が分かっていて自動的に削除されるか置換されるような、`raw`もしくは `org` 構文の結果を挿入する時にこのオプションを使うと便利です。

Handling

以下の result のオプションは、結果が 1 度集められた後に何が起こるかを示しています。

- `silent` 結果はミニバッファに表示されますが、Org-mode のバッファには挿入されません。例えば、`:results output silent`と記述します。
- `replace` デフォルトの設定です。どんな結果でも存在していれば削除し、新しい結果を Org-mode のバッファのその場所に挿入します。例えば、`:results output replace`と記述します。

- append コードブロックの結果が以前から存在する場合、現在存在する結果の最後に新しい結果を付け加えます。結果が存在しない場合は、replaceのように新しい結果を挿入します。
- prepend もしコードブロックの結果が既に存在する場合、現在存在する結果の先頭に新しい結果を追加します。結果が存在しない場合は、replaceのように新しい結果を挿入します。

14.8.2.3 :file

コードブロックの結果を保存する外部ファイルを指定するために、ヘッダー引数:fileを使います。コードブロックの評価の後に、そのファイルへの Org-mode スタイルのリンク[[file:]](4.1 節「Link format」p.36 を参照) が、Org-mode のバッファに挿入されます。R, gnuplot, dot や ditaa などの言語では、ヘッダー引数:fileを扱う特別な機能が用意されています。その機能とは、指定されたファイルへアウトプットを保存するために、必要な定型のコードでコードブロックの本文を囲むというものです。これは、コードブロックの画像出力を特定のファイルに保存する場合に、しばしば役立ちます。

:fileに対する引数は、ファイルのパスを示す文字列、もしくは、次の 2 つの文字列からなるリスト、のどちらかでなくてはなりません。2 つの文字列からなるリストとは、第 1 要素はファイルのパス、第 2 要素はそのリンクの説明から構成されます。

14.8.2.4 :dirとリモートでの実行

ヘッダー引数:fileがアウトプットするファイルのパスを指定する一方で、:dir はコードブロックを実行する間のデフォルトのディレクトリを指定します。指定されない場合、現在のバッファに関連するディレクトリが使われます。つまり、:dir pathを一時的に与えることは、*M-x cd path*でカレントディレクトリを変えるだけで:dirを設定しない場合と同じ効果を持ちます。水面下では、:dirが Emacs の変数 default-directoryを変更しているだけなのです。

:dirを使う場合、アウトプットするファイルを相対パス (例 :file myfile.jpgもしくは:file results/myfile.jpg) で与えなければなりません。その場合、与えたパスはデフォルトディレクトリに対する相対パスとして解釈されます。

つまり、ホームディレクトリ内の 'Work'フォルダでプロット (グラフを描画) したい場合、次のように使えます。

```
#+begin_src R :file myplot.png :dir ~/Work
matplot(matrix(rnorm(100), 10), type="l")
#+end_src
```

リモートでの実行

tramp のファイル構文を使う事でリモートマシン上のディレクトリを指定できます。その場合、リモートマシン上でコードが評価されます。1 つの例です。

```
#+begin_src R :file plot.png :dir /dand@yakuba.princeton.edu:
plot(1:10, main=system("hostname", intern=TRUE))
#+end_src
```

従来通りローカルの Org-mode のバッファにテキストの結果が返されますが、リモートマシン上のリモートディレクトリを基準とした相対パスにファイルのアウトプットが作成されます。リモートのファイルへの Org-mode のリンクが生成されます。

従って、上の例では、リモートマシン上にグラフが生成され、Org-mode バッファには次の形式のリンクが挿入されます。

```
[[file:/scp:dand@yakuba.princeton.edu:/home/dand/plot.png][plot.png]]
```

tramp のおかげで、:dirが Emacs の変数default-directoryの値をセットした後すぐに、この機能を使えます。これらの機能を正しく動かすためには、XEmacs や version23 より前の GNU Emacs を使っている人は tramp を別にインストールする必要があるかもしれません。

更なるポイント

- もし:sessionと並行して:dirが使われるとき、新規セッションを開始する場合には開始ディレクトリを予想して決定します。しかし、すでにセッションが存在する場合には、現バージョンでは、そのセッションと関連したディレクトリを変更しません。
- :exports resultsや:exports bothを使ってファイルをエクスポートする間、通常は:dirを使ってファイルを生成するべきではありません。理由は以下の通りです。2 つのマシンでエクスポートされた結果の移植性を維持するために、エクスポート時にバッファへと挿入されるリンクは default-directoryに対して展開*されていません* 。それゆえ、もし:dirを使って default directoryを変えてしまうと、リンクで意図していない場所にファイルが作られるという恐れがあります。

14.8.2.5 :exports

ヘッダー引数:exportsは、Org-mode ファイルの HTML エクスポートや LaTeX エクスポートに、何を含めるかを指定します。

- code デフォルトの設定です。コードの本文がエクスポートされるファイルに含まれます。例えば、:exports codeのように記述します。
- results コードの評価結果がエクスポートされるファイルに含まれます。例えば、:exports resultsのように記述します。
- both コードと結果の両方がエクスポートされるファイルに含まれます。例えば、:exports bothのように記述します。
- none エクスポートされるファイルには何も含まれません。例えば、:exports noneのように記述します。

14.8.2.6 :tangle

ヘッダー引数:tangleは、ソースコードファイルを抽出したファイルにコードブロックを含めるか否かを指定します。

- tangle コードブロックは、Org-mode のファイルの basename(拡張子を取り除いた名前)にちなんだソースコードファイルにエクスポートされます。例えば、:tangle yesのように記述します。
- no デフォルトの設定です。コードブロックはソースコードファイルにエクスポートされません。例えば、:tangle noのように記述します。
- other ヘッダー引数:tangleに渡された何か他の文字列を、ブロックをエクスポートするファイルの basename と解釈します。例えば、:tangle basenameのように記述します。

14.8.2.7 :mkdirp

抽出 (tangle) されたファイルの親ディレクトリが無いときに、ヘッダー引数:mkdirpを使って親ディレクトリを作成できます。ディレクトリ作成を許可するにはこのヘッダー引数を yes と設定し、ディレクトリ作成を禁止するには noと設定します。

14.8.2.8 :comments

デフォルトの設定では、コードブロックをソースコードへと抽出するときに、コードブロック本文に既に存在するコメントだけを挿入します。抽出されたコードファイルに追加のコメントを挿入するには、ヘッダー引数:commentsを以下のように設定します。

- no デフォルトの設定です。抽出 (tangle) する間に、コメントは追加挿入されません。
- link コードブロックは、コードが抽出された元の Org-mode ファイルへ戻るポインタを含むコメントに囲まれます。
- yes “link” と同じ意味です。これは後方互換性を保持するためです。
- org Org-mode のファイルからのテキストをコメントとして含みます。

  抽出されたコードに先行する文脈からテキストが選ばれます。テキストは、状況に応じて、一番近い見出しもしくはソースブロックに限定されます。
- both コメントのオプションの “link” と “org” との両方を有効にします。
- noweb コメントのオプション “link” を有効にし、noweb の参照記号をコードブロック本文に展開し、それをリンクのコメントで囲みます。

14.8.2.9 :no-expand

デフォルトの設定では、抽出の際に関数org-babel-expand-src-blockによってコードブロックが展開されます。これには、:varを使って変数を割り当てる (14.8.2.1 節「var」p.167 を参照してください) 効果に加えて、“noweb” の参照記号 (14.10 節「Noweb reference syntax」p.179 を参照してください) をそのターゲットで置き換える効果があります。この動作を無効にするためにヘッダー引数:no-expandを使います。

14.8.2.10 :session

ヘッダー引数:sessionは、読み取られた言語のにセッションを開始します。セッションでは、状態は保存されます。

デフォルトでは、セッションは開始されません。

ヘッダー引数:sessionに渡された文字列が、セッション名になります。これにより、読み取られた言語のそれぞれについて並列セッションを実行可能です。

14.8.2.11 :noweb

ヘッダー引数:nowebはコードブロック内の “noweb” スタイル (14.10 節「Noweb reference syntax」p.179 を参照のこと) の参照記号を展開する動作を管理している。このヘッダー引数は yes、noもしくは tangleの 3 つのうち 1 つを選ぶことができる。

- yes 評価、抽出、エクスポートされる前に、コードブロックの本文にある全ての “noweb” 構文の参照記号が展開されます。
- no デフォルトの設定です。コードブロックの評価では “noweb” 構文のアクションはありません。しかし、抽出の間に noweb 参照記号は展開されます。
- tangle ブロックを抽出する前に、コードブロックの本文にある全ての “noweb” 構文の参照記号は展開されますが、ブロックが評価されるときやエクスポートされるときは “noweb” の参照記号は展開されません。

Noweb の行頭部分

現在の Noweb の挿入機能は、<<reference>>の行頭部分の後ろで実行されます。この機能は次の例で説明されます。nowweb の識別記号<<example>>が SQL のコメント構文の後に見つかったため、noweb の参照記号は展開され、それぞれの行がコメントとして挿入されます。

このコードブロックは:

```
-- <<example>>
```

次のように展開されます:

```
-- これは example の
-- 複数行の本文です
```

改行を含まない noweb のテキスト置換 (機能) は、この変化の影響を受けないことに注目してください。したがって、インラインにおいても noweb の識別記号を使えます。

14.8.2.12 :cache

ヘッダー引数:cacheはコードブロックの評価結果をバッファ内に保存するか (=キャッシュするか) を管理します。これにより、変化していないコードブロックを再度評価することを避けられます。このヘッダー引数は:yesあるいは noのどちらかの値をとります。

- no デフォルトの設定です。実行結果をバッファに保存せず、呼び出される度にコードブロックを評価します。
- yes コードブロックが実行される度に、コードの SHA1 ハッシュ値と引数とをブロックへ渡すために生成します。このハッシュ値は#+results:行に格納され、後のコードブロックの実行時にチェックされます。前回評価した時からコードブロックに変化が無い場合、そのコードブロックは改めて評価されません。

コードブロックの引数である変数の値が変化した場合でも、コードブロックのキャッシュは変化として検知します。この場合、キャッシュは無効となり、そのコードブロックは再度実行されます。次の例では、randomの結果が前回実行時から変化しない限り、callerは実行されません。

```
#+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

Org-mode から外部のファイルへと渡す表形式の結果を書く場合、ヘッダー引数:sep を使って区切り文字のを管理できます。この機能は、次の 2 つの方法のどちらかで使用できます。

表形式で書かれたコードブロックの結果を開くときに、コードブロック上で関数 org-open-at-pointを呼び出します。キーバインドは *C-c C-o*です。もう 1 つの方法では、コードブロックの実行結果を外部ファイルへ出力するヘッダー引数を設定します。(14.8.2.3 節「file」p.172 を参照してください。)

:sepが設定されていない場合、デフォルトでは、タブで区切られたテーブルが出力されます。

14.8.2.14 :hlines

テーブルは頻繁に 1 本かそれ以上の水平な線、すなわち hlines(=_h_orizontal _lines_) を使って表現されます。コードブロックの引数:hlinesの値として yesもしくは noを使えます。デフォルトでは noと設定されています。

- no 入力されたテーブルを表示します。ほとんどの言語ではこれが望ましい設定です。なぜなら hlineの記号を定義されていない変数と見なされて、エラーとなるからです。:hlines noと設定するか、デフォルトの設定値を使う場合、次のように出力されます。

```
#+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 テーブル内の水平線を残したままにします。:hlines yesと設定した場合の結果です。

```
#+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

ヘッダー引数:colnamesは、yesと noそして宣言していない場合の nilを値として持てます。デフォルトでは nilに設定されています。

- nil (2 行目が水平線であることから) 入力されたテーブルに列の名前があるように考えられる場合には、処理の前に列の名前が除外されてから、結果の列に名前を与えます。

```
#+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* |
```

  variable indexing 14.8.2.1 節「var」p.167 を参照してください. を使ってテーブルをインデックス化する前に、テーブルの列の名前が取り除かれないことに注意してください。

- no 列の名前に関する前処理は行われません。
- yes テーブルが列の名前を持っている「ように見えない」場合でも (すなわち 2 行目が水平線でない場合でも)、nilを設定されたように、列の名前を除外し、改めて結果の列に名前を割り振ります。

14.8.2.16 :rownames

ヘッダー引数:rownamesは yesもしくは noを値として持つ事ができ、デフォルトでは noと設定されています。

- no 行に関する前処理を行いません。
- yes 前処理としてテーブルの 1 列目を取り除き、処理結果に列の名前を付け加えます。

```
#+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 |
```

variable indexing 14.8.2.1 節「var」p.167 を参照してください. を使ってテーブルをインデックス化する前に、テーブルの行の名前が取り除かれないことに注意してください。

14.8.2.17 :shebang

ヘッダー引数:shebangに文字列の値 (例えば `:shebang "#!/bin/bash"`) を設定することで、コードブロックを抽出した全てのファイルの 1 行目にその文字列を挿入し、抽出されたファイルのパーミッションは実行可能と設定されます。

14.8.2.18 :eval

ヘッダー引数:evalを使って、特定のコードブロックの評価を制限できます。:evalは “never” と “query” の 2 つの値をとれます。`:eval never`はコードブロックが評価されないことを確かにします。 これは危険なコードブロックを評価することを防ぐのに便利です。`:eval query`と設定すると、変数 org-confirm-babel-evaluateの値に関係なく、コードブロックを実行する度に確認のダイアログがでます。

14.9 評価の結果

結果の取り扱われ方は、セッションが呼び出されたかに依存することに加えて、`:results value`もしくは`:results output`が使われているかに依存します。(この一文分からず) 次のテーブルは起こりうることを表形式にまとめました。ヘッダー引数 results が取り得る完全なリストは、14.8.2.2 節「results」p.170 を見て下さい。

	Non-session	**Session**
`:results value`	最後の式の値	最後の式の値
`:results output`	STDOUT の内容	インタプリタの出力をつなげたもの

以下に注意すること::`results value`と設定することで、`:session`と non-session の両方の結果は Org-mode にテーブル (文字列もしくは数字からなる 1 次元もしくは 2 次元のベクトル) を適切に返されます。

14.9.1 Non-session

14.9.1.1 :results value

デフォルトの設定です。外部の言語で書かれた関数定義のコードを囲むことと、内部でその関数を評価することによってその値が得られます。それゆえ、コードはその関数の本文でであるかのように書かなければなりません。特に、Python は `return`文が存在しない限り自動的に関数から返り値を返しませんので、Python では多くの場合に ‘`return`’文が必要なことに注意してください。

コンテキストを評価する 4 つの方法の中で、これは関数定義にコードが自動的に囲まれるただ一つの方法です。

14.9.1.2 :results output

外部プロセスとしてコードはインタープリタに渡され、標準出力のストリームの内容がテキストとして返されます。(いくつかの言語ではエラー出力のストリームも含まれます; これは今後の課題です。)

14.9.2 Session

14.9.2.1 :results value

対話式の Emacs の下位プロセスとして実行しているインタープリタ (解釈プログラム) にコードが渡されます。そのインタープリタが最後に評価した結果を、結果として受け取ります。(これは言語固有の方法で得られています:Python や Ruby での変数_の値と、R での.Last.valueの値です)。

14.9.2.2 :results output

対話式の Emacs の下位プロセスとして実行しているインタープリタ (解釈プログラム) にコードが渡されます。返される結果は、そのインタープリタからの一連の (テキストの) アウトプットを結合したものです。もし外部プロセスとして実行している対話式でないインタープリタに同じコードが送られるなら、STDOUTへと送られるであろうものと全く同じである必要はかならずしも無いことに気づいて下さい。例として、次の 2 つのブロックを比べて下さい:

```
#+begin_src python :results output
 print "hello"
 2
 print "bye"
#+end_src

#+resname:
: hello
: bye
```

non-session モードでは、'2' はプリントされず、表示されません。

```
#+begin_src python :results output :session
 print "hello"
 2
 print "bye"
#+end_src

#+resname:
: hello
: 2
: bye
```

しかし:sessionモードでは、対話的なインタープリタは入力 '2' を受け取り、その値 '2' を表示します。(実際には、ここでは他の print 文は不要でした)。

14.10 Noweb reference syntax

"noweb" (http://www.cs.tufts.edu/~nr/noweb/を参照のこと) による文芸的プログラミングのシステムは、名付けられたコードブロックがよく知られた Noweb の構文を使って参照されることを許しています。:

```
<<code-block-name>>
```

コードブロックが抽出されるか評価されるとき、"noweb" 参照記号を展開するかしないかは、ヘッダー引数:nowebの値が決めます。もし:noweb yesなら、Noweb の参照記号が展開さ

れてから評価が始ます。デフォルトの設定ですが、もし:noweb noなら、参照記号は展開されずに評価が始まります。

注意:ある言語内で正しいコードが破壊されないように、デフォルトの値を:noweb noと設定しています。例えば、<<arg>>が構文的に有効な構成要素である Ruby などの言語を想定しています。もしあなたが使っている言語で<<arg>>が構文的に有効な構成要素でない場合、デフォルトの値を使用してください。

14.11 Key bindings and useful functions

多くの一般の Org-mode のキー操作がコンテキストに応じてキーバインドされた。

コードブロックの中でも、以下のキーバインドは有効です。

```
C-c C-c              org-babel-execute-src-block
C-c C-o              org-babel-open-src-block-result
C-up                 org-babel-load-in-session
M-down               org-babel-pop-to-session
```

Org-mode のバッファ内では、以下のキーバインドが有効になっています。

```
C-c C-v a  もしくは  C-c C-v C-a      org-babel-sha1-hash
C-c C-v b  もしくは  C-c C-v C-b      org-babel-execute-buffer
C-c C-v f  もしくは  C-c C-v C-f      org-babel-tangle-file
C-c C-v g                             org-babel-goto-named-source-block
C-c C-v h                             org-babel-describe-bindings
C-c C-v l  もしくは  C-c C-v C-l      org-babel-lob-ingest
C-c C-v p  もしくは  C-c C-v C-p      org-babel-expand-src-block
C-c C-v s  もしくは  C-c C-v C-s      org-babel-execute-subtree
C-c C-v t  もしくは  C-c C-v C-t      org-babel-tangle
C-c C-v z  もしくは  C-c C-v C-z      org-babel-switch-to-session
```

14.12 バッチ処理

コマンドラインから関数を呼び出すことができます。このシェルスクリプトは、引数のそれぞれに対して org-babel-tangleを呼び出します。

必ずあなたのシステムに合うようにパスを設定してください。

```
#!/bin/sh
# -*- mode: shell-script -*-
#
# org-mode でファイルを抽出する
#
DIR=`pwd`
FILES=""
ORGINSTALL="~/src/org/lisp/org-install.el"

# 関数 tangle を呼び出すためのコードでそれぞれの引数を囲む
for i in $@; do
    FILES="$FILES \"$i\""
```

```
done

emacs -Q --batch -l $ORGINSTALL \
--eval "(progn
(add-to-list 'load-path (expand-file-name \"~/src/org/lisp/\"))
(add-to-list 'load-path (expand-file-name \"~/src/org/contrib/lisp/\"))
(require 'org)(require 'org-exp)(require 'ob)(require 'ob-tangle)
(mapc (lambda (file)
       (find-file (expand-file-name file \"$DIR\"))
       (org-babel-tangle)
       (kill-buffer)) '($FILES)))" 2>&1 |grep tangled
```

15 Miscellaneous

15.1 Completion

Emacs は補完無しでは Emacs とはいえません、そして org-mode はそれが意味をなすたびに使用します。もしあなたが *iswitchb-* か *ido-*のようなインタフェースを補完のプロンプトとして好むのであれば、あなたは `org-completion-use-iswitchb`や `org-completion-use-ido` 変数のいずれかを設定することで指定することができます。

Org-mode はバッファ中の補完をサポートします。この種類の補完はミニバッファを活用します。あなたは簡単に数文字をバッファに入力し、補完キーを補完するテキストの右側で押します。

M-TAB　　ポイント位置での補完

- 見出しの先頭では、TODO キーワードを補完します。
- '\'の後では、エクスポート機能によりサポートされる TEX のシンボルを補完します。
- '*'の後では、'`[[*find this headline]]`'のようにリンクを検索できるように、カレントバッファで見出しを補完します。
- 見出し中の '`:`'の後では、タグを補完します。タグのリストは `org-tag-alist` 変数 (6.2 節「Setting tags」p.57 を参照より、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 (4.6 節「Link abbreviations」p.41 を参照)。
- '`+`'の後では、Org-mode 向けのファイル固有の設定としてセットする '`TYP_TODO`'や '`OPTIONS`'のようなスペシャルキーワードを補完します。オプションキーワードが既に補完されているなら、*M-TAB*を再び押すことでこのキーワードの設定の例を挿入します。
- '`#+STARTUP:` 'の後の行の中では、STARTUP キーワードを補完します、すなわち、はこの行では正しいキーです。
- 他の場所では、Ispell を用いた辞書補完が行われます。

15.2 Easy Templates

Org-mode は僅かなキーストロークのみによる空の構造の (`#+BEGIN_SRC`や`#+END_SRC`のような) 要素の挿入をサポートします。これはネイティブなテンプレート拡張機構を通じて得られるものです。ここで留意すべきこととして、Emacs は例えば '`yasnippet`'のような同じように使うことができるいくつかの他のテンプレート機構を持ちます。

構造要素の挿入には、'`<`'をタイプし、続いてテンプレートセレクタと `TAB`をタイプします。補完は上記のキーストロークが単独で行に入力されている場合のみ働きます。

以下のテンプレートセレクタが現在サポートされています。

s　　`#+begin_src ... #+end_src`

```
e         #+begin_example ... #+end_example
q         #+begin_quote ... #+end_quote
v         #+begin_verse ... #+end_verse
c         #+begin_center ... #+end_center
l         #+begin_latex ... #+end_latex
L         #+latex:
h         #+begin_html ... #+end_html
H         #+html:
a         #+begin_ascii ... #+end_ascii
A         #+ascii:
i         #+include: line
```

例えば、空の行で"<e"と入力し、その後 TAB を入力すると、EXAMPLE テンプレートが補完されます。

あなたは org-structure-template-list変数をカスタマイズすることで追加のテンプレートをインストールすることができます。詳細は変数の docstring を参照してください。

15.3 Speed keys

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-commans-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ー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`　[オプション]
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 (4.3 節「External links」p.37 を参照) 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えます。

`org-confirm-shell-link-function`　[オプション]
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`　[オプション]
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 (3.5 節「The spreadsheet」p.25 を参照) 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 (15.6 節「In-buffer settings」p.184 を参照) 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*を押しれください。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。4.6 節「Link abbreviations」p.41 を参照してください. 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-idented[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	donflt 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-inine-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	donflt 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、ノートをを記録します
nologrepeat	アイテムが再開されたとき、記録を行わいません
lognoteclock-out	時間測定が終了したとき、ノートを記録します
nologlognoteclock-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

[1] Emacs22 と Org-mode6.29 が必要です

`lognoteredeadline`　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

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。

`hidestars`　全てのヘッドラインの「*」を見えなくします
`showstars`　全てのヘッドラインの「*」を見えるようにします
`indent`　virtual indentation according to outline level
`noindent`　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
`odd`　奇数のアウトラインレベル (1、3……) のみ
`oddeven`　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。

`constcgs`　‘`constants.el`’ should use the c-g-s unit system
`constSI`　‘`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。

`fninline`　脚注のインラインを定義します
`fnnoinline`　分割されたセクションでの脚注を定義します
`fnlocal`　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
`fnprompt`　脚注ラベルのプロンプト
`fnauto`　`[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)
`fnconfirm`　編集と確認用の自動ラベルを用意します
`fnplain`　`[1]`のようなラベルを自動的に作成します
`fnadjust`　自動的に脚注の番号を振り直し、ソートします
`nofnadjust`　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。

`hideblocks`　開始時に全ての開始/終了ブロックを見えなくします
`nohideblocks` 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。

`entitiespretty`　可能な時、UTF-8 の文字を表示します
`entitiesplain`　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 12.2 節「Export options」p.132 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か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 (第 6 章「Tags」p.57 を参照)。他の多くの状況では、“ここを見て、見たものに応じて更新する” 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 の見出しについている「*」の数が増えたときや見出しの下の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 Emacs23.2[2] 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 2[4] 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. 見出しの下の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の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. 先頭の「*」を隠す
   あなたは先頭の「*」を非表示にするという方法で変更することができます。これを行うグローバルな方法は、org-hide-leading-stars変数を設定するか、以下のようにファイル毎に設定するかです。

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

---

[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

「*」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
S-TAB	*C-u TAB*	*C*	
M-left	*C-c C-x l*	*l*	*Esc left*
M-S-left	*C-c C-x L*	*L*	
M-right	*C-c C-x r*	*r*	*Esc right*
M-S-right	*C-c C-x R*	*R*	
M-up	*C-c C-x u*		*Esc up*
M-S-up	*C-c C-x U*	*U*	
M-down	*C-c C-x d*		*Esc down*

[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

M-S-down	*C-c C-x D*	*D*	
S-RET	*C-c C-x c*		
M-RET	*C-c C-x m*		*Esc RET*
M-S-RET	*C-c C-x M*		
S-left	*C-c left*		
S-right	*C-c right*		
S-up	*C-c up*		
S-down	*C-c down*		
C-S-left	*C-c C-x left*		
C-S-right	*C-c C-x right*		

15.10 他のパッケージとの関係

Org-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 Org-mode と強調して動くパッケージ

‘`calc.el`’ by Dave Gillespie
: Org-mode はテーブル (3.5 節「The spreadsheet」p.25 を参照) 中の表計算関数の実装に Calc パッケージを使います。Org-mode は Calc が適切にインストールされている場合、設定中に自動で読み込まれる `calc-eval`関数を探し、Calc が利用できることを確認します。Emacs22 現在で、Calc は Emacs に最初から組込まれています。2 つのパッケージの連携の別の方法は Calc を組込み計算に使うことです。*GNU Emacs Calc Manual* の “Embedded Mode” 節 を参照してください.

‘`constants.el`’ by Carsten Dominik
: テーブル関数中 (3.5 節「The spreadsheet」p.25 を参照) で、自然定数や単位に名前を使うことができるようになります。あなたが自分で `org-table-formula-constants`変数に定数を定義する代わりに、多くの定数や単位を定義している ‘`constants`’パッケージをインストールすることで、あなたは ‘`Mega`’に ‘`M`’のような表現を使うことができるようになります。このパッケージのバージョン 2.0 が必要で、`http://www.astro.uva.nl/~dominik/Tools`から利用できます。Org-mode は設定中に自動で読み込まれる `constants-get`関数をチェックします。‘`constants.el`’のインストール説明を参照してください。

‘`cdlatex.el`’ by Carsten Dominik
: Org-mode は LaTeX フラグメントを Org-mode ファイルに効率的に入力するために CDLaTeX パッケージを活用できます。11.7.5 節「CDLaTeX mode」p.130 を参照してください。

‘`imenu.el`’ by Ake Stenhoff and Lars Lindberg
: Imenu はファイル中のアイテムのインデックスへのアクセスメニューを提供します。Org-mode は Imenu をサポートします—インデックスを得るために、あなたは以下のようにする必要があります:

```
(add-hook 'org-mode-hook
          (lambda () (imenu-add-to-menubar "Imenu")))
```

: デフォルトではインデックスは 2 レベルの深さです—あなたは `org-imenu-depth` オプションを用いることで深さを変更できます。

‘remember.el’ by John Wiegley
: Org-mode はこのパッケージをキャプチャに使用します、しかし、もはやそうではありません。

‘speedbar.el’ by Eric M. Ludlam
: Speedbar はファイルとファイル中のインデックスを表示するためのスペシャルフレームを作成するパッケージです。Org-mode は Speedbar をサポートし、Speedbar から Org-mode ファイルへ直接繋げます。Speedbar フレームで<コマンドを使うことでファイルまたはサブツリーへのアジェンダコマンドの範囲を制限します。

‘table.el’ by Takaaki Ota
: 自動的な行の折り返し、列、行の広がり、調整を伴なう複雑な ASCII テーブルは Ota Takaaki(http://sourceforge.net/projects/table、もしくは Emacs22 に含まれています) による Emacs のテーブルパッケージを用いることで作成可能です。Org-mod はこれらのテーブルを認識し、適切にセクスポートします。Org-mode の別の機能による干渉のために、あなたは不幸にもこれらのテーブルをバッファ中で直接編集することができません。代わりに、あなたはこのテーブルの編集のためにソースコードスニペットに似た *C-c '* コマンドを使う必要があります。

  C-c '　org-edit-special
  : ‘table.el’のテーブルを編集します。カーソルが table.el のテーブル上の時動作します。

  C-c ~　org-table-create-with-table.el
  : ‘table.el’のテーブルを挿入します。ポイント位置が既にテーブルなら、このコマンドは‘table.el’のフォーマットと Org-mode のフォーマットで相互変換します。これが可能なことと制限については、org-convert-table コマンドのドキュメントを参照してください。

‘table.el’は Emacs22 以降の Emacs では内蔵されています。

‘footnote.el’ by Steven L. Baur
: Org-mode はこのパッケージが提供する数字の脚注を認識します。しかしながら、Org-mode は自身の脚注 (2.10 節「Footnotes」p.17 を参照) もサポートしているため、‘footnote.el’を使う必要はありません。

15.10.2 Org-mode との衝突に繋がるパッケージ

Emacs23 では、Shift キーと組み合わせたカーソルの動きを開始するかリージョンを広げるための shift-selection-mode がデフォルトで有効になっています。カーソルがそのような位置にある場合、Org-mode でのタイムスタンプ、TODO キーワード、プライオリティ、アイテム bullet タイプの変更と *S-cursor* コマンドは衝突します。デフォルトでは、*S-cursor* コマンドは特別なコンテクスト以外では何も起きませんが、org-support-shift-select 変数をカスタマイズすることができます。Org-mode はスペシャルコマンドが適用される特別なコンテクストの外で (i) 使用することにより Shift 選択を提供しようとし、また (ii) アクティブなリージョンを拡張することによっても特別なコンテクストを通してカーソルが移動します。

‘CUA.el’ by Kim. F. Storm
リージョンの選択と拡張について、Org-mode でのキーバインディングは (pc-select-modeや s-region-modeと同様に)CUA モードで使われる *S-<cursor>*と衝突します。実際、前の段落を見れば分かりますが、Emacs23 は shift-selection-modeの形でこのビルトインを持ちますます。あなたが Emacs23 を使っているのであれば、まず間違いなくこの目的のための別のパッケージは使いたくないでしょう。しかし、Org-mode での作業中に別のパッケージにこれらのキーを渡すことを選ぶのであれば、org-replace-disputed-keys 変数を設定してください。設定したとき、Org-mode は Org-mode ファイルとアジェンダバッファ(日付の選択を除きます) 中で以下のキーバインディングを変えるでしょう。

S-UP	⇒	M-p	S-DOWN	⇒	M-n
S-LEFT	⇒	M--	S-RIGHT	⇒	M-+
C-S-LEFT	⇒	M-S--	C-S-RIGHT	⇒	M-S-+

はい、残念ながら覚えることがより困難です。もしあなたが他の代わりのキーを持ちたいのであえれば、org-disputed-keys変数を見てください。

‘yasnippet.el’
Org-mode は TAB キー ("\t"の代わりに。[tab]をバインドします) をバインドするこのキーで YASnippets のアクセスを優先します。以下のコードはこの問題を修正します:

```
(add-hook 'org-mode-hook
          (lambda ()
            (org-set-local 'yas/trigger-key [tab])
            (define-key yas/keymap [tab] 'yas/next-field-group)))
```

yasnippet の最新のバージョンは Org-mode と相性がよくありません。上記のコードが衝突を修正しないなら、以下の関数を定義してください:

```
(defun yas/org-very-safe-expand ()
       (let ((yas/fallback-behavior 'return-nil)) (yas/expand)))
```

それから、Org-mode に実行すべき新しい関数を教えてくさい:

```
(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
このパッケージも、*S-<cursor>*キーを使用し。そして CUA モードが適用されている状態の段落で全てが書かれます。もし、Org-mode が *S-cursor*上に特別な関数を持たない場所であなたが windmove 関数を有効にしたいのであれば、設定に以下を追加します:

```
;; Make windmove work in org-mode:
(add-hook 'org-shiftup-final-hook 'windmove-up)
(add-hook 'org-shiftleft-final-hook 'windmove-left)
(add-hook 'org-shiftdown-final-hook 'windmove-down)
(add-hook 'org-shiftright-final-hook 'windmove-right)
```

‘viper.el’ by Michael Kifer
: Viper は *C-c /*を使い、それ故に Org-mode の org-sparse-treeコマンドに対応しているキーを使えないようにします。あなたはこのコマンドに別のキーを割り当てるか、viper-vi-global-user-mapでキーを上書きする必要があります:

```
(define-key viper-vi-global-user-map "C-c /" 'org-sparse-tree)
```

付録 A Hacking

この付録では、ユーザーが Org-mode の機能を拡張できる幾つかの特徴を紹介します。

A.1 Hooks

Org-mode には、機能を追加するためのフック変数が数多くあります。ハッキングに関するこの付録では、それらの使い方を説明します。全フックの完全なリストと説明文は Worg project によって維持管理されており、`http://orgmode.org/worg/org-configs/org-hooks.php` にて見つけられます。

A.2 Add-on packages

様々な人によって膨大な数のアドオンパッケージが書かれました。これらのパッケージは Emacs の一部ではありませんが、投稿されたパッケージとして独自のリリースで配布されており、Org-mode のホームページ `http://orgmode.org`にて入手できます。投稿されたパッケージのリストとその説明文は Worg project によって `http://orgmode.org/worg/org-contrib/`にて維持管理されています。

A.3 Adding hyperlink types

Org-mode にはビルトインのハイパーリンク形式が多数用意されています (第 4 章「Hyperlinks」p.36 を参照)。新しいタイプのリンクを追加したい場合、Org は追加するためのインターフェスを用意します。例となるファイル ‘`org-man.el`’を見て下さい。これは Unix の man ページを Emacs で表示するために、‘`[[man:printf][printf の man ページ]]`’ というリンクを生成するサポートを作成します:

```
;;; org-man.el - Org-mode で man ページのリンクをサポートする

(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
  "man のページを表示するための Emacs のコマンド。"
  :group 'org-link
  :type '(choice (const man) (const woman)))

(defun org-man-open (path)
  "パス (PATH) にある manpage を開きます。
パスはコマンド man に渡せる内容でなければなりません。"
  (funcall org-man-command path))

(defun org-man-store-link ()
  "man ページへのリンクを保存します。"
  (when (memq major-mode '(Man-mode woman-mode))
    ;; これは man ページなので、リンクを作成します。
    (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 ()
  "バッファ名からページ名を抽出します。"
  ;; `Man-mode' と `woman-mode' の両方で動作します。
  (if (string-match " \\(\\S-+\\)\\*" (buffer-name))
      (match-string 1 (buffer-name))
    (error "この man ページへのリンクを作成できません")))

(provide 'org-man)

;;; org-man.el ends here
```

以下を‘.emacs’に加えて、この新しいタイプのリンクを有効にします。

```
(require 'org-man)
```

そのファイルを一つずつ順番に検討して、何をするのか見てみましょう。

1. (require 'org)を実行して‘org.el’が読み込まれたことを確認します。
2. 次の行ではorg-add-link-typeを呼び出して、新しいタイプのリンクを接頭辞‘man’で定義します。またこの呼び出しでは、そのようなリンクを辿るために呼び出される関数の名前も含まれています。
3. 次の行では、man ページを表示したバッファ内にて有用なリンクを *C-c l*で保存できるように、関数を org-store-link-functionsに追加します。

ファイルの残りは必要な変数と関数を定義しています。最初に、man ページの表示にどの Emacs のコマンドを使うかを決める変数のカスタマイズがあります。2 つのオプション、man と woman があります。次に、リンクを辿る関数が定義されています。リンクのパスは引数として渡されます — この場合、リンクのパスは単に man コマンドのトピックです。この関数は man ページの表示に org-man-commandの値を呼び出します。

最後に、関数 org-man-store-linkが定義されています。*C-c l*でリンクを保存する時に、リンクを作成するためにこの関数が呼び出されます。関数は、このバッファのタイプでリンク作成がサポートされているかどうかを最初に判断します; 変数 major-modeの値をチェックして判断します。サポートされていない場合、関数は終了して、nilの値を返します。サポートされている場合、バッファ名から man のトピックを取得して、文字列‘man:’の後に繋げることで、リンクを生成します。次に、コマンド org-store-link-props を呼び出して、:type と:linkプロパティを設定します。任意で:descriptiion プロパティを設定できます。これは、後に *C-c C-l*で Org-mode のバッファにリンクが挿入される時のリンクの説明のデフォルト値を用意します。

新しいタイプのリンクを正しく設定できると、*C-c C-l*でリンクを挿入するときの特別な機能 (例えば 補完機能) のサポートを実装する関数 org-PREFIX-complete-linkを定義できます。そのような関数は引数をとらず、接頭辞がついたリンク全体を返します。

A.4 Context-sensitive commands

Org-mode では、文脈に依存して動作が変わるコマンドが幾つかあります。最も重要な例は *C-c C-c*(15.7 節「The very busy C-c C-c key」p.188 を参照) です [typo?: The most important example +it+ *is* the *C-c C-c* (15.7 節「The very busy C-c C-c key」p.188 を参照). また、*M-cursor* と *M-S-cursor* キーもこの性質を持っています。

アドオンにとって特別な文脈を検知し、文脈に沿って適切に実行する関数を準備することによって、アドオンはこの機能を活用できます。ここに例として Dan Davison の 'org-R.el' があります。このファイルは 'R' プログラミング言語[1] に基づいてコマンドを実行できます。このパッケージでは、特別な文脈とは#+R: や #+RR:で始まる行のことです。

```
(defun org-R-apply-maybe ()
  "org-R の文脈なら検知して、R のコマンドを実行します。"
  (if (save-excursion
        (beginning-of-line 1)
        (looking-at "#\\+RR?:"))
      (progn (call-interactively 'org-R-apply)
             t) ;; 処置したと信号を送ります
    nil)) ;; 処置しなかったと信号を送ります

(add-hook 'org-ctrl-c-ctrl-c-hook 'org-R-apply-maybe)
```

関数は最初にそのような行にカーソルがあるのかをチェックします。そうである場合、org-R-applyが呼び出されて関数は処置したと信号を送るためにtを返し、そして *C-c C-c*は他の文脈を探すことを止めます。もし関数がそこですべきことが無いと分かった場合は、他の同じような関数が試せるようにnilを返します。

A.5 任意のシンタックスによる表やリスト

Orgtbl モードを任意のバッファでマイナーモードとして使えるため、LaTeX のような特定言語のテーブルでモードを動作させる機能の要望がよく出されます。しかし、通常の方法でこれを実装するのは極めて難しく、結局悪夢のような設定となってしまい、Orgtbl モードのテーブルエディタによって得られる単純さのほとんどを失ってしまうでしょう。

この付録では、別の方法を説明します。Orgtbl モード本来のフォーマットでテーブルを保ち (もとのテーブル)、その表を正しいシンタックスへと変換するカスタム関数を使い、(目的のテーブルを) 正しい場所に配置します。これはユーザーに変換する関数を書く負担をかけますが、まさにフレキシブルな方法が可能となります。

Bastien はリストに対して同じ機能を Orgstruct モードに加えました。リストを編集したり構造化する Org-mode の機能を使うには、orgstruct-modeをオンにして、そこで当該リストを別のフォーマット (HTML, LaTeX もしくは Texinfo) へとエクスポートしてください。

A.5.1 Radio tables

ターゲットテーブルの場所を定義するには、Orgtbl モードが見つけ出せるマジックワード含んだ 2 行のコメントを最初にカレントモード内で作成します。Orgtbl モードはその 2 行の間に変換されたテーブルを挿入します。前に何が在ったかは考慮せずに置き換えます。例えば:

[1] 'org-R.el'は第 14 章「Working With Source Code」p.160 で説明された Org-mode の機能によって置き換えられ、現在使われておりません。

```
/* BEGIN RECEIVE ORGTBL table_name */
/* END RECEIVE ORGTBL table_name */
```

ソーステーブル (元のテーブル) のすぐ上に、変換する方法と挿入する場所を Orgtbl モードに知らせる特別な行を付け加えます。例えば:

```
#+ORGTBL: SEND table_name translation_function arguments....
```

`table_name`はそのテーブルの参照名で、それは変換結果を挿入する行でも使われます。`translation_function`は変換する Lisp の関数です。加えて、その行の最後では、(key と value を交互においた) 引数のリストを含むことができます。引数はプロパティのリストとして解釈するために、変換する関数 (translation function) に渡されます。変換する関数 (translation function) が呼び出される前に、いくつかの基本的なパラメータは既に認識され、影響を受けています。

`:skip N`　テーブルの最初の N 行を飛ばして読み進みます。このパラメータは、行を分ける横線をカウントします。

`:skipcols (n1 n2 ...)`
飛ばすべき列のリストです。もしもテーブルに計算用にマークされた列がある場合、同じようにに自動的に無視されます。列を除去した後で変換する関数がテーブルを見るので、そこには更に列があったとは知る由もないことに注意して下さい。

残る 1 つの問題は、ファイルの通常動作を妨げずに、バッファ内のソーステーブルを保つ方法です。例えば、C 言語のファイルや LaTeX ファイルを編集している場合です。いくつかの方法があります:

- その言語でサポートされたブロックコメント内にテーブルを設置できます。例えば C モードの場合、テーブルを '`/*`'と '`*/`'の行で括れます。
- 時には、TeX の '`\bye`'や LaTeX の '`\end{document}`'のような、ある種の終了文の後にテーブルを書けることがあります。
- 単純に、ファイルを編集したいときはテーブルを一行一行コメントにし、テーブルを編集したいときはコメントを外すこともできます。これは退屈そうに聞こえますが—コマンド *M-x orgtbl-toggle-comment*を使えばコメントのオンオフを簡単にできます。キーバインドを設定すると特に便利です。

A.5.2 LaTeX でのラジオテーブルの例

LaTeX でソーステーブルを囲む最良の方法は、'`comment.sty`'で提供される `comment` 環境を使うことです。文書のヘッダーに`\usepackage{comment}`と書くことで有効になります。Orgtbl モードで *M-x orgtbl-insert-radio-table*とすると、テーブルのスケルトンを挿入できます[2]。テーブル名を確認されますので、'`salesfigures`'と入力します。 そして以下のテンプレートを得ました:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| | |
```

[2] デフォルトでは、LaTeX, HTML と Texinfo の場合だけ動作します。他のモードのテンプレートをインストールするには、変数 `orgtbl-radio-tables`を編集して下さい。

```
\end{comment}
```

#+ORGTBL: SENDの行は、関数 orgtbl-to-latexを使ってテーブルを LaTeX へ変換し、salesfiguresの名前で受け取る場所に設置するようにと、Orgtbl モードに指示しています。これからテーブルに記入します—スプレッドシートの機能を自由に使ってください[3]:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| 月  | 日数 | 販売数 | 一日あたり |
|-----+------+--------+------------|
| 1月 |   23 |     55 |        2.4 |
| 2月 |   21 |     16 |        0.8 |
| 3月 |   22 |    278 |       12.6 |
#+TBLFM: $4=$3/$2;%.1f
% $ (font-lock 色分けをうまく動作させるための余分なドル記号、脚注参照)
\end{comment}
```

入力し終わり、テーブルの中で *C-c C-c*と押下すると、目印となる 2 行の間に変換されたテーブルが挿入されます。

さて、列の文字揃えを制御したいなどの理由から、テーブルのヘッダーを書きたいとします。この場合は次のことを確認します。テーブルを変換する関数がソーステーブルの最初の 2 行を飛ばすことと、接合部として動作すること、すなわち ターゲットテーブルのヘッダーとフッターを生成しないことです:

```
\begin{tabular}{lrrr}
月 & \multicolumn{1}{c}{日数} & 販売数 & 一日あたり \\
% BEGIN RECEIVE ORGTBL salesfigures
1月 & 23 & 55 & 2.4 \\
2月 & 21 & 16 & 0.8 \\
3月 & 22 & 278 & 12.6 \\
% END RECEIVE ORGTBL salesfigures
\end{tabular}
%
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex :splice t :skip 2
| 月  | 日数 | 販売数 | 1日あたり |
|-----+------+--------+-----------|
| 1月 |   23 |     55 |       2.4 |
| 2月 |   21 |     16 |       0.8 |
| 3月 |   22 |    278 |      12.6 |
#+TBLFM: $4=$3/$2;%.1f
\end{comment}
```

[3] '#+TBLFM'行に奇数個のドル記号 (dollar character, $) が在る場合、LaTeX モードの色分けで問題が生じるかも知れません。この問題を解決するには、例で示すように commentを環境内に行を追加してドル記号の表現を調整してください。AUCTeX を font-latex ライブラリと使っている場合のはるかに優れた解決法は、変数LaTeX-verbatim-environmentsに commentを環境追加することです。

LaTeX へ変換する関数 orgtbl-to-latexは既に Orgtbl モードの一部です。tabular 環境を使ってテーブルをタイプセットし、横線に\hlineで印を付けます。加えて、以下のパラメータを解釈します (“char 65.5.3 節「Translator functions」p.200 を参照も参照してください):

:splice nil/t
: t のとき、tabular 環境で囲まずに、テーブルの本文だけを返します。デフォルトでは nil です。

:fmt fmt
: 各フィールドを囲むのに使われるフォーマットで、%sにはフィールドの元の値が入っています。例えばドルマークで囲む場合には、:fmt "$%s$"と書けます。また、列の番号とフォーマットというプロパティのリストにもなります。例えば:fmt (2 "$%s$" 4 "%s\\%%")と書けます。文字列の変わりに、1 つの引数を持つ関数を使えます; その関数は書式設定された文字列を返します。

:efmt efmt
: 指数表記で数値を出力する時に使います。仮数部と指数部を挿入するため、この形式では%sが 2 回出現します。例えば"%s\\times10^{%s}"。デフォルトでは"%s\\,(%s)"です。これも列数と書式設定のリスト、例えば:efmt (2 "$%s\\times10^{%s}$" 4 "$%s\\cdot10^{%s}$")となり得ます。値に efmtが適用された後に、fmtも適用されます。fmtと同様に、文字列の代わりに引数 2 つの関数を使えます。

A.5.3 Translator functions

Orgtbl モードには幾つかの組み込み変換関数があります: orgtbl-to-csv (カンマ区切り), orgtbl-to-tsv (タブ区切り) orgtbl-to-latex, orgtbl-to-html そして orgtbl-to-texinfoです。orgtbl-to-html[4] を除いた、これら全ての関数は包括的な変換関数 orgtbl-to-genericを使います。例えば、orgtbl-to-latexそれ自体はとても短い関数で tabular環境の列の定義を計算したり、幾つかのフィールドセパレータや行セパレータを定義してそれから処理を包括的な変換関数へと渡しています。ここに完全なコードがあります:

```
(defun orgtbl-to-latex (table params)
  "Orgtbl モードのテーブルを LaTeX に変換します。"
  (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))))
```

ご覧のように、関数に渡されたプロパティ(変数 *PARAMS*) が、関数の中で新しく定義されたプロパティ(変数 *PARAMS2*) と結合されています。関数に渡されたもの (すなわち ‘ORGTBL SEND’行でセットされたプロパティ) が優先されます。よって、もし LaTeX 変換関数を使いたけれども、行の最後はデフォルトの ‘\\’の代わりに ‘\\[2mm]’としたいなら、デフォルトの設定をこのように上書きできます

[4] HTML の変換関数は、HTML をエクスポートする時にテーブルを作るコードと全く同じものを使っています。

```
#+ORGTBL: SEND test orgtbl-to-latex :lend " \\\\[2mm]"
```

新しい言語に対して、LaTeX 変換関数とのアナロジーから自分で変換関数を書くか、もしくは、包括的な変換関数を直接使うこともできます。例えば、もしある言語では、テーブルが始まるときは '!BTBL!'、終わるときには '!ETBL!'、そしてテーブル行は '!BL!'で始まり、'!EL!'で終わり、フィールドセパレータはタブだとしたら、このように (1 行だけで!) 包括的な変換関数を呼び出すことができます:

```
#+ORGTBL: SEND test orgtbl-to-generic :tstart "!BTBL!" :tend "!ETBL!"
                                      :lstart "!BL! " :lend " !EL!" :sep "\t"
```

関数によって理解されるパラメータの完全なリストのために、関数 orgtbl-to-genericの説明文をチェックしてください。そして、それらの 1 つずつを orgtbl-to-latex、orgtbl-to-texinfoや、包括的な変換関数を使った何か他の関数に渡せることを覚えて下さい。

もちろん包括的な変換関数では行えない複雑な事を処理する全く新しい関数を書くこともできます。変換関数は 2 つの引数を持ちます。1 つ目の引数はテーブル、行のリスト、それぞれの行のシンボルコード hline もしくはフィールドのリストです。2 つめの引数はプロパティリストで、'#+ORGTBL: SEND'行で指定された全てのパラメータを含みます。関数は書式設定されたテーブルを含む 1 つの文字列を返します。もし一般に有用な変換関数を書いた場合、他の人があなたの仕事から恩恵に預かれるように、どうか emacs-orgmode@gnu.orgに投稿して下さい。

A.5.4 ラジオリスト

ラジオリストを送受信する方法は、ラジオテーブル ("char 65.5.1 節「Radio tables」p.197 を参照) を送受信する方法と全く同じです。ラジオテーブル (ラジオリストの typo?) に関しては、関数 org-list-insert-radio-listを呼び出して、HTML モード、LaTeX モードそして Texinfo モードにラジオリストを挿入できます。

ここではラジオテーブルとの違いを説明します:

- Orgstruct モードが有効でなければなりません。
- ORGTBLの代わりに、キーワード ORGLSTを使います。
- 現在利用できるラジオリストの変換関数は、パラメータをとりません。
- リストの 1 つ目の項目で *C-c C-c* と押すと、動作します。

これは LaTeX の例です。LaTeX ファイルに以下が書いてあるとします:

```
% BEGIN RECEIVE ORGLST to-buy
% END RECEIVE ORGLST to-buy
\begin{comment}
#+ORGLST: SEND to-buy org-list-to-latex
- 新しい家
- 新しいコンピュータ
  + 新しいキーボード
  + 新しいマウス
- 新しい生活
\end{comment}
```

新しい家の上で 'C-c C-c' とタイプすると、LaTeX に変換されたリストが目印の 2 行の間に挿入されます。

A.6 Dynamic blocks

Org-mode の文書は動的なブロックを含むことができます。これらのブロックはユーザーが書いた関数によって更新される印のついた領域です。そのようなブロックの良い例は、*C-c C-x C-r*(8.4 節「Clocking work time」p.77 を参照) というコマンドで挿入した時計のテーブル (クロックテーブル) です。

動的なブロックは、そのブロックに名前を指定し、そしてそのブロックの内容を実行する機能のためのパラメータを指定する「BEGIN-END」構造によって囲まれている必要があります。

```
#+BEGIN: myblock :parameter1 value1 :parameter2 value2 ...

#+END:
```

動的なブロックを更新するには以下のコマンドを使用します。

C-c C-x C-u　　org-dblock-update
: その場所での動的なブロックを更新します。

C-u C-c C-x C-u
: 現在のファイルの全ての動的なブロックを更新します。

動的なブロックを更新するということは、すべての「BEGIN」と「END」の間のテキストを取り除き、「BEGIN」の行に書かれているパラメーターを解析し、それからそのブロックのための特定の書き出し用の関数を呼び出して新しい内容を書き込むことを意味します。もしもその書き出し用の関数の中で元の中身を使いたい場合は、追加のパラメータである:contentを使えます。

myblockという名前のブロックに対する書き出し用の関数はorg-dblock-write:myblockです。BEGIN の行にて与えられるパラメータのプロパティリストを単一パラメータとして受け取ります。ブロックの一例として、ブロックを更新する関数が最後に実行されたのはいつかを記録します:

```
#+BEGIN: block-update-time :format "on %m/%d/%Y at %H:%M"

#+END:
```

対応するブロックへの書き出し用の関数は、このようになっています:

```
(defun org-dblock-write:block-update-time (params)
   (let ((fmt (or (plist-get params :format) "%d. %m. %Y")))
     (insert "ブロックの最終更新日時は: "
             (format-time-string fmt (current-time)))))
```

もしもすべての動的なブロックが常に最新のものに更新されるようにしたいならば、関数org-update-all-dblocksをフック、例えば変数before-save-hookに追加します。org-update-all-dblocksはorg-mode内ではないバッファの中では何もしないように書かれています。

org-narrow-to-blockを使って、(他のブロックの様に) 現在のバッファを現在の動的なブロックへとナローイングできます。

A.7 Special agenda views

Org-mode では、以下のアジェンダビューによって選択されたものを絞り込む特別なフック関数を用意しています: `todo`, `alltodo`, `tags`, `tags-todo`, `tags-tree`。検索でマッチする度に、実際にアジェンダビューでマッチしたかを確認したり、もしマッチしなければどれくらい飛ばすのかを、関数で指定できます。全てのアジェンダビューに適用されるようにグローバルな条件を設定でき、それは変数 `org-agenda-skip-function-global`に格納されます。通常は、変数 `org-agenda-skip-function`を使って、このような定義を特定のカスタムサーチに適用します。

プロジェクトのツリーのどこかに WAITING というタグを含むプロジェクトのリストを作成する場合を考えてみましょう。ツリーの見出しには、プロジェクトを定義する TODO のキーワード「PROJECT」でマークされているとします。この場合、キーワードの PROJECT で TODO を検索し、プロジェクトのサブツリーに WAITING のタグが無ければ検索でマッチしてもスキップしたいとします。

これを達成するには、サブツリーをタグで検索する関数を書かなければなりません。もしそのタグが見つかったら、そのマッチはスキップされるべきではないと示すために、その関数は `nil`を返さなければなりません。もしそのようなタグが無ければ、そこから検索し続けるということを示すために、そのサブツリーの終端の場所を返さなくてはなりません。

```
(defun my-skip-unless-waiting ()
  "waiting でないツリーをスキップします"
  (let ((subtree-end (save-excursion (org-end-of-subtree t))))
    (if (re-search-forward ":waiting:" subtree-end t)
        nil          ; タグが見つかったので、スキップしません
      subtree-end))) ; タグが見つからないので、サブツリーの最後から検索を再開します
```

今からこの関数をアジェンダのカスタムコマンドで使えます。例えばこうなります:

```
(org-add-agenda-custom-command
 '("b" todo "PROJECT"
   ((org-agenda-skip-function 'my-skip-unless-waiting)
    (org-agenda-overriding-header "waiting (待ち) 状態のプロジェクト: "))))
```

関数 `org-agenda-overriding-header`を使って、アジェンダビューで意味のある見出しに変更したことにも注意して下さい。

カスタム検索を作成するには、あるレベルの制限を設けた検索を基づくのが一般的な方法です。自作の検索関数で全てのエントリーを検索したい場合は、単に 'LEVEL>0'[5] を検索した後に、必要なエントリを選択してください。

変数 `org-agenda-skip-function`に Lisp フォームを渡すこともできます。具体的に言うと、例えばこのように関数 `org-agenda-skip-entry-if`と関数 `org-agenda-skip-subtree-if`とを使えます:

`'(org-agenda-skip-entry-if 'scheduled)`
: 現在のエントリがスケジューリングされている場合、それをスキップします。

`'(org-agenda-skip-entry-if 'notscheduled)`
: 現在のエントリがスケジューリングされていない場合、それをスキップします。

[5] `org-odd-levels-only`を使用している場合、level number は階層構造のレベルと対応していて、スター「*」の数ではないことに注意して下さい。

`'(org-agenda-skip-entry-if 'deadline)`
: 現在のエントリがデッドラインを設定されている場合、それをスキップします。

`'(org-agenda-skip-entry-if 'scheduled 'deadline)`
: 現在のエントリがデッドラインを設定されているかもしくはスケジューリングされている場合、それをスキップします

`'(org-agenda-skip-entry-if 'todo '("TODO" "WAITING"))`
: 現在のエントリの TODO のキーワードが TODO もしくは WAITING の場合、それをスキップします。

`'(org-agenda-skip-entry-if 'todo 'done)`
: 現在のエントリの TODO のキーワードが DONE とマークされている場合、それをスキップします。

`'(org-agenda-skip-entry-if 'timestamp)`
: 現在のエントリが何かしらのタイムスタンプを持っている場合、それをスキップします。タイムスタンプはデッドラインかスケジューリングが考えられます。

`'(org-agenda-skip-entry 'regexp "regular expression")`
: 現在のエントリの中で正規表現「regular expression」にマッチした場合、それをスキップします。

`'(org-agenda-skip-entry 'notregexp "regular expression")`
: 現在のエントリの中で正規表現「regular expression」にマッチしない場合、それをスキップします。

`'(org-agenda-skip-subtree-if 'regexp "regular expression")`
: 2 つ上と同じですが、サブツリー全体をチェックしてスキップします。

従って特別な関数を定義しなかったとしても、WAITING があるプロジェクトの検索をこのように書くこともできました:

```
(org-add-agenda-custom-command
 '("b" todo "PROJECT"
   ((org-agenda-skip-function '(org-agenda-skip-subtree-if
                                'regexp ":waiting:"))
    (org-agenda-overriding-header "waiting (待ち) 状態のプロジェクト: "))))
```

A.8 Extracting agenda information

Org-mode は、コマンドラインから Emacs のバッチモードでアジェンダの情報にアクセスするコマンドを用意しています。この抽出された情報はプリンタに直接送ることも可能ですし、更にデータを処理させるプログラムに読み込ませることも可能です。1 つ目のコマンドは関数 `org-batch-agenda`であり、アジェンダビューを作成し、それを ASCII テキストとして STDOUT へと送ります。コマンドはパラメータとして 1 つの文字列をとります。文字列の長さが 1 のとき、`org-agenda-custom-commands`にあなたが設定したコマンドの 1 つが使われます。基本的に、*C-c a*の後にはどんなキーも使えます。例えば、現在の TODO リストを直接印刷するなら、このように使えます

```
emacs -batch -l ~/.emacs -eval '(org-batch-agenda "t")' | lpr
```

パラメータが 2 文字以上の文字列の場合、tags/TODO の検索に使われます。例えば、地元でのショッピングリスト (‘`shop`’というタグが付いていて、‘`NewYork`’というタグを含んでいない全てのアイテム) を印刷するには、このように使えます

```
emacs -batch -l ~/.emacs                                        \
      -eval '(org-batch-agenda "+shop-NewYork")' | lpr
```

このように、その場でパラメータを変えることもできます:

```
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
```

これは30日のアジェンダを作成します。情報は、Org-modeのファイル‘~/org/projects.org’だけに制限され、~/diary(日誌) ファイルでさえ含まれません。

もっと洗練された方法でアジェンダのデータを処理したい場合は、アジェンダのアイテム毎に値をカンマで区切られたリストを作成する org-batch-agenda-csvというコマンドを使えます。アウトプットの各行にはカンマで区切られたいろいろなフィールドがあります。行内のフィールドは以下の通りです:

category	アイテムのカテゴリーです
head	TODO キーワード、TAGS そして PRIORITY を除いた見出しです
type	アジェンダのタイプで、以下が考えられます
	todo　TODO 検索で選ばれた
	tagsmatch　tags 検索で選ばれた
	diary　diary からインポートされた
	deadline　デッドラインを設定された
	scheduled　スケジューリングされた
	timestamp　アポイントメント、タイムスタンプで選ばれた
	closed　日付に閉じられたエントリ
	upcoming-deadline　デッドラインが近づいている警告している
	past-scheduled　forwarded scheduled item
	block　エントリが日付のブロックを持っていて日付を含む
todo	もしあれば、TODO のキーワード
tags	継承したものも含めた全ての tags。コロンで区切られている。
date	2007-2-14 のような基準日
time	15:00-16:50 のような時間
extra	追加のプラニング情報の文字列
priority-l	もし与えられていれば優先順位の文字
priority-n	計算された数字の優先度

タイムスタンプ (すなわちデッドラインを設定された/スケジューリングされた) がそのアイテムのセクションを lead する場合にのみ Time もしくは date が与えられます。

このような CSV リストは後処理のスクリプトでとても使いやすいです。例の Perl プログラムは、Emacs/Org-mode から TODO リストを取得して、前にチェックボックスを挿入して、全てのアイテムを出力します。

```
#!/usr/bin/perl
```

```
# 実行する Emacs のコマンドを定義します
$cmd = "emacs -batch -l ~/.emacs -eval '(org-batch-agenda-csv \"t\")'";

# 実行してアウトプットを取得します
$agenda = qx{$cmd 2>/dev/null};

# 全ての行についてループします
foreach $line (split(/\n/,$agenda)) {
  # それぞれの値を取得します
  ($category,$head,$type,$todo,$tags,$date,$time,$extra,
   $priority_l,$priority_n) = split(/,/,$line);
  # 処理して出力します
  print "[ ] $head\n";
}
```

A.9 Using the property API

プロパティを扱うために使用できる関数を説明します。

`org-entry-properties` **&optional** *pom which*　[関数]

ポイントかマーカー (point-or-marker POM) の位置でのエントリのプロパティを全て取得します。
これには、TODO キーワード, tags, デッドラインの time 文字列, スケジューリングの time 文字列, 計測の time 文字列, エントリ内で更に定義されているプロパティが含まれます。返り値は連想リストです。プロパティのキーが何度も使われた場合は、キーは複数回出現します。
POM は nil となり得ます。その場合は現在のエントリが使われます。WHICH が nil もしくは 'all' の場合、全てのプロパティを取得します。WHICH が 'special' もしくは 'standard' の場合、部分集合 (subclass) を取得します。

`org-entry-get` *pom property* **&optional** *inherit*　[関数]

ポイントかマーカー (point-or-marker POM) の位置でのエントリーのプロパティPROPERTY の値を取得します。デフォルトでは、そのエントリーで局所的に定義されたプロパティのみを調べます。もし INHERIT が non-nil かつエントリーがプロパティを持っていない場合、階層のより高いレベルをチェックします。もし INHERIT がシンボル `selective`である場合、継承を使います。また、継承を使うのは、継承のためのプロパティを `org-use-property-inheritance`の設定が選んだときに限ります。

`org-entry-delete` *pom property*　[関数]

ポイントかマーカー (point-or-marker POM) の位置のエントリーからプロパティPROPERTY を削除します。

`org-entry-put` *pom property value*　[関数]

ポイントかマーカー (point-or-marker POM) の位置のエントリにプロパティPROPERTY の値 VALUE をセットします。

`org-buffer-property-keys` **&optional** *include-specials*　[関数]

カレントバッファにある全てのプロパティのキーを取得します。

`org-insert-property-drawer` [関数]
ポイントの位置にプロパティの引き出し (drawer) を挿入します。

`org-entry-put-multivalued-property` *pom property* **&rest** *values* [関数]
ポイントかマーカー (point-or-marker POM) のプロパティPROPERTY を値 VALUES にセットします。値 VALUES は文字列のリストでなければなりません。それらはスペースを区切り文字として連結されます。

`org-entry-get-multivalued-property` *pom property* [関数]
プロパティPROPERTY の値を、値がスペースで区切られたリストとして扱い、文字列のリストを返します。

`org-entry-add-to-multivalued-property` *pom property value* [関数]
プロパティPROPERTY の値を、値がスペースで区切られたリストとして扱い、このリストの中に値 VALUE があることを確認します。

`org-entry-remove-from-multivalued-property` *pom property value* [関数]
プロパティPROPERTY の値を、値がスペースで区切られたリストとして扱い、このリストの中に値 VALUE がないことを確認します。

`org-entry-member-in-multivalued-property` *pom property value* [関数]
プロパティPROPERTY の値を、値がスペースで区切られたリストとして扱い、このリストの中に値 VALUE があるかをチェックします。

`org-property-allowed-value-functions` [オプション]
特定のプロパティに許可された値を提供する関数へのフックです。その関数はプロパティの名前を単一の引数として受け取り、許可された値の単純なリストを返します。‘`:ETC`’ がその値の 1 つである場合、その値を補完の候補として使いますが、他の値も入力することができます。もし関数がプロパティと関係無い場合には、関数は `nil` を返します。

A.10 マッピング API を使う

Org-mode は一定の基準を満たす全てのエントリーを探すために、洗練されたマッピング機能を持っています。アジェンダビューを作成するために内部でこの機能を使われていますが、エントリのそれぞれもしくは選択されたエントリに対して任意の関数を実行するために API も使われます。

`org-map-entries` *func* **&optional** *match scope* **&rest** *skip* [関数]
範囲 SCOPE の内で MATCH にマッチして選択された各見出しで関数 FUNC を呼び出します。

FUNC は関数か Lisp のフォームです。関数は引数なしですが、見出しの始まりと終わりのカーソル位置とともに呼び出されます。関数への全ての返り値は、リストとして集められて返されます。

save-excursion フォームの中で FUNC を呼び出されるので、ポイントを保存する必要がありません。評価の後に、カーソルは (処理しているエントリの) 行末に移動し、検索を続行します。ある条件では、この方法では希望する結果を得られないかも知れません。例えば、もし現在の (サブ) ツリーを削除した (例えば アーカイブした) 場合、次のエントリをスキップすることを意味します。そのような場合、変数‘org-map-continue-from’ に希望するバッファポジションを FUNC に設定させることで、どこから検索を続けるべきかというポジションを指定できます。

アジェンダビューに使われるとき、MATCH は tags/property/todo にマッチします。繰り返しの間、このクエリーにマッチする見出しだけが考慮されます。もし MATCH が nil か t の場合、繰り返しで全ての見出しがチェックされます。

SCOPE はこのコマンドのスコープ (対象とする範囲) を決定します。以下のどれかになり得ます:

`nil`　カレントバッファ、 制限があればそれに従う
`tree`　現在位置のエントリから始まるサブツリー
`file`　カレントバッファ、制限なしで
`file-with-archives`
　カレントバッファ、関連したアーカイブも含む
`agenda`　全てのアジェンダファイル
`agenda-with-archives`
　全てのアジェンダフィル、関連したアーカイブファイルも含む
`(file1 file2 ...)`
　ファイルがリスト形式の場合、リスト内の全てのファイルをスキャンします

残りの引数は、スキャナー (読み取る関数?) のスキップ機能の設定として取り扱われます。以下のアイテムをとることができます。

`archive`　アーカイブタグ (archive tag) があるツリーをスキップします
`comment`　キーワード COMMENT があるツリーをスキップします
`function or Lisp form`
　`org-agenda-skip-function`の値として使われます。
　関数であればいつも t を返します、
　FUNC はは呼び出されません、
　関数がポイントを置いたままにした位置から検索は続きます

マッピングルーティーンへと与えられる関数はどのような動作もできます。エントリに関する情報を更に集めるためにプロパティAPI(“char 65.9 節「Using the property API」p.206 を参照) を使えますし、もしくはエントリのメタデータを変更するために使えます。 便利な幾つかの関数を紹介します:

`org-todo` **&optional** *arg*　[関数]
エントリの TODO 状態を変更します。引数 ARG が取り得る多くの値については、関数のドキュメント文字列を見て下さい。

`org-priority` **&optional** *action*　[関数]
エントリのプライオリティを変更します。ACTION が取り得る値についてはこの関数のドキュメント文字列を見て下さい。

`org-toggle-tag` *tag* **&optional** *onoff*　[関数]
現在のエントリのタグ TAG をトグルします。`on`か`off`のいずれかを設定するということは、タグをトグルするのではなく、on か off のどちらかであることを確かにします。

`org-promote`　[関数]
現在のエントリを一階層上に引き上げます。

`org-demote`　[関数]
現在のエントリを一階層下へ引き下げます。

これは、現在のファイルの中で TODOTOMORROWを持つ全てエントリを、キーワード UPCOMINDをつけた TODO のエントリへと変化させる簡単な例です。

```
(org-map-entries
   '(org-todo "UPCOMING")
   "+TOMORROW" 'file 'archive 'comment)
```

以下の例では、全てのアジェンダファイルに渡って、TODO のキーワードに WAITING があるエントリの数をカウントします。

```
(length (org-map-entries t "/+WAITING" 'agenda))
```

付録 B MobileOrg

MobileOrg (`http://mobileorg.ncogni.to/`) は Richard Moreland よって開発された *iPhone/iPod Touch* シリーズの携帯端末のためのアプリケーションです。*MobileOrg* は「リアル」のコンピュータ上にある Org-mode システムのために、オフラインのビューとキャプチャーによるサポートを提供します。その機能によって、実際のエントリーがどのように変化したかについて記録することができます。Android のユーザーは Matt Jones よって作成された MobileOrg Android (`http://wiki.github.com/matburt/mobileorg-android/`) のアプリをチェックしてください。

この付録では、*MobileOrg* で表示されるフォーマットの中でアジェンダビューを作成し、キャプチャーされたノートと *MobileOrg* で変更を、メインのシステムに統合していくために、Org-mode のサポートについて説明します。

MobileOrg の中でタグや TODO の状態を変更するためには、あなたは、例え、ひとつひとつのファイルが、一部しか使っていないとしても、全ての重要なタグや TODO キーワードを網羅するように `org-todo-keywords`と `org-tags-alist`変数のカスタマイズを設定しなければなりません。MobileOrg は、同様にインバッファの設定で状態やタグを提供しますが、これらの変数の中で設定されているものについてのみ、TODO の状態についての設定(5.2.5 節「Per-file keywords」p.47 を参照) や相互に排他的な タグ (6.2 節「Setting tags」p.57 を参照) についての装備状況を理解してください。

B.1 Setting up the staging area

MobileOrg はサーバー上のディレクトリを通して、Emacs と相互に連携させる必要があります。もしも公開のサーバーを使用しているなら、そのサーバーにアップロードされるファイルを暗号化したいと考えるかもしれません。この機能は Org-mode7.02 の *MobileOrg 1.5* (iPhone バージョン) で実現していますが、あなたのシステムに '`openssl`' をインストールしておく必要があるでしょう。暗号化するために、*MobileOrg* にパスワードを設定し、Emacs 上では、`org-mobile-use-encryption`[1] 変数を設定しておく必要があります。

無料の Dropbox.com (`http://dropbox.com`) のアカウント[2] を使い、ディレクトリを作成するのが最も簡単な方法です。MobileOrg で最初に Dropbox に接続したときに Dropbox の中に MobileOrg のディレクトリが作成されます。そのディレクトリが作成されたあと、次のように Emacs に書き込みます。

```
(setq org-mobile-directory "~/Dropbox/MobileOrg")
```

Org-mode はそのディレクトリの中に、*MobileOrg* 用のファイルを置いたり、そこからキャプチャーされたノートを読み込んだりするコマンドを持っています。

B.2 Pushing to MobileOrg

この操作では、`org-mobile-files`の中にリストアップされている全てのファイルを、`org-mobile-directory`で指定したディレクトリにコピーします。デフォルトではこのリストにはすべてのアジェンダファイル (`org-agenda-files`に登録されている) を含んでいます。しか

[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。

しながら、org-mobiles-filesをカスタマイズすることでファイルを追加できます。ファイル名は、org-directoryとの相対パスで登録されるので、すべてのファイルがこのディレクトリの中に入ることになります。プッシュする操作で、ユーザー[3] によって定義されたすべてのカスタマイズされたアジェンダビューを持った‘agendas.org’という特別な Org-mode ファイルを作成します。最後に、Org-mode は全ての他のファイルへのリンクを含んだ‘index.org’というファイルを書き込みます。*MobileOrg* は、最初サーバーからこのファイルを読み込み、それから、そこに置かれているすべてのアジェンダファイルと Org-mode ファイルをダウンロードします。ダウンロードのスピードを上げるために、MobileOrg は、どのファイルのチェック記号[4] が変更されたかどうかを読み取るだけなのです。

B.3 MobileOrg から pull する

MobileOrg がサーバーと同期する際に、Org-mode のファイルを閲覧するために呼び出すだけではありません。それによってサーバー上の‘mobileorg.org’というファイルに対して、フラグがつけられたり、変更されたりしたエントリーに対して、キャプチャーされたエントリーやポインターを追加します。Org-mode では、この情報を InBox ファイルに統合し、フラッグがつけられたエントリーにポインタを使って操作するという *pull* の操作機能をもっています。どのように動作するのでしょうか。

1. Org-mode は、‘mobileorg.org’[5] の中で発見した全てのエントリーを移動し、org-mobile-inbox-for-pull変数によって、ポインターが付けられたファイルに追加します。記録されたエントリと編集されたイベントは、それぞれ InBox ファイル中でトップレベルのエントリーとして位置づけられるでしょう。
2. エントリーを移動したあと、Org-mode は、*MobileOrg* の中で作られた変更を実行することを試みます。いくつかの変更は直接、ユーザーの確認無しに適用されます。例では、タグ、TODO の状態、見出しそして本文に対するすべての変更がはっきりと適用されるというものです。将来の行動のために、フラグを付けられたエントリーは、:FLAGGED:というタグが付けられるでしょう。そのため、再び簡単に見つけることができるでしょう。あるエントリを探したり、変更を適用するさいに問題があれば、ポインターのついたエントリーは inbox に残され、エラーメッセージの印がつけられるでしょう。あなたはあとでこれらの案件を手動で解決する必要があります。
3. Org-mode では、その際にフラグがつけられたすべてのエントリーとともに、アジェンダビューを作成できます。そしてユーザーはそれらの項目をやり終えたり、必要な行動を実行するでしょう。*MobileOrg* のエントリーにフラグが付けられている間に、ノートが保存されていたら、そのノートは、カーソルがアジェンダの行の上に置かれた時に、エコーエリア上に表示されるでしょう。

   `?`　そういう特別なアジェンダの中で、?が入力されたときには、別のウインドウでフラグの付けられたノートの全てが表示され、キルリング上に内容がコピーされます。そして、`? z C-y C-c C-c`を使用することで、フラグのつけられたノートを、そのエントリーの通常のノートとして保存することができ

[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。

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。この方法で、あなたはこのフラグの付けられたエントリーを意図したプロセスで完了させるという指示をすることになります。

もしも、すべてのフラグのついたエントリーを直接処理することができないならば、あなたは *C-c a ?*を入力して、アジェンダビュー[6] にいつでも戻ることができます。

[6] しかしながら、微妙な差があることに注意してください。*M-x org-mobile-pull RET*によって、自動的に作成されたビューは、最後に pull されて配置されたすべてのファイルを検索することを保証されています。これは、あなたのアジェンダファイルのリストに、現在含まれていないファイルも含みます。もしもあなたが、ビューを再作成するために、*C-c a ?*を最後に使用したならば、カレントのアジェンダファイルのみが検索されます。

付録 C 歴史と謝辞

Org-mode は 2003 年に誕生しました。Emacs Outline モードのユーザインターフェイスに対するフラストレーションから自由になるためでした。私（Carsten Dominik）は、自分のノートとプロジェクトを整理しようと試みていて、Emacs 使うことが自然なやり方に思えました。ところが、アウトラインツリーの一部を隠したり表示したりするだけでも、2〜3 個のキーを組み合わせたコマンドを、7 種類も覚えなければならず、これは全く受け入れがたいことでした（訳注：org-mode では TAB だけでよい）。また、アウトラインでノートを取るとき、私は絶えずツリーの構造を変更して、自分の考えや計画に合わせて整理しておきたかったのです。*Visibility cycling* と *structure editing* は、当初 ‘outline-magic.el’ パッケージに実装されていましたが、すぐにより一般的な ‘org.el’ に移しました。プロジェクトを計画するための心地良い環境になったので、次の段階は *TODO* リスト、基本的なタイムスタンプそしてテーブル機能を追加することでした。これらの機能は、org-mode が今日も追求している 2 つの主要なゴールを明らかにしました。すなわち、現代的で、アウトラインベースの、革新的かつ直感的な編集機能を持ったプレーンテキストモードになること。そして、ノートファイルに、プロジェクトプランイングの機能を直接組み込むことです。

Org-mode をリリースして以来、私や emacs-orgmode@gnu.org に送られてくる、文字通りに何千もの e メールは、バグレポート、フィードバック、新しいアイディア、そして時にはパッチやアドオンを絶えず提供してくれます。org-mode を改良するために手助けしてくれるすべての人に感謝します。org-mode の様々な側面で、改善に多大な影響を与えた方々のリストをここに記したいと思います。このリストは完全ではないと思うので、もし書き忘れてしまった方がいればお詫びすると共に、連絡をください。

リストを記す前に、何名かについてアルファベット順で特別に紹介します。

Bastien Guerry
: Bastien は、LaTeX エクスポートとプレーンなリストを構文解析する機能を含む、org-mode の数多くの拡張機能を実装しました（その多くが、現在は org-mode の中心に組み込まれています）。彼が副管理者としての役割を果たしていた開発初期の尽力は、org-mode プロジェクトの成功の中心となりました。また彼は Worg（訳注：org-mode のコミュニティサイト。http://orgmode.org/worg/）を考案し、org-mode のウェブにおける存在の認知を手助けし、orgmode.org のホスティングコストのスポンサーになりました。

Eric Schulte and Dan Davison
: Eric と Dan は、共同で org-babel システムに構築しました。これにより org-mode を、コードの評価、文芸的プログラミングそして再現可能な研究に対応する、他言語環境へと変えました。

John Wiegley
: John は、数々の素晴らしいアイディアとパッチを直接的に org-mode に提供してくれました。具体的には、ファイル添付システム（‘org-attach.el’）、AppleMail との一体化（‘org-mac-message.el’）、TODO リストの階層的な依存関係、習慣のトラッキング（‘org-habits.el’）そして、暗号化（‘org-crypt.el’）です。そして実は、org-mode のキャプチャシステムは、彼の素晴らしい ‘remember.el’ を拡張したものです。

Sebastian Rose
: Sebastian が居なければ、org-mode の HTML/XHTML エクスポートは、無知なアマチュアによる痛ましい機能になっていたでしょう。彼は org-mode の該当部

分をより高いレベルに押し上げました。また、‘org-info.js’の作者でもあります。この JavaScript は、org-mode から生成されたウェブページを info のように表示したり、単一キーによるナビゲーションでツリーを折り畳むインターフェイスを提供します。

さて！いよいよ貢献してくれた方々のリストに移ります。繰り返しますが、忘れているところがあれば教えてください。

- *Russel Adams* は、引き出しのアイディアを思いつきました。
- *Thomas Baumann* は、‘org-bbdb.el’と‘org-mhe.el’を作成しました
- *Christophe Bataillon* は、org-mode のウェブサイトで使っている素晴らしいユニコーンのロゴを作成しました。
- *Alex Bochannek* は、タイムスタンプの丸め込みのためのパッチを提供しました。
- *Jan B^^c3^^b6cker* は、‘org-docview.el’を作成しました。
- *Brad Bozarth* は、org-mode のファイルに RSS フィードの情報を引き込む方法を示しました。
- *Tom Breton* は、‘org-choose.el’を作成しました。
- *Charles Cave* の提案は、Remember のためのテンプレートの実装を活性化しました。現在は Capture のテンプレートになっています。
- *Pavel Chalmoviansky* は、時間指定したアイテムについてアジェンダの扱いに影響を与えました。
- *Gregory Chernov* は、テーブルの計算で Lisp 形式をサポートするパッチを作成し、XEmacs との互換性を改善しました。特に、XEmacs へ‘nouline.el’を移植しました。を行ないました。
- *Sacha Chua* は、Planner からのいくつかのリンクコードをコピーすることを提案しました。
- *Baoqiu Cui* は、DocBook エクスポートに貢献しました。
- *Eddward DeVilla* は、チェックボックスの統計を提案し、テストしました。また、プロパティのアイディアを思いつきました。しかも、そのための API があります。
- *Nick Dokos* は、いくつもの扱いにくいバグを見つけ出しました。
- *Kees Dullemond* は、HTML にあるプロジェクトのリストを直接編集していました。そのため、HTML エクスポートを含む初期の開発の一部をとても活性化しました。彼は、テーブルの列を狭めたり広げたりする手段を求めました。
- *Thomas S. Dye* は、Worg にドキュメントを寄稿し、マニュアルに Org-babel のドキュメントを組み込む手助けをしました。
- *Christian Egli* は、ドキュメントを Texinfo 形式に変換し、アジェンダを呼び起こし、HTML エクスポートに CSS フォーマットのパッチを生成しました。さらに、‘org-taskjuggler.el’を作成しました。
- *David Emery* は、エクスポートされた HTML のアジェンダでカスタムな CSS をサポートするパッチを提供しました。
- *Nic Ferrier* は、mailcap と XOXO サポートに貢献しました。
- *Miguel A. Figueroa-Villanueva* は、階層的なチェックボックスを実装しました。
- *John Foerch* は、隠されたアウトラインツリーで、どうすればインクリメンタルサーチが検索結果の周辺にコンテクストを表示するようになるかを明らかにしました。

- *Raimar Finken* は、‘`org-git-line.el`’を作成しました。
- *Mikael Fornius* は、メーリングリストの司会をしています。
- *Austin Frank* は、メーリングリストの司会をしています。
- *Eric Fraga* は、アイディアを出しテストを行ない、BEAMER エクスポートの開発を進めました。
- *Barry Gidden* は、Network Theory Ltd. を通した本の出版のための準備で、マニュアルを校正してくれました。
- *Niels Giesen* は、DONE としたツリーを自動的にアーカイブするアイディアを提供しました。
- *Nicolas Goaziou* は、プレーンなリストのコードの多くの部分を書き直しました。
- *Kai Grossjohann* は、他のパッケージとのキーバインディングの衝突を指摘しました。
- Network Theory Ltd. の *Brian Gough* は、org-mode のマニュアルを書籍として出版しました。
- *Bernt Hansen* は、タスクの自動リピート、タスクの状態遷移ログ、クロックテーブルのサポートで、多くの部分を担当しました。彼の明瞭な説明は、Git のバージョン管理システムを採用した時にとても重要でした。
- *Manuel Hermenegildo* は、いくつものアイディア、小数のバグフィックスとパッチを提出しました。
- *Phil Jackson* は、‘`org-irc.el`’を作成しました。
- *Scott Jaderholm* は、フットノート、折り畳んだエントリー間の空行の制御、そしてプロパティの列表示を提案しました。
- *Matt Jones* は、*MobileOrg Android* を作成しました。
- *Tokuya Kameshima* は、‘`org-wl.el`’と ‘`org-mew.el`’を作成しました。
- *Shidai Liu* ("Leo") は、LaTeX の組み込みを探求し検証しました。また彼は頻繁なフィードバックといくつかのパッチを提供しました。
- *Matt Lundin* は、テーブルの数式と名前付きの非表示アンカーへの参照をテーブルの最終行に置くことを提案しました。
- *David Maus* は、‘`org-atom.el`’を作成し、org-mode に関する git のチケットを管理しました。またメーリングリストに、有益な返信といくつかのバグフィックス、そしてパッチを多数提出する貢献者でした。
- *Jason F. McBrayer* は、アジェンダの CSV 形式のエクスポートを提案しました。
- *Max Mikhanosha* は、ノートの再配置のアイディアを思いつきました。
- *Dmitri Minaev* は、ファイルごとに優先順位の制限を設定するパッチを提出しました。
- *Stefan Monnier* は、Emacs Lisp コンパイラの出力を快適に保つためのパッチを 提供しました。
- *Richard Moreland* は、iPhone 向けに *MobileOrg* を作成しました。
- *Rick Moynihan* は、一つのファイルで複数の TODO の連なりを扱うことを可能にし、アジェンダをサブツリーに素早く制限可能にしました。
- *Todd Neal* は、INFO ファイルと Elisp 形式へのリンクについてパッチを提供しました。
- *Greg Newman* は、ユニコーンのロゴを現在の形にリフレッシュしました。
- *Tim O'Callaghan* は、ファイル内リンク、一般的なファイルリンクのための検索オプション、そしてタグを提案しました。

- *Osamu Okano* は、‘`orgcard2ref.pl`’を作成しました。リファレンスカードのテキスト版を作るための Perl スクリプトです。
- *Takeshi Okano* は、org-mode のマニュアルと、David O’Toole’s のチュートリアルを日本語に翻訳しました。
- *Oliver Oppitz* は、TODO アイテムが複数の状態を持つことを提案しました。
- *Scott Otterson* は、他の要素とのリンクに説明文を導入するきっかけを作りました。
- *Pete Phillips* は、タグの開発をする際に助力しました。また、頻繁にフィードバックを提供しました。
- *Martin Pohlack* は、文字列の挿入をアンドゥのために使いやすい束にするコードスニペットを提供しました。
- *T.V. Raman* は、バグをレポートして改善を提案しました。
- *Matthias Rempe* (Oelde) は、アイディアと Windows サポート、品質制御を提供しました。
- *Paul Rivier* は、名前付き注釈の基本的な実装を提供しました。また彼は、しばらくメーリングリストの管理者でした。
- *Kevin Rogers* は、リモートホストの VM ファイルにアクセスするコードを提出しました。
- *Frank Ruell* は、`keymapp nil`のミステリアスなバグ（‘`allout.el`’との衝突）を解消しました。
- *Jason Riedy* は、拡張パッチによって orgtbl テーブルのための送受信の仕組みを汎用化しました。
- *Philip Rooke* は、org-mode のリファレンスカードを作成しました。数多くのフィードバックを提供し、org-mode の文書化の基準を作成し適用しました。
- *Christian Schlauer* は、特に、リンクで利用するカギ括弧（<,>）について提案しました。
- *Paul Sexton* は、‘`org-ctags.el`’を作成しました。
- VM、BBDB、Gnus へリンクすることの最初のアイディアは、*Tom Shannon* の‘`organizer-mode.el`’によってもたらされました。
- *Ilya Shlyakhter* は、同一階層内でのアーカイブ、リテラルの例での行番号、そして、参照されたコード行のリモートハイライトを提案しました。
- *Stathis Sideris* は、ASCII を PNG に変換する‘`ditaa.jar`’を作成しました。現在これは、org-mode の‘`contrib`’ディレクトリに格納されています。
- *Daniel Sinder* は、サブツリーのロックによる内部的なアーカイブ機能のアイディアを思いつきました。
- *Dale Smith* は、リンクの省略記法を提案しました。
- *James TD Smith* は、便利なカスタマイズと機能のための数多くのパッチを提供しました。
- *Adam Spiers* は、グローバルなリンクコマンドを求めました。これはリンクの拡張システムの構築のきっかけとなり、行列のサポートを追加し、さらにマッピング API を提案しました。
- *Ulf Stegemann* は、特別な記号を HTML、LaTeX、UTF-8、Latin-1、そして ASCII に変換するための表を作成しました。
- *Andy Stewart* は、‘`org-w3m.el`’にコードを提供しました。org-mode のシンタックスにリンクを変換した HTML コンテンツをコピーする機能です。

- *David O'Toole* は、'`org-publish.el`'を作成しました。また、マニュアルにおける公開の章のドラフトを執筆しました。
- *Sebastien Vauban* は、LaTeX と BEAMER エクスポートについての多くの問題をレポートしました。また、GNUS のソースコードをハイライトする機能を有効にしました。
- *Stefan Vollmar* は、神経学のマックスプランク研究所での講演（ビデオ収録されている）を準備しました。また彼は、HTML エクスポートのコンセプトインデックスの生成を思いつきました。
- *Jürgen Vollmer* は、HTML 出力で目次を生成するコードを提供しました。
- *Samuel Wales* は、改善のためのフィードバックとバグレポートを提供しました。
- *Chris Wallace* は、'`QUOTE`'キーワードを改良するパッチを提供しました。
- *David Wainberg* は、アーカイブと、リンクシステムの改良を提案しました。
- *Carsten Wimmer* は、いくつかの変更を提案し、GNUS へのリンクに関するバグフィックスを手助けしました。
- *Roland Winkler* は、tty 端末で org-mode を動かすためのキーバインドの追加を依頼しました。
- *Piotr Zielinski* は'`org-mouse.el`'を作成しました。アジェンダブロックを提案し、様々なアイディアをコードスニペットを提供しました。

Concept index

#

1

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Key index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Command and function index

L

N

O

P

S

W

Variable index

これは変数とフェイスの完全なインデックスではありません。このマニュアルで言及したものだけを列挙しています。さらに詳しいリストは、*M-x org-customize RET*で表示されるカスタムブラウザで確認できます。表示されるツリーをクリックしてください。

C

H

L

O

P

U