

『野鳥』

Yet Another tex-mode for emacs

// YaTeX //

© 1991-1997 by HIROSE, Yuuji [yuuji@ae.keio.ac.jp]

1 はじめに

野鳥は、GNU Emacs で LaTeX 用の文書を作成する時に jLaTeX などのタイプセットコマンドや、プレビューアの起動を Emacs 編集画面中から行えるようにすると共に、拡張性の高い種々の補完機能によりソーステキストの編集を支援します。さらに LaTeX コマンドのオンラインヘルプによりマニュアルを調べる手間を軽減します。

Japanese manual See *YaTeX Japanese info*.

2 本マニュアル参照上の注意

本マニュアルでは以下の表記を 사용합니다。

- begin 型コマンド
`\begin{環境} ~ \end{環境}`という形式の LaTeX コマンドを指します。begin 型コマンドを補完入力することを begin 型補完と呼びます。
- section 型コマンド
`\section{タイトル}`や`\mbox{内容}`のように引数を取る LaTeX コマンドを指します。
- maketitle 型コマンド
`\maketitle`や`\tableofcontents`のように引数を取らない LaTeX コマンドを指します。
- large 型コマンド
`{\large ...}` や `{\tt ...}` のようなフォント/サイズ指定子を指します。
- `[prefix]`
野鳥の機能呼び出すためのプリフィクスキー。デフォルトでは `C-c` に割り当てられているので、特に変更していない場合本マニュアルの `[prefix]` という表記は、`C-c` と読み換えてください。

3 主な機能

- タイプセッタやプレビューアなどの編集画面からの起動 (`C-c t`)
- カーソル位置によらない固定リジョンの部分タイプセット
- `\includeonly` のワンタッチ更新
- エラー箇所への自動ジャンプ (`C-c ’`)
- `\begin{}`, `\end{}`, `\section...` などの LaTeX コマンドの補完入力 (`C-c b`, `C-c s`, `C-c l`, `C-c m`)
- 既に入力したテキストを環境やコマンド引数の中に取り込む括り補完 (`C-u`+通常補完キー)
- セクション区切り入力時の文書構造アウトライン表示
- セクションコマンドの一括シフト (Section 7.2.4 [view-sectioning], page 12)
- 補完辞書の学習
- LaTeX の環境やコマンドに応じたガイド付き引数入力
- 野鳥にないガイド付き引数入力関数の自動生成 (`yatexgen.el`)
- LaTeX コマンドの削除/変更 (`C-c k`, `C-c c`)
- ファイル間、`\begin<->\end`間、`\ref<->\label`間、`\cite<->\bibitem`ジャンプ (`C-c g`)
- 一括コメントアウト/アンコメントアウト (`C-c >`, `C-c <`, `C-c ,`, `C-c .`)
- アクセント記号/数式環境用コマンド/ギリシャ文字の入力支援 (`C-c a`, `;`, `/`)
- `tabular/array` 環境のカラム位置ガイド
- 標準的 LaTeX コマンドのオンラインヘルプ (`C-c ?`, `C-c /`)
- ドキュメントのインクルード構造の視覚的表示とバッファ切り替え (`C-c d`)

4 起動法

4.1 野鳥起動のための設定

~/`.emacs` に下の 2 項目を加えます。

```
(setq auto-mode-alist
      (cons (cons "\\\\.tex$" 'yatex-mode) auto-mode-alist))
(autoload 'yatex-mode "yatex" "Yet Another LaTeX mode" t)
```

次に野鳥の `emacs-lisp` ファイル群を置くディレクトリを `load-path` に加えます。たとえば、~/`src/emacs/yatex` に置くのであれば、

```
(setq load-path (cons (expand-file-name "~/src/emacs/yatex") load-path))
```

などとします。

以上の設定により、拡張子が `.tex` であるファイルを編集すると自動的に野鳥がロードされます。野鳥が正常に起動できたときはモードラインの表示が「やてふ」に変わります。

4.2 タイプセッタ/プレビューア環境の設定

利用する外部プログラムに関する以下の変数を確認し、必要なら正しい値に変更します。

`tex-command`

... 起動するタイプセッタのコマンド名

`dvi2-command`

... 起動するプレビューアのコマンド名

`NTT-jTeX` ... 改行+インデントによって、タイプセット後の字間が空いてしまうのを抑制する場合に `t` にする (古い `NTT-jTeX` で顕著に現れる)。具体的には、`fill` するときに各行の終わりに `%` を付加するようになる。

`YaTeX-kanji-code`

... 文書を作成する時の漢字コード

`dviprint-command-format`

... ファイルの印刷に使われるコマンド列の書式

`makeindex-command`

... `makeindex` コマンド

これらを変更する場合は、やはり~/`.emacs`にて、たとえば

```
(setq tex-command "bigjlatex")
```

のようにしてください。どのような値をセットすれば良いかについては、Section 20.1.1 [All customizable variables], page 29 を参照してください。

5 latex コマンド起動

LaTeX ソースの編集中、次のキー入力により、jlatex などのタイプセットプログラム (以後タイプセッタと呼ぶ)、プレビューアなどの起動ができます。

- [prefix] t j
... タイプセッタ (jlatex) 起動
- [prefix] t r
... タイプセッタ起動 (領域指定)
- [prefix] t k
... 動作中のタイプセッタの停止
- [prefix] t b
... jbibtex 起動
- [prefix] t p
... プレビューア起動
- [prefix] t l
... lpr(プリントアウト用) コマンド起動
- [prefix] t s
... xdvi -remote でのサーチ

5.1 タイプセッタ起動

タイプセッタを起動すると、編集ウィンドウが2つに分割され、片方のウィンドウにタイプセット画面が表示されます。出力されるメッセージと連動しタイプセットバッファは自動的にスクロールします。もし、途中でエラーが起こって止まってしまった場合にはタイプセットバッファに移り、(C-x o) タイプセッタの出している ? プロンプトに対して、x (処理の中断) などの指示を送ることができます。エラーを修正する場合は、

- [prefix] ' (prefix+アポストロフィ)
... 直前のエラー発生行へジャンプ

を入力することにより、タイプセッタがエラーを発生した行に移ることができます。また、タイプセッタの出力する overfull hbox などのウォーニング行にも対応していますので、順次 [prefix] ' を押すことにより、一つ前のウォーニング発生行にジャンプしていきます。

もし、気になるエラー行があった場合は、タイプセットバッファで、エラーの表示されている行にカーソルを合わせスペースキーを押すと LaTeX ソースの対応する行にジャンプします。

5.1.1 領域タイプセット

ポイントとマークの間、あるいはテキスト中に埋め込んだ %\#BEGIN と%\#ENDの間の領域 (Chapter 6 [percent#notation], page 7) だけを切り取ってタイプセットすることができます。この場合の documentstyle は、単一ファイルの LaTeX ソースの編集時にはそのファイルの documentstyle が、別ファイルから include している場合にはメインファイルの documentstyle が自動的に選択されます。指定領域は、メインファイルのあるディレクトリの texput.tex という臨時ファイルに出力されますので、上書きには注意してください。

5.2 プレビュー起動

`[prefix] t p` によりプレビューの起動ができます。さらに、もしあなたが、`-remote` 機能付きの `xdvi` を利用している場合は `[prefix] t s` を押すことによりカーソル位置の文字列を検索してそのページを表示するようにすでに起動中の `xdvi` に命令を送ります。これにより、現在編集箇所のタイプセット結果を即座に見ることができます。なお、`-remote` 機能付きの `xdvi` は `ftp://ftp.ae.keio.ac.jp/pub/text/xdvi/xdvi-remote` などから入手することができます。

5.3 プリントアウト

`[prefix] t l` を押してプリントアウトを指示すると、出力開始/終了ページを聞いてくるので、それぞれに答えます。これを省略したい時は、`universal-argument` をつけ、

```
C-u [prefix] t l
    ... ページ確認省略 lpr 起動
```

のように起動してください。

6 %#記法

本文中に %# ではじまるキーワードを埋め込むことでタイプセット起動等の制御をすることができます。

6.1 タイプセット用コマンド変更

起動するコマンドを変えたい時は本文中に次のような行を書きます。

```
#!jlatex-ntt
```

NTT jTeX と、ASCII jTeX を使い分けたいような場合に便利でしょう。

6.2 入力ファイル分割

また、章毎に別ファイルの .tex を作成している場合で、main.texから sub.texを \include しているような時は、sub.texの任意の位置に次のような行を埋め込みます。

```
#!jlatex main.tex
```

上の例のようにコマンド名だけでなく引数も書いた場合には、全てをそのまま shell に渡すので次のように書けば、ghostviewなどをプレビューアに使う時に便利です。

```
#!jlatex main ; dvi2ps main.dvi > main
```

なお、この行の最後の単語のピリオド以前を「メインファイル」のベースネームであると仮定します(上の2つの場合どちらも main)。

[*prefix*] *g* (Chapter 10 [Cursor jump], page 18 参照) でのファイル間ジャンプを有効に機能させるため、入力ファイル分割時には次のことに注意して下さい。

1. サブディレクトリを作って、その中にサブファイルを置くことはできるがサブディレクトリのサブディレクトリには置けない。
2. メインファイルからサブディレクトリ内のファイルを include する時には、相対パス指定を用いて、\include{chap1/sub} のように記述。
3. メインファイルが一つ上のディレクトリにある場合も、サブファイルには%#!jlatex main.tex のように記述する (../main ではない)。

6.3 領域の固定

[*prefix*] *tr* の領域指定のタイプセットでは、とくに指定のないかぎり、C-SPCでマークした位置と、ポイント(カーソル位置)の間を領域とみなしますが、必ず決まった領域をタイプセットしたい場合は、その領域を

```

%#BEGIN
<渡したい領域>
%#END

```

のように%#BEGINと%#ENDで囲み、カーソルを「%#BEGIN以降」に置いてください。この時の領域決定規則をまとめると次のようになります。

1. カーソル位置よりバッファの先頭方向に%#BEGINというキーワードがある場合
 1. %#BEGINよりバッファの末尾方向に%#ENDというキーワードが見つかった場合。⇒%#BEGIN から、その %#END のある位置まで。
 2. %#END が見つからなかった場合。⇒バッファの最後尾まで。

2. カーソル位置よりバッファの先頭方向に %#BEGIN というキーワードが見つからなかった場合。⇒マーク (C-SPC位置) とポイント (カーソル位置) の間の領域。

tabular 環境を何度も試行錯誤しているような場合は、\beginの前の行に %#BEGIN と書き、\endの次の行に %#END と書いておくと簡単に作表結果をテストすることができます。また、長い .tex ファイルの後半に %#BEGIN を書いておけば、前半の部分は無視できます。この BEGIN と END の消し忘れには十分ご注意ください。

6.4 lpr フォーマット

まず、プリントアウト用コマンド列のフォーマットについて説明します。コマンド列フォーマットは、3つの Lisp 変数によって表現されます。デフォルトの dvi2ps 用のフォーマットを例に説明します。

- (1)dviprint-command-format
"dvi2ps %f %t %s | lpr"
(2)dviprint-from-format
"-f %b"
(3)dviprint-to-format
"-t %e"

実際にプリントアウトする時は、(1)中の %s がファイル名に置き換えられ、%f が(2)の内容、%t が(3)の内容に置き換えられます。その際に(2)の文字列中の %b は「出力開始ページ」、(3)の文字列中の %e は「出力終了ページ」に置き換えられます。もし、ページを指定しない時には、%f, %t 両方とも無視されます。

この、dviprint-command-format を臨時に変えたい時は、LaTeX のソーステキスト中の任意の場所に、

```
##LPR dvi2ps %f %t %s | 4up -page 4 | texfix | lpr -Plp2
```

のように書いて下さい。プリントアウトするページ範囲をいちいち聞かせないようにする時に

```
##LPR dvi2ps %s | lpr
```

などとするのも便利かもしれません。

6.5 %#記法自体の編集

以上のような %# で始まる各種制御記法を編集するためには

```
[prefix] %  
... %#記法編集メニュー
```

を押します。

```
!)Edit-%#! B)EGIN-END-region L)Edit-%#LPR
```

というメニューが出て来るので、%#!に続くコマンドを変更したい時には!を、%#LPRで lpr フォーマットを変えたい時は L を、あらかじめ設定したリジョンを %#BEGIN ~ %#END で括りたい時は、bを押します。bを選んだ時には、それまでバッファ中に置かれていた %#BEGIN, %#END が自動的に消去されます。

7 補完入力

LaTeX での環境名などは、野鳥の補完機能を利用して能率的に入力することができます。

7.1 begin 型補完

`\begin{env}... \end{env}` の様な形式の入力の補完を begin 型補完と呼ぶことにします。begin 型補完は、

```
[prefix] b
    ... begin 型補完開始 (標準では C-c b)
```

で始まります。頻繁に用いられる次の LaTeX 環境の補完は、`[prefix] b` に続く次の 1 文字を入力するだけで、`\begin{xxx}... \end{xxx}` を完成させます。

```
[prefix] b c
    ... \begin{center}... \end{center}
```

```
[prefix] b d
    ... \begin{document}... \end{document}
```

```
[prefix] b D
    ... \begin{description}... \end{description}
```

```
[prefix] b e
    ... \begin{enumerate}... \end{enumerate}
```

```
[prefix] b E
    ... \begin{equation}... \end{equation}
```

```
[prefix] b i
    ... \begin{itemize}... \end{itemize}
```

```
[prefix] b l
    ... \begin{flushleft}... \end{flushleft}
```

```
[prefix] b m
    ... \begin{minipage}... \end{minipage}
```

```
[prefix] b t
    ... \begin{tabbing}... \end{tabbing}
```

```
[prefix] b T
    ... \begin{tabular}... \end{tabular}
```

```
[prefix] b ^T
    ... \begin{table}... \end{table}
```

```
[prefix] b p
    ... \begin{picture}... \end{picture}
```

```
[prefix] b q
    ... \begin{quote}... \end{quote}
```

```
[prefix] b Q
    ... \begin{quotation}... \end{quotation}
```

```
[prefix] b r
... \begin{flushright}...\end{flushright}
```

```
[prefix] b v
... \begin{verbatim}...\end{verbatim}
```

```
[prefix] b V
... \begin{verse}...\end{verse}
```

上記のもの以外の環境名は Emacs の持つインクリメンタルな補完機能を用いて入力します (上記の環境名も以下の補完入力可能)。

```
[prefix] b SPC
... begin 型補完入力
```

[prefix] b SPC と入力すると、最下行のミニバッファに

```
Begin environment(default document):
```

と表示されます。ここで、何も入れずにリターンキーのみ押すと、括弧内に出ているデフォルトの環境名が入力されますが、適当な環境名を入力すると、`\begin{環境名} ... \end{環境名}` が文書中に挿入されます。ミニバッファで環境名を入力するときに、環境名の頭文字を入力し「スペース」をたたくと、一致する環境名が内部テーブルに存在した場合、正しい環境名に補完されるので、入力の手間が省けます。内部テーブルに存在しない環境名を入力した時はユーザ専用のテーブルに登録され、さらにそのテーブルを自動的に、ユーザ辞書 (デフォルトでは `~/yatexrc`) に保存します。

さらに、特定の環境を補完入力した時にはその環境で必ず用いられるエントリを自動挿入します (例: `itemize` 環境における `\item` など)。挿入されたエントリが不要な場合には `undo` によって消去して下さい。

7.1.1 既にしたテキストを環境で括る

ところで、最初に書いてしまったブロックを後から、`itemize` 環境の中に閉じこめたいと思うことがあります。そのようなときは、あらかじめ閉じこめたい段落をマークして、`begin` 型補完の各コマンドの [prefix] の次の『小文字の 'b'』を『大文字』に変えて起動して下さい。(または、`C-u` を先に打ち、`universal argument` をつけても可能です)

例えばあるパラグラフを `description` 環境の中に入れたいときは、そのパラグラフをマークしてから、

```
[prefix] B D
(または ESC 1 [prefix] b D)
(または C-u [prefix] b D など)
```

とタイプしてください。これは、[prefix] b SPC の補完入力にもあてはまり、`b` を大文字に変えて、[prefix] B SPC とタイプすれば、あらかじめマークしておいたりジョンを、`begin` と `end` の環境で括ります。

7.2 section 型補完

`\section{目的}` のような形式の入力の補完を `section` 型補完と呼ぶことにします。`section` 型補完は、

```
[prefix] s
... section 型補完
```

で実行します。[prefix] s を入力するとミニバッファに、

```
(C-v for view-section) \???{} (default documentstyle):
```

というプロンプトが現れるので、そこで ‘section’ のような LaTeX コマンド名を入力します。ここでもリターンキーのみで括弧内のデフォルト値が選択されるほか、‘chapter’ などのような頻度の高い名称入力にはスペースキーによる補完機能が有効です。次に、{} の中身の入力を促す、

```
\section{???}:
```

というプロンプトが現れるので、セクションのタイトルなどを入力します。たとえば、

```
(C-v for view-section) \???{} (default documentstyle): section
\section{???}: 目的
```

のように入力した場合は、文章中に

```
\section{目的}
```

が挿入され、

```
(C-v for view-section) \???{} (default section): vspace*
\vspace*{???}:
```

のように {} の中身を省略したときは、

```
\vspace*{}

```

だけが挿入され、改行はせずカーソルは自動的に中括弧の内側に移動します。

7.2.1 2個以上の引数をとる section 型コマンド

ところで、‘\addtolength{\topmargin}{8mm}’ などのように、引数を二つ以上取る LaTeX コマンドがあります。このようなコマンドの補完入力には、section 型補完呼び出しに引数を付けてください。例えば上の ‘addtolength’ の例であれば、引数 2 を指定します。つまり、

```
C-u 2 [prefix] s (または、ESC 2 [prefix] s)
```

と section 型補完を呼び出した後、

```
(Ctrl-v for view-section) \???{} (default vspace*): addtolength
\addtolength{???}: \topmargin
Argument 2: 8mm
```

のように入力してください。最初の addtolength の部分と、第一引数である \topmargin の入力は当然スペースによる補完入力が可能です。ユーザ辞書に登録される LaTeX コマンドには、この引数の数も学習されるので、最初の補完の時引数の数を指定して起動しておけば、以後の補完時には、記憶された個数だけ引数を聞いて来るようになります。あとで引数の個数を変えたい時は、再び C-u を用いて個数を指定し直すことで、自動的に辞書中の引数の個数の部分を更新します。

7.2.2 既にしたテキストを括る

また、起動コマンドの s を大文字に変えて起動すると、あらかじめ書いた文章を section 型コマンドの第一引数として括ります。

7.2.3 再帰補完

高度な使い方になるかもしれませんが、section 型補完の引数の入力時にさらに補完入力を利用することができます (section/large/maketitle 型に限る)。section 型コマンドの引数に更なる LaTeX コマンドが来る場合にはミニバッファで野鳥の補完キーを再帰的に入力することで引数の入力も効率的に行なえます。

7.2.4 セクション区切りのアウトライン表示

通常の section 型補完の時にミニバッファで `C-v` を押すと現在存在するセクション区切りコマンド全てを `*Sectioning Lines*` というバッファに一覧表示します (「<<-」のついている行がもっとも近いセクション区切り)。この時ミニバッファで `C-p`, `C-n` を押すと 'part', 'chapter', ..., 'subparagraph' のコマンドが論理階層の高さにしたがって上下します。また、`C-v`, `M-v` を押すとセクション区切り一覧バッファがスクロールし、数字の 0~7 を押すとある高さ以上のセクション区切りだけを選んで表示します (実際にやってみれば分かります)。

`*Sectioning Lines*` バッファは、

`M-x YaTeX-section-overview`

... セクション区切り一覧バッファを生成

で作成することができます。このバッファを選択し任意の行でスペースを押すと、該当するセクション区切りのある本文中の場所にジャンプします。さらに、同バッファで `u` を押すと、ソーステキストの対応するセクションコマンドが一階層上がり (例: subsection が section に変わる)、`d` を押すと一階層下がります。`*Sectioning Lines*` バッファにあるセクション区切りの行をマークしておいて `U` を押すとリジョン内のもに対応するソーステキストのセクションコマンドすべてが一階層上がり、`D` を押すと下がります。セクション区切り一覧バッファで利用できるキーコマンドには以下のものがあります。

<code>SPC</code>	... 対応するソース行へジャンプ
<code>.</code>	... 対応するソース行を表示
<code>u</code>	... カーソル位置に対応するセクションコマンドを一階層上げる
<code>d</code>	... カーソル位置に対応するセクションコマンドを一階層下げる
<code>U</code>	... マークしたセクションコマンドを一階層上げる
<code>D</code>	... マークしたセクションコマンドを一階層上げる
<code>0~6</code>	... レベル <code>n</code> 以下のセクションコマンドを隠して表示

7.3 large 型補完

`{\large}` のような形式の補完を large 型補完と呼ぶことにします。

`[prefix] l`

... large 型補完開始

が large 型補完の開始です。`[prefix] l` を押すと、ミニバッファに

`{\???}` (default large):

と表示されるので、上記のものと同じ要領で補完入力して下さい。補完候補に用意されているのは、'footnotesize' や 'huge' のような文字サイズ指定子と、'bf' や 'dg' のようなフォント指定子です。

7.3.1 既にかいた文字を括る

また、begin 型補完の時と同様、先に書いてしまった一連の文章の文字のサイズを変えたいと思う時がありますが、そのような時は、サイズや大きさを換えたい文字の範囲をマークしてから、呼び出しキーを `[prefix] L` と、大文字の `L` に変えて呼び出せば、そのリジョン全体が、ブレースで囲まれます。

7.4 maketitle 型補完

`\maketitle` の形式の補完を `maketitle` 型補完と呼ぶことにします。

```
[prefix] m
    ... maketitle 型補完開始
```

で、`maketitle` 型補完を開始します。補完の要領は今までのものとまったく同じです。LaTeX 用のコマンド名が補完候補として用意されています。

7.5 随時補完

さて、今まで述べた典型的な LaTeX コマンド形式の補完入力を用いずに、今入力しようとしている LaTeX コマンドを文書中の任意の位置で随時補完することもできます。LaTeX コマンド (先頭が `\` で始まる) を入力している途中で、

```
[prefix] SPC
    ... 随時補完
```

を入力すれば、全ての補完候補の中から一致するものが選ばれカーソル位置に挿入されます。

7.6 end 補完

現在開いたままの環境名を自動的に検出し、`\end{環境名}` を挿入します。begin 型補完を用いれば環境の閉じ忘れはないのですが、時にはついつい手で `\begin{環境名}` を入れてしまい、悲しい思いをすることがあります。そのような時には気にせず続けて文章を入力し、しかるのちに

```
[prefix] e
    ... end 補完
```

とすることで、現在開いている環境名で `\end{}` が補われます。

7.7 アクセント記号補完

欧文のアクセント記号 (`\{o}` など) を入力する時は、

```
[prefix] a
    ... アクセント記号入力
```

を押すと、ミニバッファに

```
1: ' 2: ' 3: ^ 4: " 5: ~ 6: = 7: . u v H t c d b
```

というメニューが出て来るので、数字、または対応する記号/英字を入力して下さい。すると編集バッファに、

```
\{ }
```

があらわれ、カーソルが `{}` 内に位置するので、さらに一文字入力する事で、

```
\{o}
```

が完成され、カーソルは `{}` の外に戻ります。

7.8 数式記号イメージ補完

主に数式モードで使用される、矢印や Σ などの記号を擬似的に表現するキー入力で、 LaTeX コマンドを入力できます。これは野鳥自身の「数式モード」でのみ動作します。野鳥はカーソルが TeX の数式環境の中にある時に、`;` や `:` に特殊な機能を持たせます。

さて、例えば、 \leftarrow (`\leftarrow`) を ASCII 文字だけで表現する場合、一般的には「<-」のようにしますが、これを利用して、数式記号イメージ入力モードで `\leftarrow` を入力するには、`;` (セミコロン) を打ってから `<-` と入力します。同様に、長い矢印 \longleftarrow (`\longleftarrow`) を ASCII 文字だけで表現する場合「<-」とするので、`\longleftarrow` を入力するためには、`;<--` と入力します。あるいは無限大記号を ASCII 文字だけで表現する時は「oo」のようにすることから、`\infty` を入力する時は、`;oo` とキー入力します。

これらの操作をまとめると次のようになります。

INPUT	入力される LaTeX コマンド
<code>;<-</code>	<code>\leftarrow</code>
<code>;<--</code>	<code>\longleftarrow</code>
<code>;<--></code>	<code>\longleftarrow</code>
<code>;o</code>	<code>\circ</code>
<code>;oo</code>	<code>\infty</code>

いずれの場合も、イメージ入力を行っている途中で望みのものがバッファに表示されたなら、そこでイメージ入力を止めて次の編集動作に移っても構いません。

数式環境中で `;` 自身を入力するには、`;;` のようにします。イメージ入力の途中で `TAB` を押すと、それまで入力した文字で始まるものの一覧が表示されます。ここで目的の LaTeX コマンドまでカーソルを移動し再度 `TAB` を押すことでその LaTeX コマンドがバッファに挿入されます。

どのキー入力にどの記号が対応しているか全て知りたい時は、`;` を押した直後に `TAB` を押してください。以下の例は、`;<` と押した後に `TAB` を押したものです。

KEY	LaTeX sequence	sign
<code><</code>	<code>\leq</code>	\leq
<code><<</code>	<code>\ll</code>	\ll
<code><-</code>	<code>\leftarrow</code>	\leftarrow
<code><=</code>	<code>\Leftarrow</code>	\Leftarrow

左から [入力キー]、[対応する LaTeX コマンド]、[(擬似) 記号図示]、という順でメニューが出て来るので、よく使うものを覚えておくと良いでしょう。ものによっては ASCII 文字で表現することが困難なので、あまり覚えやすいキー並びではないものがあるでしょうから、そのような場合は `\maketitle` 型補完で入力するか、以下に述べる対応表の設定を行って単純なキー並びのものを設定すると良いでしょう。

入力キーと LaTeX コマンド、記号の対応表を個人的に設定したい場合は Emacs-Lisp 変数 `YaTeX-math-sign-alist-private` に定義してください。その内容とデフォルトのものを合わせたものが対応表として使用されます (`private` の方が優先される)。なお、この変数の構造については `yatexmth.el` を参照してください。

7.9 ギリシャ文字補完

もう一つ、数式環境中で `:` を押すとギリシャ文字入力モードに入ります。`:` を押した直後に `a` を押すと `\alpha` が、`g` を押すと `\gamma` が、などアルファベットに対応したギリシャ文字が挿

入されます。操作方法は;の数式記号補完とまったく同じです。まずは:の直後にTABを押してどのアルファベットにどのギリシャ文字が対応しているか調べてみてください。

;と:を数式環境中で押しているにもかかわらず、イメージ補完が働かない場合は、C-u ;のように universal-argument をつけてキーを押すことにより、強制的にイメージ補完に入ることができます。また、この時にどのような状態で数式環境内判定に失敗したかをご連絡下さい。

8 ローカル辞書

補完入力用の候補は三種類の辞書から構成されています。一つは `yatex.el` に組み込まれた「標準辞書」、もう一つはユーザが個人的に常用するコマンドを保存する「ユーザ辞書」、そしてもうひとつはあるディレクトリでのみ有効なコマンドを保存する「ローカル辞書」です。

補完入力時に新しい単語を入れた場合に、その単語をどの辞書に入れるか聞いて来ます。

```
'foo' is not in table. Register into: U)serDic L)ocalDic N)one D)iscard
```

というプロンプトに対し、`u`と答えると「ユーザ辞書」を、`l`と答えるとローカル辞書を更新し、`n`と答えると辞書ファイルは更新せず現在の Emacs セッションのみ有効な単語とし、`d`と答えると新たな単語を学習せずに捨てることになります。

もし、ローカル辞書の機能はならず、全てユーザ辞書の更新のみでよいと言う場合には `~/.emacs`などで、

```
(setq YaTeX-nervous nil)
```

として下さい。

9 コメントアウト

LaTeXの編集には試行錯誤がつきものです。ある部分を一括でコメントアウトしたり、コメントを外したりしたいことがあります。

[prefix] >
... リジョンを % でコメントアウト

[prefix] <
... リジョンの % のコメントを外す

は、あらかじめ設定したリジョンに対しての操作、

[prefix] .
... 現在のパラグラフをコメントアウト

[prefix] ,
... 現在のパラグラフのコメントを外す

は、カーソルの位置するパラグラフ全体に対しての操作です。なお、ここでいう「パラグラフ」は (mark-paragraph) 関数によりマークされる範囲を指します (標準設定で ESC h にバインドされている)。なお、既に % でコメントアウトされているパラグラフに対して繰り返しパラグラフのコメントを使用した場合の動作は保証しませんので御注意ください。

さて、文章に対してだけでなく、時には `\begin`, `\end` 自体に対してもコメントアウトの操作をしたいときがあります。このようなときは、`\begin{}` あるいは `\end{}` の行にカーソルを合わせ、

[prefix] >
... `\begin{}`~`\end{}` 全てコメントアウト

[prefix] <
... `\begin{}`~`\end{}` 全てコメントを外す

とすることで、`\begin`~`\end` で囲まれる環境全てに対してコメント操作し、

[prefix] .
... `\begin{}` と `\end{}` をコメントアウト

[prefix] ,
... `\begin{}` と `\end{}` のコメントを外す

は、対応する `\begin` と `\end` 2行だけを、コメント操作の対象とします。リジョンをコメントアウトしようとして、マークを設定したのちにカーソルを移動し [preifx] > を押してもカーソルが `\begin{}` の上にあると `\begin{}`~`\end{}` モードでコメント機能が働いてしまうので注意して下さい。

10 カーソルジャンプ

10.1 対応オブジェクトへのジャンプ

文書中のいろいろな場所で

`[prefix] g`

... 対応するオブジェクトにジャンプ

を押すことにより、カーソル位置の LaTeX コマンドに対応する場所にジャンプします。対応関係が存在すると解釈されるコマンドには以下のものがあります。

- `\begin{}` \leftrightarrow `\end{}`
- `##BEGIN` \leftrightarrow `##END`
- 画像ファイルの取り込みマクロ \rightarrow 対応する viewer/お絵かきツール起動
- `\label{}` \leftrightarrow `\ref{}`
- `\include(\input)` \rightarrow 対応するファイル
- `\bibitem{}` \leftrightarrow `\cite{}`

`\begin{}` か `\end{}` の行で `[prefix] g` を押すことにより、対応する end/begin の行にジャンプします。もちろん対応するものがない場合はエラーになります。またこれは、領域固定のための `##BEGIN` と `##END` のペアに対しても同様に動作します。なお、label/ref や cite/bibitem 対応するものが別ファイルにある時は、ジャンプ先となるファイルがオープンされていなければなりません。See Chapter 6 [percent#notation], page 7. メインの .tex ファイルの `\include{chap1}` などにカーソルを合わせ、`[prefix] g` を押すと、chap1.tex にジャンプします。

また、

`[prefix] 4 g`

... 別ウィンドウで対応オブジェクトにジャンプ

を押すと、対応するオブジェクトへのジャンプを別ウィンドウで行います。ただし、この機能は begin/end, ##BEGIN/##END 間のジャンプに対しては (意味がないと思われるので) 機能しないので注意してください。

10.2 お絵描きツール起動

上記の「画像ファイルの取り込みマクロ」とは、例えば `\epsfile{file=foo}` のような挿絵取り込みコマンドのことで、この行にカーソルを合わせて `[prefix] g` を押すとその画像ファイルの元となったファイルに対応するお絵描きツールを起動してオープンします。起動するツールの判定は以下のようなになります。

1. カレント行が変数 `YaTeX-processed-file-regexp-alist` に定義されている正規表現のいずれかとマッチしたら、ファイル名に相当する部分を `\\(\\)` から抜き出して覚えておく (何番目の `\\(\\)` かは変数の各リストの cdr 部に入れておく)。マッチしなければ何もしない。
2. 行末に、変数 `YaTeX-file-processor-alist` に登録されているコマンドが「%コマンド」のように書いてあれば強制的に「コマンド ファイル名. 拡張子」を起動。

3. なければ、変数 `YaTeX-file-processor-alist` の各リストの `cdr` 部に入っている拡張子を「ファイル名」の後ろに足したファイルが存在するか順次調べて、存在した場合 `car` 部に入っているコマンドを起動する。
4. 以上どれかにマッチしなければあきらめる。

変数 `YaTeX-file-processor-alist` と変数 `YaTeX-file-processor-alist` の設定方法についてはそれぞれの変数について `describe-variable` して説明を読んで下さい。うまく設定すると、画像ファイルにかぎらず、任意の形式のファイルを任意のプロセッサで処理するコマンドを簡単に呼び出すことができます。

10.3 メインファイルへのジャンプ

`chap1.tex`のようなサブファイルで、

```
[prefix] ^
... メインファイルにジャンプ
```

```
[prefix] 4 ^
... 別ウィンドウでメインファイルにジャンプ
```

を押すと、メインファイルの編集バッファに切替えます。もし、メインファイルをオープンしていない場合は、カレントディレクトリから探して自動的にオープンします。

10.4 環境を単位としたジャンプ

さらに現在の環境を単位として機能するコマンドに以下のものがあります。

`M-C-a` ... 環境の先頭 (`\begin`) へジャンプ

`M-C-e` ... 環境の末尾 (`\end`) へジャンプ

`M-C@` ... 環境全体をマーク

上記のコマンドは通常の `[prefix]` キーではなく `META` キーをプリフィクスとして機能するのでご注意ください。

10.5 最後の補完位置へのジャンプ

野鳥は補完入力した位置を常にレジスタ 3 に保存しています。入力途中で如何なるファイルの如何なる位置に行ったとしても、`C-x j 3` (`jump-to-register`) を使って直ちに最後の補完入力位置に戻ることができます。

11 変更/削除

既に入力されている LaTeX コマンドの変更/削除のために以下の機能が用意されています。

`[prefix] c`
 ... カーソル位置の LaTeX コマンドの変更

`[prefix] k`
 ... カーソル位置の LaTeX コマンドの削除

これらのコマンドは、コマンドを起動する場所によって動作を決定するので注意して下さい。

11.1 LaTeX コマンドの変更

変更したい LaTeX コマンドにカーソルを合わせて `[prefix] c` を押すとそのコマンドを補完入力などを用いて手軽に変えることができます。`[prefix] c` で変更できるコマンドには以下のものがあります。

- `begin/end` の環境名
- section 型コマンドのコマンド名
- section 型コマンドの引数
- section 型コマンドのオプションパラメータ ([] で囲まれたもの)
- large 型コマンド
- (イメージ補完で入力可能な) 数式モード専用の `maketitle` 型コマンド

変えたい section 型コマンドの引数がさらに LaTeX コマンドを含む場合は、その引数を囲む中括弧の上で `[prefix] c` を押すことで中のコマンドを変更対象判定から除外することができます。

11.2 LaTeX コマンドの削除

`[prefix] k` は起動する位置により次のような動作を行います。

起動位置	動作
<code>\begin, \end</code> の行	<code>\begin\end</code> ペアの削除
<code>##BEGIN, ##END</code> の行	<code>##BEGIN, ##END</code> ペアの削除
section 型コマンドの上 (中)	section 型コマンドの削除
フォント指定括弧の上	フォント指定の削除
括弧の上	対をなす括弧の削除

`\begin, \end` および `##BEGIN, ##END` を削除する場合、`\begin, \end` や `##BEGIN, ##END` の存在する行はまるごと削除されるので、それらの一行に `\begin` などを二つ以上連ねて書かないように注意してください。上記のものはすべて本文を囲う「容器」を削除するように働きますが、`universal-argument (C-u)` を打った後で `[prefix] k` をタイプすると、それぞれの「容器」に含まれる「中身」も一気に削除します。以下の例を参考にして下さい。

元のテキスト:	<code>[prefix] k</code>	<code>C-u [prefix] k</code>
本文 <code>\footnote{脚注}</code> です。	本文脚注です。	本文です。
↑ (カーソル位置)		

12 桁揃え

12.1 item の桁揃え

itemize 環境中にある `\item` の項目 (文章) が複数行に渡る場合に、項目の先頭を桁揃えしたい場合には、

```
M-q      ... 桁揃え
```

によって、その item のインデントの深さに応じて fill されます。なお、古い NTT jTeX を使用している場合には、Lisp 変数 `NTT-jTeX` を `t` にセットして下さい。

このとき、変数 `YaTeX-item-regexp` の値 (標準では `"\\item"`) を項目指定コマンドの正規表現として検索に使用します。itemize 環境で、独自のコマンドを定義して項目を列挙している場合 (例えば `\underlineitem`) は、`~/emacs` で次のように指定して下さい。

```
(setq YaTeX-item-regexp
      "\\(\\(\\(\\(sub\\))*item\\)\\)\\(\\(\\(\\(underlineitem\\)\\)\\)")
```

この変数の指定の仕方がよく分からない場合は、独自の項目列挙コマンドの名前を `"\item"` で始まるものにして下さい (例えば `"\itembf"`)。

野鳥の `M-q` では `\item` を環境に応じて以下のように「ハングインデント」します。

itemize, enumerate 環境:

```
> \item[ほげほげ] 英語では、特に意味のない単語を 'foo' であらわしま
>                  すが、これの日本語版ともいえる単語が「ほげほげ」
>                  です。
```

description 環境:

```
> \item[へろへろ] 「ほげほげ」をでたらめが単語として使った時に、第
>                  2 のでたらめな単語として「へろへろ」が使われることが多
>                  いようです。
```

12.2 パラグラフの桁揃え

itemize 環境以外でのパラグラフの桁揃え (fill) は、基本的に他のモードと同じように機能しますが、`verbatim` 環境や、`tabular` 環境など桁揃えをすると悲惨な状況になるような環境中では機能しません。また、`\verb` で括弧してあるものは決して行分割されません (変数 `YaTeX-verb-regexp` で制御)。さらに、一時的にインデントの深さを変えてある箇所では、そのインデントの先頭で `M-q` を押すことにより `fill-prefix` をいちいち変更しなくて桁揃えができます。

13 勝手にincludeonly

ファイルを分割して文章を入力している時には、メインファイル中に

```
\includeonly{現在編集中的ファイル名}
```

のように書いておくことで、タイプセットの時間を節約できますが、ちょっと他のファイルを手直ししたい時には

```
\includeonly{ちょっと手直ししたいファイル名}
```

と書き直さなければならず手間がかかります。野鳥では現在編集しているファイル名がメインファイルの\includeonlyにない場合には自動的にこれを検出し、次の指示を仰ぎます。

```
A)dd R)eplace %)comment?
```

現在編集中的のファイルを \includeonly のリストに加えたい時には a を、現在編集中的のファイルだけを \includeonly にしたい時は r を、\includeonly の行をコメントアウトして無効化したい時には、%をそれぞれ押して下さい。

14 ここはどこ？

項目数の多い tabular などたくさん書いていると下の方の行で、いま書いている桁がどこに対応するのかわからなくなってしまうことがあります。例えば、以下のような tabular において、

```
\begin{tabular}{|c|c|c|c|c|c|c|c|}\hline
氏名&所属&〒&住所&電話&FAX&帰省先&帰省先電話\\ \hline
矢上二郎 & 6 & 223 & 横浜市港北区日吉 & xxx-yyy &
          zzz-www & トンガ & 9876-54321 \\
日吉小僧 & 2 & \multicolumn{2}{c|}{教えない}
          &&&{???}
\\ \hline
\end{tabular}
```

(???) の部分がどの項目なのかすぐに判断するのは難しいでしょう。こんな時は、

```
[prefix] &
... 現在のカラム表示
```

を押すとカーソル位置のカラムがどの項目に該当するかをミニバッファに表示します。tabular/array 環境の第 1 行目を項目名の並びとみなして対応するものを探します。もし項目名として別のものを表示して欲しい場合は、行頭を%にしてダミーの項目並びを作っておくと良いでしょう。

15 おまかせ改行

`tabular[*]`, `array`, `itemize`, `enumerate`, `tabbing` 環境を `begin` 型補完で入力した時、または各環境内で

`ESC RET` ... おまかせ改行

を押すと、その環境に応じた行エントリを次の行に挿入します (`begin` 型補完時に自動挿入されたエントリが不要な場合は `undo` によって消去できます)。例えば、`tabular` 環境では、その環境のカラム数に対応した個数の `&` に加え、行末の `\\` を入れます。この時それ以前に `\hline` があればそれも付け加えます。環境とそれに応じて自動入力するものの対応は以下のようになります。

- `tabular`, `tabular*`, `array`
カラム数-1 だけの `&` と `\\`。必要に応じて `\hline`
- `tabbing`
一行目で定義している `\=` と同じ個数の `\>`。
- `itemize`, `enumerate`, `description`, `list`
`\item` または `item[]`

`tabular` 環境の例のように、本機能は各環境の一行目の内容を参考にして動作するので、なるべく二行目以降で呼び出すようにしてください。

もし、その他の環境、例えば `foo`、に対しておまかせ改行を動作させたい時は、`YaTeX-intelligent-newline-foo` という名前の関数を定義します。定義した関数は、現在の行に改行を挿入した直後の行頭の位置で呼ばれます。関数 `YaTeX-indent-line` を呼ぶと現在の環境のネストに応じた深さにインデントされるので、これと呼んでから何かを挿入するようなコードを書くとよいでしょう。`yatexenv.el`内の関数 `YaTeX-intelligent-newline-itemize` の定義などを参考にしてください。

16 野鳥の動作モード切り替え

[*prefix*] *w*

... 野鳥動作モード切り替えメニュー

で野鳥自身の動作を決定する以下のモードを切り替えます。

- 修正モード
- 野鳥数式モード

修正モードは、開き括弧入力時の処理をコントロールし、修正モード ON の時は開き括弧の入力は開き括弧のみの入力になり、修正モード OFF の時は開き括弧の入力だけで閉じ括弧まで入力します。デフォルト (起動時) の設定は *OFF* です。

野鳥数式モードは、変数 `YaTeX-auto-math-mode` が `nil` の時のみ有効で、このとき ; や : を押した時 (Section 7.8 [Image completion], page 14 参照) に、どのようなイメージ補完を機能させるか、通常のキーとして機能させるかを手動で切り替えます。自動判定が遅いマシンでは `YaTeX-auto-math-mode nil` にセットし、野鳥数式モードを手動で切り替えると良いでしょう。

17 オンラインヘルプ

使おうとする LaTeX コマンドの用法がよく分からない時は、オンラインヘルプをひきましょう。ヘルプに関するキーには以下のものがあります。

```
[prefix] ?
    ... オンラインヘルプ

[prefix] /
    ... オンライン apropos
```

17.1 オンラインヘルプ

「オンラインヘルプ」は、一般的な LaTeX コマンド (デフォルトでカーソル位置のコマンド) に対する説明を隣のバッファに表示します。この時参照されるヘルプ用ファイルには「グローバルヘルプ」と「プライベートヘルプ」の二種類があり、前者は LaTeX の標準コマンドの主なものの説明を含むファイルで、変数 `YaTeX-help-file` の値で指定されます。このファイルは通常公共の場所 (デフォルトで `$EMACSEXECPATH`) に置かれ、誰もがその内容を更新できるように全員に書き込み権が与えられるべきものです。後者は、非標準もしくは個人的なマクロ定義に関する説明が書かれているファイルで、変数 `YaTeX-help-file-private` の値で指定されます。こちらはユーザのホームディレクトリの下などに置かれます。

17.2 オンライン apropos

「オンライン apropos」は GNU Emacs の apropos と同様、ユーザが指定したキーワードを説明文に含む項目すべてを隣のバッファに表示します。

もし、調べようとした LaTeX コマンドに対する説明がヘルプファイル中に見つからなかった場合は、説明文の入力を求めてくるので、可能であれば参考書などを調べてそのコマンドの説明を入力してください。もし、なにか標準的なコマンドに対する説明を書いたならばぜひ私までその説明をお送り下さい。次回の配布に含めたいと思います。

18 インクルード構造ブラウザ

複数のファイルに分割しているドキュメントを書いている場合、

[*prefix*] *d*

... インクルード構造ブラウザ

を押すと、そのドキュメントの親ファイルを聞いて来ます。ここで全てのファイルの親となるファイル(デフォルトが示されているので大抵は RET のみ)を入力するとインクルードしている全てのファイルを解析し、インクルード状況を視覚的に表示します。このバッファでは以下のキー操作が有効です。

<i>n</i>	... 次の行に移動し対応するファイルを隣のバッファに表示
<i>p</i>	... 上の行に移動し対応するファイルを隣のバッファに表示
<i>N</i>	... 同じインクルードレベルの次のファイルに移動
<i>P</i>	... 同じインクルードレベルの前のファイルに移動
<i>j</i>	... 次の行に移動
<i>k</i>	... 上の行に移動
<i>u</i>	... 一代親にあたるファイルに移動
.	... カーソル位置のファイルを隣のバッファに表示
<i>SPC</i>	... 隣のバッファの対応ファイルをスクロールアップ
<i>DEL, b</i>	... 隣のバッファの対応ファイルをスクロールダウン
<	... 隣のバッファの対応ファイルの先頭を表示
>	... 隣のバッファの対応ファイルの末尾を表示
'	... (<や>の後で) 元の表示位置に戻る
<i>RET, g</i>	... カーソル位置のファイルを隣のバッファでオープン
<i>mouse-2</i>	... RET と同じ (ウィンドウ使用時のみ)
<i>o</i>	... 隣のウィンドウに移動
<i>1</i>	... 他のウィンドウを消す
-	... ブラウズウィンドウを小さくする
+	... ブラウズウィンドウを大きくする
?	... ヘルプ表示
<i>q</i>	... 表示前の状態に戻る

ただし、隣のウィンドウのファイルの内容を表示する機能に関しては、対応するファイルをクローズしてしまうとうまく働きませんのでご注意ください。

19 他パッケージとの連携

19.1 gmhist

gmhist.elと gmhist-mh.el をロードしている場合、プレビューコマンドの入力 (`[prefix] tp`)、印刷コマンドの入力 (`[prefix] t1`) の時に独立したヒストリを利用できます。それぞれのプロンプトで、`M-p` を押すと直前に利用したコマンド文字列をくり返し呼び出すことができます。

19.2 min-out

min-out.el (outline-minor-mode) と野鳥を組み合わせて使うことももちろん可能です。設定の方法に関しては `yatexm-o.el` をご覧ください。

20 カスタマイズ

野鳥の動作を制御する種々の変数を独自に設定することにより、補完入力を起動するキーアサインを変えたり、環境名の補完候補をさらに充実させることなどができます。

20.1 lisp 変数

例えば prefix キーを C-c 以外のキーにしたい場合は、YaTeX-prefix に prefix キーにしたいシンボルを定義してください。さらに、「C-c 英字」というキーバインドは独自の関数が割り当ててあるので使いたくない。このような時は、YaTeX-inhibit-prefix-letter を t に設定することにより、C-c 英字…のバインドが全て、対応する C-c C-英字…に変わります (ただし、begin 型 large 型補完の大文字起動によるリジョン指定は可能なままです。これも無効にしたい場合は t ではなく 1 にセットして下さい。)

20.1.1 カスタマイズ変数一覧

yatex-mode における次の変数がカスタマイズ可能です。~/emac で setq しておけば、そちらの定義が優先されます。括弧の中はデフォルト値です。実際に変数の値を変更する場合は M-x describe-variable で変数の詳細な説明を参照してください。

YaTeX-prefix	[Variable]
yatex-mode 中のプリフィクスキー (\C-c)	
YaTeX-inhibit-prefix-letter	[Variable]
prefix キーの直後のキーバインドで 英字 のものを C-英字 に変更 (nil)	
YaTeX-fill-prefix	[Variable]
本文を書く時の行頭に挿入する接頭辞すなわち fill-prefix ("(nil))	
YaTeX-user-completion-table	[Variable]
学習した LaTeX コマンド保存ファイル名 ("~/yatexrc")	
YaTeX-kanji-code	[Variable]
文書を作成する時の漢字コード 1=Shift JIS, 2=JIS, 3=EUC (2 (MS-DOS では 1))	
tex-command	[Variable]
LaTeX タイプセッタコマンド名 ("jlatex")	
dvi2-command	[Variable]
プレビューアコマンド名 ("xdvi -geo +0+0 -s 4")	
dviprint-command-format	[Variable]
dvi ファイルの印刷に使われるコマンド式 ("dvi2ps %f %t %s lpr")	
dviprint-from-format	[Variable]
上の%fに相当する開始ページ指定書式、%b が開始ページ番号に変わる ("-f %b")	
dviprint-to-format	[Variable]
%t に相当する終了ページ指定書式、%eが終了ページ番号に変わる ("-t %e")	
makeindex-command	[Variable]
makeindex コマンド ("makeindex" (MS-DOS では"makeind"))	

section-name	[Variable]
最初の section 型補完の規定値 ("documentstyle")	
env-name	[Variable]
最初の begin 型補完の規定値 ("document")	
fontsize-name	[Variable]
最初の large 型補完の規定値 ("large")	
single-command	[Variable]
最初の maketitle 型補完の規定値 ("maketitle")	
YaTeX-need-nonstop	[Variable]
\nonstopmode{}を自動的に付加するか (nil)	
latex-warning-regexp	[Variable]
latex コマンドの出力するウォーニング行の正規表現 ("line.* [0-9]*")	
latex-error-regexp	[Variable]
同じくエラー行の正規表現 ("l\\.[1-9][0-9]*")	
latex-dos-emergency-message	[Variable]
MS-DOS 上で動作する latex コマンドが、エラーにより停止するとき出力するメッセージ ("Emergency stop")	
latex-message-kanji-code	[Variable]
jlatex コマンドの出力するメッセージの漢字コード. タイプセットバッファの出力が化ける時は、これを設定する (2, Nemacs でのみ有効)	
NTT-jTeX	[Variable]
古い NTT-jTeX 使用時のようにインデントした行の先頭と前の行の (タイプセット後の) 字間が空いてしまうのを嫌う場合は t にする (nil)	
YaTeX-item-regexp	[Variable]
item の桁揃えの時に用いる、item の正規表現 ("\\(sub\\)*item")	
YaTeX-verb-regexp	[Variable]
verb コマンドの正規表現。先頭の\\(sub\\)はつけない ("verb*?\\ path")	
YaTeX-nervous	[Variable]
ローカル辞書を用いる時 t (t)	
YaTeX-sectioning-regexp	[Variable]
セクション区切り設定コマンドの正規表現 ("\\(part\\ chapter*?\\ \\(sub\\)*\\(section\\ p	
YaTeX-fill-inhibit-environments	[Variable]
fill を抑止する環境名のリスト ('("tabular" "tabular*" "array" "picture" "eqnarray" "eqnarray*" "equation" "math" "displaymath" "verbatim" "verbatim*"))	
YaTeX-uncomment-once	[Variable]
領域 uncomment で行頭の複数の%を全て削除するか (nil)	

YaTeX-close-paren-always	[Variable]
開き括弧の入力で常に閉じ括弧を入力する (t)	
YaTeX-auto-math-mode	[Variable]
数式モードの切り替えを自動的に行う (t)	
YaTeX-math-key-list-private	[Variable]
数式イメージ補完で用いる (プリフィクスキー、対応補完テーブル) の alist (nil)。補完テーブルの書き方については <code>yatexmth.el</code> を参照。	
YaTeX-default-pop-window-height	[Variable]
1 画面の時にタイプセットバッファを初めて作成する時の高さ。数値で行数、数字文字列で Emacs ウィンドウに対する百分率 (10)	
YaTeX-help-file	[Variable]
共用ヘルプファイル (<code>\$EMACS/etc/YATEXHLP.jp</code>)	
YaTeX-help-file-private	[Variable]
個人用ヘルプファイル (" <code>~/YATEXHLP.jp</code> ")	
YaTeX-no-begend-shortcut	[Variable]
<code>[prefix] b ??</code> のショートカットを使わず、 <code>[prefix] b</code> だけで補完入力に入る (nil)	
YaTeX-hilit-pattern-adjustment-private	[Variable]
正規表現とそれにマッチするものの論理的意味をシンボルであらわしたもののリスト…のリスト。hilit19 を組み込んでいる時のみ有効。詳しくは (<code>assq 'yatex-mode hilit-patterns-alist</code>) した結果と、変数 <code>YaTeX-hilit-pattern-adjustment-default</code> の値 (と場合によっては hilit19 のドキュメント) を参照せよ。	
YaTeX-sectioning-level	[Variable]
LaTeX のセクション単位宣言コマンドとその論理的高さの alist。	
YaTeX-hierarchy-ignore-heading-regexp	[Variable]
Hierarchy バッファは通常ファイルヘッダとして、LaTeX のセクション宣言コマンドの引数を検索し、それがなければコメント行を探すが、その際にヘッダとしては意味を持たないパターンをこの変数に設定する。デフォルトでは RCS ヘッダとモード指定行 (<code>-* xxx -*</code>) が設定されている。	
YaTeX-skip-default-reader	[Variable]
Non-nil に設定すると section 型コマンドの引数入力時、アドイン関数がなければミニバッファでの読み込みをせずに入力を完了させる (nil)	
YaTeX-create-file-prefix-g	[Variable]
<code>\include</code> などで <code>prefix g</code> した時に、ジャンプ先が存在しないファイルであってもオープンする (nil)	
YaTeX-simple-messages	[Variable]
各種補完時のメッセージ出力を簡素化する (nil)	
YaTeX-hilit-sectioning-face	[Variable]
色付けが有効な時の <code>\part</code> の色 ('yellow/dodgerblue yellow/slateblue)。リストの第一要素は <code>hilit-background-mode</code> が 'light の時の、第二要素は 'dark の時の <code>\chapter</code> の色で、文字色/背景色 のように指定する。	

YaTeX-hilit-sectioning-attenuation-rate [Variable]
 色付けが有効な時の、`\ subparagraph` の色を `\ chapter` の濃度の何%薄くしたものに
 するか ('(15 40) `YaTeX-hilit-sectioning-face` の項参照。

YaTeX-use-AMS-LaTeX [Variable]
 AMS-LaTeX を使用する場合は `t` に設定する (`nil`)

YaTeX-template-file [Variable]
 新規ファイル作成時に自動挿入するファイル名 (`~/work/template.tex`)

20.1.2 カスタマイズ変数設定例

たとえば、`prefix` キーとして `ESC` を使用し、新たな補完候補を格納するファイルを、`~/src/emacs/yatexrc` にし、行頭の `prefix` をタブ文字一つに変えたいときは、

```
(setq YaTeX-prefix "\e"
      YaTeX-user-completion-table "~/src/emacs/yatexrc"
      YaTeX-fill-prefix "    ")
```

を `~/ .emacs` に加えます。

20.1.3 hook 変数

また、hook 変数 `yatex-mode-hook`, `yatex-mode-load-hook` を用意しています。すべての `yatex-mode` のバッファで作用させたいものは、`yatex-mode-hook` に記述し、`yatex.el` をロードする時だけ作用させたいものは `yatex-mode-load-hook` に記述します。例えば、`outline-minor-mode` を利用する場合、それぞれのバッファで `outline-minor-mode` を有効にしたいので、`yatex-mode-hook` を次のように設定します。

```
(setq yatex-mode-hook
      '(lambda () (outline-minor-mode t)))
```

逆に、独自のキー定義を行いたい時などは、`yatex-mode-load-hook` を利用します。例えば、`begin` 型補完において、`document` や、`enumerate` 以外の環境名もショートカットキーで入れたいなどという時は、次のようにします。以下の例は、`[prefix] ba` で `\begin{abstract}`, `\end{abstract}` を挿入します。

```
(setq yatex-mode-load-hook
      '(lambda() (YaTeX-define-begend-key "ba" "abstract")))
```

なお、新たなキーの定義には、関数 `YaTeX-define-key` `YaTeX-define-begend-key` を利用するようにしてください。

20.1.4 hook 用ファイル

変数 `yatex-mode-load-hook` で定義する内容が多い時は、`yatexhks.el` というファイルを作り、その中に野鳥関連の設定を書く事で、初期化の時に自動的にロードします。

20.2 付加関数 (アドイン関数)

各種補完時に、環境名やコマンド名に応じたきめ細やかな補完入力機能を実現するための関数を作成することができます。この関数の作成方法や、組み込み方法に関しては、`yatexadd.doc` をご覧ください。

21 その他

野鳥の標準の LaTeX コマンドの辞書には、作者が頻繁に使うものしか登録されていません。これは、補完候補に使いそうもないコマンドが存在して、補完したいコマンドを出すまでのストローク数を増やしてしまう事を防止するためです。標準辞書にないコマンドも、できるだけ補完入力方式を利用し、ユーザ辞書を充実させることで、あなたの LaTeX スタイルにあった野鳥へと育っていくことでしょう。

22 取り扱い

本プログラムはフリーソフトウェアです。本プログラムを使用して生じたいかなる結果に対しても作者は責任を負わないこととします。転載等に関しては制限いたしません。常識的に扱ってください。また、使用している旨をメールでお知らせいただくと、作者は喜んでサポートに励むことでしょう。

苦情、希望、バグ報告、感想等は歓迎いたします。連絡は yuuji@ae.keio.ac.jp または、pcs39334@ascinet.or.jp まで (1995年3月現在)。継続的に使用してくださる方はメイリングリスト「fj野鳥の会」に是非加入してください。加入方法については本パッケージの docs/qanda ファイルの「その他」の章を御覧ください。

仕様は、予告なく確実に (気分次第で) 変更されます:-p。

広瀬雄二

索引

%

%#BEGIN	7
%#END	7
%#記法自体の編集 [%#きほうした いのへんしゅう]	8

&

&入力 [&にゆうりよく]	24
---------------------	----

.

.emacs	4
.yatexrc	16

:

:	14
---------	----

;

;	14
; 自身 [: ししん]	14

A

apropos	26
auto-mode-alist	4
autoload	4

B

begin 型補完 [begin かたほかん]	9
-------------------------------	---

C

C-c	5
-----------	---

D

Demacs	1
dvi2-command	29
dviprint-command-format	29
dviprint-from-format	29
dviprint-to-format	29

E

end 補完 [end ほかん]	13
env-name	30

F

fontsize-name	30
---------------------	----

G

ghostview	7
gmhist	28

H

hook 変数 [hook へんすう]	32
---------------------------	----

I

includeonly	22
Install	4
item などの桁揃え [item などのけたそろえ]	21

J

jlatex	5
--------------	---

L

large 型補完 [large かたほかん]	12
LaTeX	1
latex-dos-emergency-message	30
latex-error-regexp	30
latex-message-kanji-code	30
latex-warning-regexp	30
leftarrow	14
lpr format	8
lpr ふおーまっと [lpr ふおおまっと]	8
lpr フォーマット [lpr ふおおまっと]	8
lpr フォーマットの変更 [lpr ふおおま つとのへんこう]	8

M

M-C-@	19
M-C-a	19
M-C-e	19
M-q	21
makeindex-command	29
maketitle 型補完 [maketitle かたほかん]	13
min-out	28
Mule	1

N

NTT-jTeX	30
NTT-jTeX[えぬていいていいしえいてつく]	21

P

prefix &	23
prefix ,	17
prefix	17
prefix /	26
prefix <	17
prefix >	17
prefix ?	26
prefix a	13
prefix b	9
prefix c	20
prefix d	27
prefix e	13
prefix g	18
prefix i	21
prefix k	20
prefix key	5
prefix l	12
prefix m	13
prefix s	10
prefix SPC	13
prefix w	25
prefix キー変更 [prefix きいへんこう]	29

S

section-name	30
section 型補完 [section かたほかん]	10
single-command	30

T

tex-command	29
-------------------	----

Y

YaTeX-auto-math-mode	31
YaTeX-close-paren-always	31
YaTeX-create-file-prefix-g	31
YaTeX-default-pop-window-height	31
YaTeX-fill-inhibit-environments	30
YaTeX-fill-prefix	29
YaTeX-help-file	26
YaTeX-help-file	31
YaTeX-help-file-private	26
YaTeX-help-file-private	31
YaTeX-hierarchy-ignore-heading-regexp	31
YaTeX-hilit-pattern-adjustment-private	31
YaTeX-hilit-sectioning-attenuation-rate	32
YaTeX-hilit-sectioning-face	31
YaTeX-inhibit-prefix-letter	29
YaTeX-item-regexp	21
YaTeX-item-regexp	30
YaTeX-kanji-code	29
YaTeX-math-key-list-private	31
YaTeX-math-sign-alist-private	14
yatex-mode-hook	32
yatex-mode-load-hook	32

YaTeX-need-nonstop	30
YaTeX-nervous	16
YaTeX-nervous	30
YaTeX-no-begend-shortcut	31
YaTeX-prefix	29
YaTeX-sectioning-level	31
YaTeX-sectioning-regexp	30
YaTeX-simple-messages	31
YaTeX-skip-default-reader	31
YaTeX-template-file	32
YaTeX-uncomment-once	30
YaTeX-use-AMS-LaTeX	32
YaTeX-user-completion-table	29
YaTeX-verb-regexp	30

∞

∞ [むけんたい]	14
-----------------	----

Σ

Σ [しくま]	14
---------------	----

あ

アウトライン [あうとらいん]	12
アクセント記号補完 [あくせんときこうほかん] ...	13

い

イメージ補完 [いめえしほかん]	14
インクルード構造 [いんくるうとこうそう]	27
インストール [いんすとおる]	4

え

エラー修正 [えらあしゆうせい]	5
------------------------	---

お

おまかせ改行 [おまかせかいぎよう]	24
オンラインヘルプ [おんらいんへるふ]	26
お絵描きツール起動 [おえかきつうるぎとう]	18

か

カーソルジャンプ [かあそるしやんぷ]	18
カスタマイズ [かすたまいす]	29
カスタマイズ変数一覧 [かすたまいす へんすういちらん]	29

き

- キーアサイン [きいあざいん]..... 29
- キーワード検索 [きいわあとけんさく]..... 26
- ギリシャ文字補完 [きりしやもしほかん]..... 14

く

- グローバルヘルプ [くろおはるへるふ]..... 26

こ

- ここはどこ?[ここはどこ?]..... 23
- コマンドヒストリ [こまんとひすとりに]..... 28
- コメントアウト [こめんとあうと]..... 17

し

- ジャンプ [しやんぷ]..... 12

せ

- セクション区切り [せくしよんくきり]..... 12
- セクション区切り一覧バッファ [せくしよんくきりいちらんはつふあ]..... 12

た

- タイプセッタ [たいふせつた]..... 5
- タイプセッタの使い分け [たいふせつたのつかいわけ]..... 7
- タイプセッタ起動 [たいふせつたぎとう]..... 5
- タイプセットエラー [たいふせつとえらあ]..... 5

は

- パラグラフの桁揃え [はらくらふのけたそろえ] ... 21

ひ

- ヒストリ [ひすとりに]..... 28

ふ

- フォント指定子 [ふおんとしていし]..... 12
- プライベートヘルプ [ふらいへえとへるふ]..... 26
- プリントアウト [ふりんとあうと]..... 5
- プレビューア [ふれひゆうあ]..... 5
- ブロック [ふろつく]..... 10

へ

- ページ確認省略 lpr 起動 [へえしかくにんしようりやく lpr きとう]..... 6

も

- モード切り替え [もうときりかえ]..... 25

や

- やちよう [やちよう]..... 1

ゆ

- ユーザ辞書 [ゆうさししよ]..... 10

ろ

- ローカル辞書 [ろおかるししよ]..... 16

引

- 引数 [ひきすう]..... 11
- 引数の個数を変える [ひきすうのこすうをかえる]..... 11

欧

- 欧文 [おうふん]..... 13

括

- 括弧 [くくろ]..... 11, 12

環

- 環境の削除 [かんきようのさくしよ]..... 20
- 環境の先頭へ [かんきようのせんとうへ]..... 19
- 環境の末尾へ [かんきようのまつひへ]..... 19
- 環境をマーク [かんきようをまあく]..... 19
- 環境名の変更 [かんきようめいのへんこう]..... 20
- 環境名の補完 [かんきようめいのほかん]..... 9

起

- 起動するコマンドを変える [きとうするこまんとをかえる]..... 7

桁

- 桁揃え [けたそろえ]..... 21

現

- 現在のカラム表示 [けんさいのからむひようし] ... 23

固

固定領域のタイプセット [こていりょうい
きのたいふせつと]..... 7

再

再帰補完 [さいきほかん]..... 11

出

出力開始ページ [しゅつりよくかいしへえし]..... 6
出力終了ページ [しゅつりよくしゅ
うりょうへえし]..... 6

勝

勝手に includeonly [かつてに includeonly]..... 22

随

随時補完 [すいしほかん]..... 13

数

数式モード [すうしきもおと]..... 14
数式記号イメージ補完 [すうしききこ
ういめえしほかん]..... 14

設

設定例 [せつていれい]..... 32

他

他のファイルの手直し [ほかのふあ
いるのでなおし]..... 22
他パッケージとの連携 [たはつけえ
しとのれんけい]..... 28

長

長いファイルの編集 [ながいふあ
いのへんしゅう]..... 8

入

入力ファイル分割 [にゅうりよくふ
あいるふんかつ]..... 7

複

複雑な tabular [ふくさつな tabular]..... 23

文

文字サイズ指定子 [もしさいすしていし]..... 12

閉

閉じ込める [としこめる]..... 10

変

変更/削除 [へんこう/さくじよ]..... 20

矢

矢印 [やしるし]..... 14

領

領域決定規則 [りょういきけつていきそく]..... 8

論

論理階層 [ろんりかいそう]..... 12

Table of Contents

1	はじめに	1
2	本マニュアル参照上の注意	2
3	主な機能	3
4	起動法	4
4.1	野鳥起動のための設定.....	4
4.2	タイプセッタ/プレビューア環境の設定.....	4
5	latex コマンド起動	5
5.1	タイプセッタ起動.....	5
5.1.1	領域タイプセット.....	5
5.2	プレビューア起動.....	6
5.3	プリントアウト.....	6
6	%#記法	7
6.1	タイプセット用コマンド変更.....	7
6.2	入力ファイル分割.....	7
6.3	領域の固定.....	7
6.4	lpr フォーマット.....	8
6.5	%#記法自体の編集.....	8
7	補完入力	9
7.1	begin 型補完.....	9
7.1.1	既にしたテキストを環境で括る.....	10
7.2	section 型補完.....	10
7.2.1	2 個以上の引数をとる section 型コマンド.....	11
7.2.2	既にしたテキストを括る.....	11
7.2.3	再帰補完.....	11
7.2.4	セクション区切りのアウトライン表示.....	12
7.3	large 型補完.....	12
7.3.1	既にした文字を括る.....	12
7.4	maketitle 型補完.....	13
7.5	随時補完.....	13
7.6	end 補完.....	13
7.7	アクセント記号補完.....	13
7.8	数式記号イメージ補完.....	14
7.9	ギリシャ文字補完.....	14

8	ローカル辞書	16
9	コメントアウト	17
10	カーソルジャンプ	18
10.1	対応オブジェクトへのジャンプ	18
10.2	お絵描きツール起動	18
10.3	メインファイルへのジャンプ	19
10.4	環境を単位としたジャンプ	19
10.5	最後の補完位置へのジャンプ	19
11	変更/削除	20
11.1	LaTeX コマンドの変更	20
11.2	LaTeX コマンドの削除	20
12	桁揃え	21
12.1	item の桁揃え	21
12.2	パラグラフの桁揃え	21
13	勝手に includeonly	22
14	ここはどこ?	23
15	おまかせ改行	24
16	野鳥の動作モード切り替え	25
17	オンラインヘルプ	26
17.1	オンラインヘルプ	26
17.2	オンライン apropos	26
18	インクルード構造ブラウザ	27
19	他パッケージとの連携	28
19.1	gmhist	28
19.2	min-out	28
20	カスタマイズ	29
20.1	lisp 変数	29
20.1.1	カスタマイズ変数一覧	29
20.1.2	カスタマイズ変数設定例	32
20.1.3	hook 変数	32
20.1.4	hook 用ファイル	32
20.2	付加関数 (アドイン関数)	32

21	その他	33
22	取り扱い	34
	索引	35

