Yet Another tex-mode for emacs

Wild Bird

// YaTeX //

1 What is YaTeX?

YaTeX automates typesetting and previewing of LaTeX and enables completing input of LaTeX mark-up command such as `\begin{}`..`\end{}`.

YaTeX also supports Demacs which runs on MS-DOS(386), Mule (Multi Language Enhancement to GNU Emacs), and latex on DOS.

2 Main features

- Invocation of typesetter, previewer and related programs(*C-c t*)
- Typesetting on static region which is independent from point
- Semiautomatic replacing of `\includeonly`
- Jumping to error line(*C-c '*)
- Completing-read of LaTeX commands such as `\begin{}`, `\section` etc. (*C-c b*, *C-c s*, *C-c l*, *C-c m*)
- Enclosing text into LaTeX environments or commands (*AboveKeyStrokes* after region setting)
- Displaying the structure of text at entering sectioning commands
- Lump shifting of sectioning commands (Section 6.2.1 [view-sectioning], page 11)
- Learning unknown/new LaTeX commands for the next completion
- Argument reading with a guide for complicated LaTeX commands
- Generating argument-readers for new/unsupported commands(‘`yatexgen`’)
- Quick changing or deleting of LaTeX commands(*C-c c*, *C-c k*)
- Jumping from and to inter-file, begin<->end, ref<->label(*C-c g*)
- Blanket commenting out or uncommenting (*C-c >*, *C-c <*, *C-c ,*, *C-c .*)
- Easy input of accent mark, math-mode’s commands and Greek letters (*C-c* `a`, `;`, `:`)
- Online help for the popular LaTeX commands (*C-c ?*, *C-c /*)
- Document files hierarchy browser (*C-c d*)
- Adding automatically \usepackage corresponding to inputting LaTeX macro with completion
- Allow you to forget creating \label{}s, \ref{} or \cite{} completion automatically generate labels.
- \includegraphics by Drag&Drop of image file

3 Installation

Put next two expressions into your '`~/.emacs`'.

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

Next, add certain path name where you put files of YaTeX to your load-path. If you want to put them in '`~/src/emacs`', write

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

in your '`~/.emacs`'

Then, yatex-mode will be automatically loaded when you visit a file which has extension '`.tex`'. If yatex-mode is successfully loaded, mode string on mode line will be turned to "YaTeX".

4 Typesetting

The prefix key stroke of yatex-mode is *`C-c`* (Press 'C' with Control key) by default. If you don't intend to change the prefix key stroke, assume all *`[prefix]`* as *`C-c`* in this document. These key strokes execute typeset or preview command.

`[prefix] t j`
: ... invoke typesetter

`[prefix] t r`
: ... invoke typesetter on region

`[prefix] t e`
: ... 'on-the-fly preview' on current environment or whole portion of current formulas in math-mode

`[prefix] t d`
: ... invoke dvipdfmx after successful typesetting

`[prefix] t k`
: ... kill current typesetting process

`[prefix] t b`
: ... invoke bibtex

`[prefix] t i`
: ... invoke makeindex

`[prefix] t d`
: ... invoke latex && dvipdfmx

`[prefix] t p`
: ... preview

`[prefix] t l`
: ... lpr dvi-file

`[prefix] t s`
: ... search current string on xdvi-remote

4.1 Calling typesetter

Typing *`[prefix] t j`*, the current editing window will be divided horizontally when you invoke latex command, and log message of LaTeX typesetting will be displayed in the other window; called typesetting buffer. The typesetting buffer automatically scrolls up and traces LaTeX warnings and error messages. If you see latex stopping by an error, you can send string to latex in the typesetting buffer.

If an error stops the LaTeX typesetting, this key stroke will move the cursor to the line where LaTeX error is detected.

`[prefix] '`
`([prefix]+single quotation)`
: ... jump to the previous error or warning

If you find a noticeable error, move to the typesetting buffer and move the cursor on the line of error message and type *SPACE* key. This makes the cursor move to corresponding source line.

YaTeX-typeset-region invoked by *[prefix] tr* call typesetter for region. The region is specified by standard point and mark, or by `%#BEGIN` and `%#END` marks. Selected region will be copied to the temporary file '`texput.tex`' with the same preamble as the main file of current editing sources. Be sure to put all local macro settings in preamble, not after `\begin{document}`. The method of specification of the region is shown in the section See Chapter 5 [%#notation], page 6.

The documentclass for typeset-region is the same as that of editing file if you edit one file, and is the same as main file's if you edit splitting files.

The *[prefix] te* key automatically marks current inner environment or inner math mode or paragraph, and then call typeset-region with marked region. This is convenient to quick view of current tabular environment or current editing formulas. If running Emacs has the ability of displaying images, typeset image will be shown in the next window. Further more, if you modify the content within that environment, YaTeX performs 'on-the-fly' preview that automatically update preview image as you typed.

If your Emacs does not supply on-the-fly preview, keeping previewer window for '`texput.dvi`' is handy for debugging. Since *[prefix] te* selects the inner-most environment as region, it is not suitable for partial typesetting of doubly or more composed environment. If you want to do partial typesetting for a nested environment, use *[prefix] tr* for static-region, which is described in the section See Chapter 5 [%#notation], page 6.

4.2 Calling previewer

[prefix] t p invokes the TeX previewer. And if you are using xdvi-remote, which can be controled from other terminals, *[prefix] t s* enables you to search current string at the cursor on the running xdvi window.

4.3 Printing out

When you type `[preifx] t l`, YaTeX asks you the range of dvi-printing by default. You can skip this by invoking it with universal-argument as follows:

```
          C-u [prefix] tl
```

5 %# notation

You can control the typesetting process by describing `%#` notations in the source text.

5.1 To change the 'latex' command or to split a source text.

To change the typesetting command, write

```
%#!latex-big
```

anywhere in the source text. This is useful for changing typesetter.

5.2 Splitting input files

And if you split the source text and edit subfile that should be included from main text.

```
%#!latex main.tex
```

will be helpful to execute latex on main file from sub text buffer. Since this command line after *`%#!`* will be sent to shell literally, next description makes it convenient to use ghostview as dvi-previewer.

```
%#!latex main && dvi2ps main.dvi > main
```

Note that YaTeX assumes the component before the last period of the last word in this line as base name of the main LaTeX source. The `%f` notation in this line is replaced by main file name, and `%r` replaced by root name of main file name. If you specify `%f` or `%r`, YaTeX always ask you the name of main file at the first typesetting.

To make best use of the feature of inter-file jumping by *`[prefix] g`* (see Chapter 9 [Cursor jump], page 17), take described below into consideration.

- You can put split texts in sub directory, but not in sub directory of sub directory.
- In the main text, specify the child file name with relative path name such as \include{chap1/sub}, when you include the file in a sub-directory.
- In a sub-text, write `%#!latex main.tex` even if '`main.tex`' is in the parent directory(not %#!latex ../main.tex).

5.3 Static region

Typeset-region by *`[prefix] tr`* passes the region between point and mark to typesetting command by default. But when you want to typeset static region, enclose the region by `%#BEGIN` and `%#END` as follows.

```
%#BEGIN
  TheRegionYouWantToTypesetManyTimes
%#END
```

This is the rule of deciding the region.

1. If there exists %#BEGIN before point,
    1. If there exists %#END after %#BEGIN,
        - From %#BEGIN to %#END.
    2. If %#END does not exist after %#BEGIN,
        - From %#BEGIN to the end of buffer.

2. If there does not exist %#BEGIN before point,
   - Between point and mark(standard method of Emacs).

It is useful to write `%#BEGIN` in the previous line of \begin and `%#END` in the next line of `\end` when you try complex environment such as 'tabular' many times. It is also useful to put only `%#BEGIN` alone at the middle of very long text. Do not forget to erase `%#BEGIN` `%#END` pair.

5.4 Lpr format

Lpr format is specified by three Lisp variables. Here are the default values of them.

```
(1)dviprint-command-format
          "dvi2ps %f %t %s | lpr"

(2)dviprint-from-format
          "-f %b"

(3)dviprint-to-format
          "-t %e"
```

On YaTeX-lpr, `%s` in (1) is replaced by the file name of main text, `%f` by contents of (2), %t by contents of (3). At these replacements, `%b` in (2) is also replaced by the number of beginning page, `%e` in (3) is replaced by the number of ending page. But `%f` and `%t` are ignored when you omit the range of print-out by *`C-u [prefix] tl`*.

If you want to change this lpr format temporarily, put a command such as follows somewhere in the text:

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

And if you want YaTeX not to ask you the range of printing out, the next example may be helpful.

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

5.5 Controlling which command to invoke

These %# notation below can control which command to invoke for LaTEX related process.

`%#PREVIEW`
: ... Command line for DVI viewing ([prefix] t p)

`%#MAKEINDEX`
: ... Command line for makeindex ([prefix] t i)

`%#BIBTEX` ... Command line for bibtex ([prefix] t b)

`%#DVIPDF` ... Command line for dvipdf(mx) ([prefix] t b)

`%#LPR` ... Command line for printing out([prefix] t l)

`%#PDFVIEW`
: ... Command line for PDF viewing

`%#IMAGEDPI`
: ... DPI value for converting to on-the-fly prewview image

If you want to invoke "makeidx hogehoge" to update index, put the next line some upper place in the source, for example.

```
%#MAKEINDEX makeidx hogehoge
```

5.6 Editing %# notation

To edit `%#` notation described above, type

`[prefix] %`
    ... editing %# notation menu

and select one of the entry of the menu as follows.

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

Type *`!`* to edit `%#!` entry, `b` to enclose the region with `%#BEGIN` and `%#END`, and `l` to edit `%#LPR` entry. When you type *`b`*, all `%#BEGIN` and `%#END` are automatically erased.

6 Completion

YaTeX makes it easy to input the LaTeX commands. There are several kinds of completion type, begin-type, section-type, large-type, etc...

6.1 Begin-type completion

"Begin-type completion" completes commands of `\begin{env}` ... `\end{env}`. All of the begin-type completions begin with this key sequence.

[prefix] b
: . . . start begin-type completion

An additional key stroke immediately completes a frequently used LaTeX `\begin{}`...`\end{}` environment.

[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}`

Any other LaTeX environments are made by completing-read of the Emacs function.

[prefix] b SPACE
: . . . begin-type completion

The next message will show up in the minibuffer

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

by typing *[prefix] b*. Put the wishing environment with completion in the minibuffer, and `\begin{env}`...`\\end{env}` will be inserted in the LaTeX source text. If the environment you want to put does not exist in the YaTeX completion table, it will be registered in the user completion table. YaTeX automatically saves the user completion table in the user dictionary file at exiting of emacs.

At the completion of certain environments, the expected initial entry will automatically inserted such as `\item` for `itemize` environment. If you don't want the entry, it can be removed by undoing.

If you want to enclose some paragraphs which have already been written into environment, invoke the begin-type completion right after region marking.

If you set `transient-mark-mode` to `nil` in your '`~/.emacs`', typing *C-space* (`set-mark-command`) twice turns `transient-mark-mode` on temporarily. Then, type call begin-type completion to enclose text into a environment.

6.2 Section-type completion

"Section-type completion" completes section-type commands which take an argument or more such as `\section{foo}`. To invoke section-type completion, type

[prefix] s
: . . . section-type completion

then the prompt

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

will show up in the minibuffer. Section-type LaTeX commands are completed by space key, and the default value is selected when you type nothing in the minibuffer.

Next,

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

prompts you the argument of section-type LaTeX command. For example, the following inputs

```
\???{} (default documentclass): section
\section{???}: Hello world.
```

will insert the string

```
\section{Hello world.}
```

in your LaTeX source. When you neglect argument such as

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

YaTeX puts

```
\vspace*{}
```

and move the cursor in the braces.

In LaTeX command, there are commands which take more than one arguments such as `\addtolength{\topmargin}{8mm}`. To complete these commands, invoke section-type completion with universal argument as,

```
C-u 2 [prefix] s (or ESC 2 [prefix] s)
```

and make answers in minibuffer like this.

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

`\addtolength` and the first argument `\topmargin` can be typed easily by completing read. Since YaTeX also learns the number of arguments of section-type command and will ask that many arguments in future completion, you had better tell the number of arguments to YaTeX at the first completion of the new word. But you can change the number of arguments by calling the completion with different universal argument again.

Invoking section-type completion with `[Prefix] S` (Capital 'S') includes the region as the first argument of section-type command.

The section/large/maketitle type completion can work at the prompt for the argument of other section-type completion. Nested LaTeX commands are efficiently read with the recursive completion by typing YaTeX's completion key sequence in the minibuffer.

6.2.1 view-sectioning

In the minibuffer at the prompt of section-type command completion, typing *`C-v`* shows a list of sectioning commands in source text(The line with `<<--` mark is the nearest sectioning command). Then, default sectioning command appears in the minibuffer. You can go up/down sectioning command by typing *`C-p`*/*`C-n`*, can scrolls up/down the listing buffer by *`C-v`*/*`M-v`*, and can hide sectioning commands under certain level by 0 through 6. Type *`?`* in the minibuffer of sectioning prompt for more information.

You can generate this listing buffer (`*Sectioning Lines*` buffer) by typing

`M-x YaTeX-section-overview`
: . . . Generate *Sectioning Lines* buffer

from the LaTeX source buffer. In this listing buffer, typing *`u`* on the sectioning command shifts up the corresponding sectioning command in source text and *`d`* shifts down. After marking lines in the listing buffer, typing *`U`* shifts up all sectioning commands in the region, and *`U`* shifts down. Here are all the key bindings of `*Sectioning Lines*` buffer.

`SPC`	. . . Jump to corresponding source line
`.`	. . . Display corresponding source line

u	. . . Shift up a sectioning line
d	. . . Shift down a sectioning line
U	. . . Shift up sectioning lines in region
D	. . . Shift down sectioning lines in region
0...6	. . . Hide sectioning commands whose level is lower than n

6.3 Label Generation

When you want to type-in references of `\ref` or `\cite`, all you have to do is type *[prefix] s ref* without adding labels beforehand. You will see possible LaTeX-counters in the next window even if some counter does not have `\label`. Selecting the counter will automatically set the label to that counter.

All possible counter list in the buffer tends to be large. You can reduce the number of list by filtering type of counters by key-commands as follows.

M-a	. . . Show all(disable filtering)
M-c	. . . Captions only
M-e	. . . equations (with counters) only
M-i	. . . numbers items only
M-s	. . . sections only
M-m	. . . other counters only

6.4 Large-type completion

"Large-type completion" inputs the font or size changing descriptions such as `{\large }`. When you type

[prefix] l
. . . large-type completion

the message in the minibuffer

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

prompts prompts you large-type command with completing-read. There are TeX commands to change fonts or sizes, `it`, `huge` and so on, in the completion table.

Region-based completion is also invoked by calling completion after region activated.

6.5 Maketitle-type completion

We call it "maketitle-type completion" which completes commands such as `\maketitle`. Take notice that maketitle-type commands take no arguments. Then, typing

[prefix] m
. . . maketitle-type completion

begins maketitle-completion. Above mentioned method is true for maketitle-completion, and there are LaTeX commands with no arguments in completion table.

6.6 Arbitrary completion

You can complete certain LaTeX command anywhere without typical completing method as described, by typing

[prefix] SPC
: . . . arbitrary completion

after the initial string of LaTeX command that is preceded by `\`.

6.7 End completion

YaTeX automatically detects the opened environment and close it with `\\end{environment}`. Though proficient YaTeX users never fail to make environment with begin-type completion, some may begin an environment manually. In that case, type

[prefix] e
: . . . `end` completion

at the end of the opened environment.

6.8 Accent completion

When you want to write the European accent marks(like `` \`{o} ``),

[prefix] a
: . . . accent completion

shows the menu

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

in the minibuffer. Chose one character or corresponding numeric, and you will see

```
\`{}
```

in the editing buffer with the cursor positioned in braces. Type one more character 'o' for example, then

```
\`{o}
```

will be completed, and the cursor gets out from braces.

6.9 Image completion of mathematical sign

Arrow marks, sigma mark and those signs mainly used in the TeX's math environment are completed by key sequences which imitate the corresponding symbols graphically. This completion only works in the math environment. YaTeX automatically detects whether the cursor located in math environment or not, and change the behavior of key strokes *;* and *:*.

By the way, we often express the leftarrow mark by '`<-`' for example. Considering such image, you can write `\leftarrow` by typing `<-` after *;* (semicolon) as a prefix. In the same way, `\longleftarrow` (`<--`) is completed by typing *;<--*, infinity mark which is imitated by `oo` is completed by typing *;oo*.

Here are the sample operations in YaTeX math-mode.

INPUT	Completed LaTeX commands
`; < -`	`\leftarrow`
`; < - -`	`\longleftarrow`
`; < - - >`	`\longleftrightarrow`
`; o`	`\circ`
`; o o`	`\infty`

In any case, you can quit from image completion and can move to the next editing operation if the LaTeX command you want is shown in the buffer.

; itself in math-environment is inserted by `;;`. Typing `TAB` in the midst of image completion shows all of the LaTeX commands that start with the same name as string you previously typed in. In this menu buffer, press `RET` after moving the cursor (by `n`, `p`, `b`, `f`) to insert the LaTeX command.

To know all of the completion table, type `TAB` just after `;`. And here is the sample menu by `TAB` after `;<`.

```
KEY             LaTeX sequence          sign
<               \leq                    <
                                        ~

<<              \ll                     <<
<-              \leftarrow              <-
<=              \Leftarrow              <=
```

You can define your favorite key-vs-sequence completion table in the Emacs-Lisp variable `YaTeX-math-sign-alist-private`. See also '`yatexmth.el`' for the information of the structure of this variable.

6.10 Greek letters completion

Math-mode of YaTeX provides another image completion, Greek letters completion in the same method. After prefix `:`, typing `a` makes `\alpha`, `b` makes `\beta` and `g` makes `\gamma` and so on. First, type `:TAB` to know all the correspondence of alphabets vs. Greek letters.

If you will find `;` or `:` doesn't work in correct position of math environment, it may be a bug of YaTeX. Please send me a bug report with the configuration of your text, and avoid it temporarily by typing `;` or `:` after universal-argument(`C-u`) which forces `;` and `:` to work as math-prefix.

6.11 Inserting parentheses

Typing opening parenthesis, one of `(`, `{ and [`, automatically inserts the closing one. If a opening bracket is typed after `\`, `\]` is automatically inserted with computed indentation. If you stop automatic insertion, type `C-q` before opening parenthesis.

7 Local dictionaries

Tables for completion consist of three dictionaries; ‘standard dictionary’ built in ‘`yatex.el`’, ‘user dictionary’ for your common private commands, and ‘local dictionary’ that is effective in a certain directory.

When you input the command unknown to YaTeX at a completion in the minibuffer, YaTeX asks you with the following prompt;

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

In this menu, typing *`u`* updates your ‘user dictionary’, *`l`* updates your local dictionary, *`n`* updates only on-memory dictionary which go through only current Emacs session, and *`d`* updates no dictionary and throws the new word away.

If you find this switching feature meaningless and bothersome, put the next expression into your ‘`~/.emacs`’

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

8 Commenting out

You may want to comment out some region.

[prefix] `>`
: . . . comment out region by %

[prefix] `<`
: . . . uncomment region

cause an operation to the region between point and mark.

[prefix] `.`
: . . . comment out current paragraph

[prefix] `,`
: . . . uncomment current paragraph

comments or uncomments the paragraph where the cursor belongs. This ‘paragraph’ means the region marked by the function mark-paragraph, bound to *ESC h* by default. It is NOT predictable what will happen when you continuously comment out some paragraph many times.

You can also comment out an environment between `\begin` and `\end`, or a `\begin-\\end` pair themselves, by making the following key strokes on the line where `\begin{}` or `\end{}` exists.

[prefix] `>`
: . . . comment out from \begin to `\end`

[prefix] `<`
: . . . uncomment from \begin to `\end`

comment whole the contents of environment. Moreover,

[prefix] `.`
: . . . comment out \begin and `\end`

[prefix] `,`
: . . . uncomment \begin and `\end`

(un)comments out only environment declaration: `\begin{}` and `\end{}`. NOTE that even if you intend to comment out some region, invoking *[prefix]* `>` on the `\begin`,`\end` line decides to work in ‘commenting out from `\begin` to `\end`’ mode.

9 Cursor jump

9.1 Jump to corresponding object

Typing

[prefix] g

. . . go to corresponding object

in a certain place move the cursor to the place corresponding to the LaTEX command of last place. YaTeX recognize the followings as pairs that have relation each other.

- `\begin{} <-> \end{}`
- `%#BEGIN <-> %#END`
- On the image-including line `->` corresponding viewer or drawing tool
- `\label{} <-> \ref{}`
- `\include(\input) ->` included file
- `\bibitem{} <-> \cite{}`

On a `\begin`,`\end` line, typing *[prefix] g* moves the cursor to the corresponding `\end`,`\begin` line, if its partner really exists. The behavior on the line `%#BEGIN` and `%#END` are the same. Note that if the correspondent of `label/ref` or `cite/bibitem` exists in another file, that file have to be opened to make a round trip between references by *[prefix] g*.

If you type `[prefix] g` on the line of `\include{chap1}`, typically in the main text, YaTeX switches buffer to '`chap1.tex`'.

[prefix] 4 g

. . . go to corresponding object in other window

do the same job as *[prefix] g* except it's done in other window. Note that this function doesn't work on `begin/end`, `%#BEGIN/%#END` pairs because it is meaningless.

9.2 Invoking image processor

'image-including line' described above means such lines as `\epsfile{file=foo.ps}`. If you type *[prefix] g* on that line, YaTeX automatically searches source of 'foo.ps' and invokes image viewer or drawing tool correspoinding to it. For example; if you draw an image foo.obj with Tgif and enclose its product named foo.eps by `\epsfile` command. Typing *[prefix] g* on `\epsfile` line make YaTeX invoke `tgif foo.obj`. How a processor is choosen is as follows.

1. If there is an expression matching with one of the pattern defined in `YaTeX-processed-file-regexp-alist`, extract file name from regexp group surrounded by \\(\\). (Which group corresponds is written in the cdr part of each list.) If no matches were found, do nothing.
2. If there is a pattern as '%PROCESSOR' which is defined in the variable `YaTeX-file-processor-alist`, call that processor giving the file name with corresponding extension.

3. If not, check the existence of each file which is supplied the extension in the cdr part of each list of `YaTeX-file-processor-alist`. If any, call the corresponding image viewer or drawing tool.

9.3 Jump to main file

Typing

[prefix] ^
: ... visit main file

*[prefix] 4^*
: ... visit main file in other buffer

in a sub text switch the buffer to the main text specified by `%#!` notation.

9.4 Jumping around the environment

And these are the functions which work on the current LaTeX environment:

M-C-a	... beginning of environment
M-C-e	... `end` of environment
M-C-@	... mark environment

9.5 Jumping to last completion position

YaTeX always memorize the position of completion into register `3`. So every time you make a trip to any other part of text other than you are writing, you can return to the editing paragraph by calling register-to-point with argument YaTeX-current-position-register, which is achieved by typing *C-x j 3*(by default).

10 Changing and Deleting

These functions are for change or deletion of LaTEX commands already entered.

[prefix] c
: . . . change LaTEX command

[prefix] k
: . . . kill LaTEX command

10.1 Changing LaTEX commands

[prefix] c can change the various (La)TEX commands. This can change the followings.

- Environment names
- Section-type commands
- Argument of section-type commands
- Optional parameters (enclosed by []) of section-type commands
- Font/size designators
- Math-mode's maketitle-type commands that can be inputted with image completion

Typing *[prefix] c* on one of above objects you want to change brings a suitable reading function sometimes with completion. Note: If you want to change the argument of section-type command that contains other LaTEX commands, type *[prefix] c* either of surrounding braces of the argument in order to make YaTeX ignore the internal LaTEX sequences as an object of changing. Anyway, it is very difficult to know which argument position the cursor belongs because the LaTEX commands can be nested and braces can freely emerge. So keep it mind to put the cursor on a brace when you are thinking of changing a complicated argument.

10.2 Killing LaTEX commands

[prefix] k kills the LaTEX commands sometimes with their arguments. Following table illustrates the correspondence of the invoking position and what is killed.

```
[Invoking position]             [action]
\begin, \end line               kill \begin,\end pairs
%#BEGIN, %#END line             kill %#BEGIN,%#END pairs
on a Section-type command       kill section-type command
on a parenthesis                kill parentheses
```

Note that when killing `\begin, \end` or `%#BEGIN, %#END` pair, the lines `\begin, \end` or `%#BEGIN, %#END` exist will be killed entirely. So take care not to create any line that contains more than one `\begin` or so.

While all operations above are to kill 'containers' which surround some text, universal argument (*C-u*) for these commands kills not only 'containers' but also 'contents' of them. See below as a sample.

```
Original text:                  [prefix] k      C-u [prefix] k
Main \footnote{note} here.    Main note here. Main  here.
       ~(cursor)
```

11 Filling

11.1 Filling an item

To fill a term (descriptive sentences) of `\item`, type

M-q	. . . fill item

on that item.

YaTeX uses the value of the variable `YaTeX-item-regexp` as the regular expression to search item header in itemize environment. If you make a newcommand to itemize terms(e.g. `\underlineitem`), put

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

in your '`~/.emacs`'. If you are not familiar with regular expression for Emacs-Lisp, name a newcommand for 'itemize' beginning with `\item` such as `\itembf`, not `\bfitem`.

This function reformats the `\item` into 'hang-indented' style. For example:

```
  itemize, enumerate environment:
         >
         >\item[foo] `foo' is the typical word for describing an
         >           arbitrarily written....
  description environment:
         > \item[bar] When the word `for' is used as an arbitrarily
         >        word, `bar'  is bound to follow it.
```

Note that the indent depth of an `\item` word and its descriptive paragraph are the same in latter case. If you want to use different depth, invoke fill-paragraph at the beginning of non-whitespace character(see below).

11.2 Filling paragraph

Fill-paragraph is little bit adapted for LaTeX sources. It retains from filling in certain environments where formatting leads to a disaster such as verbatim, tabular, or so. And it protects `\verb` expressions from being folded (The variable `YaTeX-verb-regexp` controls this). Besides, putting cursor on the first occurrence of non-whitespace character on a line changes the fill-prefix temporarily to the depth of the line.

12 Updation of `\includeonly`

When you edit splitting source texts, the notation

```
\includeonly{CurrentEditingFileName}
```

in the main file reduces the time of typesetting. If you want to hack other file a little however, you have to rewrite it to

```
\includeonly{OtherFileNameYouWantToFix}
```

in the main file. YaTeX automatically detects that the current edited text is not in includeonly list and prompts you

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

in the minibuffer. Type `a` if you want to add the current file name to `\includeonly` list, *`r`* to replace `\includeonly` list with the current file, and type `%` to comment out the `\includeonly` line.

13 What column?

We are often get tired of finding the corresponding column in large tabulars. For example,

```
\begin{tabular}{|c|c|c|c|c|c|c|c|}\hline
 Name&Position&Post No.&Addr.&Phone No.&FAX No.&
        Home Addr.&Home Phone\\ \hline
 Thunder Bird & 6 & 223 & LA & xxx-yyy &
  zzz-www & Japan & 9876-54321 \\
   & 2 & \multicolumn{2}{c|}{Unknown}
        &&&(???)
 \\ \hline
 \end{tabular}
```

Suppose you have the cursor located at `(???)` mark, can you tell which column it is belonging at once? Maybe no. In such case, type

`[prefix] &`
: ... What column

in that position. YaTeX tells you the column header of the current field. Since YaTeX assumes the first line of tabular environment as a row of column headers, you can create a row of virtual column headers by putting them in the first line and commenting that line with `%`.

14 Intelligent newline

At the end of begin-type completion of tabular[*], array, itemize, enumerate or tabbing environment, or typing

ESC RET . . . Intelligent newline

in these environments inserts the contents corresponding to the current environment in the next line. (At the begin-type completion, this contents can be removed by 'undo'.) In `tabular` environment, for example, *ESC RET* inserts the certain number of `&` and trailing `\\`, and `\hline` if other `\hline` is found in backward. Here are the list of contents vs. environments.

- `tabular`, `tabular*`, `array`

  Corresponding number of `&` and `\\`. And `\hline` if needed.
- `tabbing`

  The same number of `\>` as `\=` in the first line.
- `itemize`, `enumerate`, `description`, `list`

  `\item` or `item[]`.

Note that since this function works seeing the contents of the first line, please call this after the second line if possible.

If you want to apply these trick to other environments, `foo` environment for example, define the function named `YaTeX-intelligent-newline-foo` to insert corresponding contents. That function will be called at the beginning of the next line after the newline is inserted to the current line. Since the function `YaTeX-indent-line` is designed to indent the current line properly, calling this function before your code to insert certain contents must be useful. See the definition of the function `YaTeX-intelligent-newline-itemize` as an example.

15 Usepackage checker

When you input begint-type, section-type, maketitle-type macros with completion, and it requires some LaTeX2e package, YaTeX examines the existence of correct `\usepackage`. If not, YaTeX inserts the `\usepackage{}` declaration corresponding to input macro.

To activate the package completion for your favarite package, set the variable `YaTeX-package-alist-private` correctly. Please refere the value of `YaTeX-package-alist-default` as an example.

16 Online help

YaTeX provides you the online help with popular LaTEX commands.

Here are the key strokes for the online help.

`[prefix] ?`
: . . . Online help

`[prefix] /`
: . . . Online apropos

16.1 Online help

'Online help' shows the documentation for the popular LaTEX commands(defaults to the commands on the cursor) in the next buffer. There are two help file, 'global help' and 'private help'. The former file contains the descriptions on the standard LaTEX command and is specified its name by variable `YaTeX-help-file`. Usually, the global help file should be located in public space (`$EMACSEXECPATH` by default) and should be world writable so that anyone can update it to enrich its contents. The latter file contains descriptions on non-standard or personal command definitions and is specified by `YaTeX-help-file-private`. This file should be put into private directory.

16.2 Online apropos

'Online apropos' is an equivalent of GNU Emacs's apropos. It shows all the documentations that contains the keyword entered by the user.

16.3 When no descriptions are found...

If there is no description on a command in help files, YaTeX requires you to write a description on that command. If you are willing to do, determine which help file to add and write the description on it referring your manual of (La)TeX. Please send me your additional descriptions if you describe the help on some standard commands. I might want to include it in the next distribution.

17 Browsing file hierarchy

When you are editing multi-file source, typing

`[prefix] d`
: . . . browse file hierarchy

asks you the parent-most file (which may be defaulted) and displays the documentation hierarchy in the next window. In this buffer, the following commands are available.

n	. . . move to the next line and show its contents
p	. . . move to the previous line and show its contents
N	. . . move to the next file in the same inclusion level
P	. . . move to the previous file in the same inclusion level
j	. . . move to the next line
k	. . . move to the previous line
u	. . . move to the parent file
.	. . . show the current files contents in the next window
SPC	. . . scroll up the current file window
DEL, b	. . . scroll down the current file window
<	. . . show the beginning of the current file
>	. . . show the end of the current file
>	. . . return to the previous postion after `<` or `>`
RET, g	. . . open the current file in the next window
mouse-2	. . . same as RET(available only with window system)
o	. . . other window
1	. . . delete other windows
-	. . . shrink hierarchy buffer window
+	. . . enlarge hierarchy buffer window
?	. . . describe mode
q	. . . quit

Note that operations on the file contents in the next window do not work correctly when you close the corresponding file.

18 Cooperation with other packages

YaTeX works better with other brilliant packages.

18.1 gmhist

When you are loading '`gmhist.el`' and '`gmhist-mh.el`', you can use independent command history list at the prompt of preview command (*`[prefix] tp`*) and print command (*`[prefix] tl`*). On each prompt, you can enter the previous command line string repeatedly by typing *`M-p`*.

18.2 min-out

'`min-out`', the outline minor mode, can be used in yatex-mode buffers. If you want to use it with YaTeX, please refer the file '`yatexm-o.el`' as an example.

19 Customizations

You can customize YaTeX by setting Emacs-Lisp variables and by making add-in functions.

19.1 Lisp variables

You can change the key assignments or make completion more comfortable by setting the values of various variables which control the movement of yatex-mode.

For example, if you want to change the prefix key stroke from *`C-c`* to any other sequence, set YaTeX-prefix to whatever you want to use. If you don't want to use the key sequence *`C-c letter`* which is assumed to be the user reserved sequence in Emacs world, set `YaTeX-inhibit-prefix-letter` to `t`, and all of the default key bind of *`C-c letter`* will turn to the corresponding *`C-c C-letter`* (but the region based completions that is invoked with *`C-c Capital-letter`* remain valid, if you want to disable those bindings, set that variable to 1 instead of `t`).

19.1.1 All customizable variables

Here are the customizable variables of yatex-mode. Each value setq-ed in '`~/.emacs`' is preferred and that of defined in '`yatex.el`' is neglected. Parenthesized contents stands for the default value. When you are to change some of these variables, see more detailed documentation of the variable by *`M-x describe-variable`*.

`YaTeX-japan` [Variable]
: Set this nil to produce all messages in English (`Depends on Japanese feature of Emacs`)

`YaTeX-kanji-code` [Variable]
: Default buffer-file-coding-system for YaTeX modes' buffer. Set this 0 to no language conversion. Nil to preserve original coding-system. 1=Shift JIS, 2=JIS, 3=EUC, 4=UTF-8 (`1 or 2`)

`YaTeX-prefix` [Variable]
: Prefix key stroke (*`C-c`*)

`YaTeX-inhibit-prefix-letter` [Variable]
: Change key stroke from *`C-c letter`* to *`C-c C-letter`* (`nil`)

`YaTeX-fill-prefix` [Variable]
: Fill-prefix used in yatex-mode (`nil`)

`YaTeX-user-completion-table` [Variable]
: Name of user dictionary where learned completion table will be stored. (`"~/.yatexrc"`)

`tex-command` [Variable]
: LaTeX typesetter command (`"latex"`)

`dvi2-command` [Variable]
: Preview command (`"xdvi -geo +0+0 -s 4"`)

`dviprint-command-format` [Variable]
: Command format to print dvi file (`"dvi2ps %f %t %s | lpr"`)

`dviprint-from-format` [Variable]
: Start page format of above %f. %b will turn to start page (`"-f %b"`)

`dviprint-to-format` [Variable]
: End page format of above %t. %e will turn to `end` page (`"-t %e"`)

`makeindex-command` [Variable]
: Default makeindex command (`"makeindex"` (`"makeind"` on MS-DOS))

`YaTeX-dvipdf-command` [Variable]
: Default command name to convert .dvi to PDF (`"dvipdfmx"`)

`YaTeX-on-the-fly-preview-interval` [Variable]
: Interval time in seconds of idle to trigger on-the-fly preview of environment by *`[prefix] t e`*(0.9). `Nil` disables on-the-fly preview.

`YaTeX-on-the-fly-math-preview-engine` [Variable]
: Function symbol to use on-the-fly preview of MATH environment started by *`[prefix] t e`* (`'YaTeX-typeset-environment-by-lmp` which calls latex-math-preview-expression function if latex-math-preview is available, otherwise `'YaTeX-typeset-environment-by-builtin` which alls built-in function).

  `Nil` disables on-the-fly preview.

`YaTeX-cmd-gimp` [Variable]
: Command name of GIMP (code"gimp")

`YaTeX-cmd-tgif` [Variable]
: Command name of tgif (code"tgif")

`YaTeX-cmd-inkscape` [Variable]
: Command name of Inkscape (code"inkscape")

`YaTeX-cmd-dia` [Variable]
: Command name of Dia (code"dia")

`YaTeX-cmd-ooo` [Variable]
: Command name of OpenOffice.org/LibreOffice (code"soffice")

`YaTeX-cmd-gs` [Variable]
: Command name of Ghostscript (code"gs")

`YaTeX-cmd-dvips` [Variable]
: Command name of dvips (code"dvips")

`YaTeX-cmd-displayline` [Variable]
: Command name of displayline (code"/Applications/Skim.app/Contents/SharedSupport/displayline"

`YaTeX-cmd-edit-ps` [Variable]
: Command name for editing PostScript files(Value of code"YaTeX-cmd-gimp")

`YaTeX-cmd-edit-pdf` [Variable]
: Command name for editing PDF files(Value of code"YaTeX-cmd-ooo")

`YaTeX-cmd-edit-ai` [Variable]
: Command name for editing '.ai' files(Value of code"YaTeX-cmd-inkscape")

`YaTeX-cmd-edit-svg` [Variable]
: Command name for editing SVG files(Value of code"YaTeX-cmd-inkscape")

`YaTeX-cmd-edit-images` [Variable]
: Command name for editing image files(Value of code"YaTeX-cmd-gimp")

`YaTeX-need-nonstop` [Variable]
: Put `\nonstopmode{}` or not (`nil`)

`latex-warning-regexp` [Variable]
: Regular expression of warning message latex command puts out (`"line.* [0-9]*"`)

`latex-error-regexp` [Variable]
: Regular expression of error message (`"l\\.[1-9][0-9]*"`)

`latex-dos-emergency-message` [Variable]
: Message latex command running on DOS puts at abort (`"Emergency stop"`)

`YaTeX-item-regexp` [Variable]
: Regular expression of item command (`"\\\\item"`)

`YaTeX-verb-regexp` [Variable]
: Regexp of verb family. Omit `\\\\`. (`"verb\\*?\\|path"`)

`YaTeX-nervous` [Variable]
: T for using local dictionary (`t`)

`YaTeX-sectioning-regexp` [Variable]
: Regexp of LaTeX sectioning command (`"\\(part\\|chapter\\*?\\|\\(sub\\)*\\(section\\|par`

`YaTeX-fill-inhibit-environments` [Variable]
: Inhibit fill in these environments (`'("tabular" "tabular*" "array" "picture" "eqnarray" "eqnarray*" "equation" "math" "displaymath" "verbatim" "verbatim*")`)

`YaTeX-uncomment-once` [Variable]
: T for deleting all preceding `%` (`nil`)

`YaTeX-close-paren-always` [Variable]
: T for always close all parenthesis automatically, `nil` for only eol (`t`)

`YaTeX-auto-math-mode` [Variable]
: Switch math-mode automatically (`t`)

`YaTeX-math-key-list-private` [Variable]
: User defined alist, math-mode-prefix vs completion alist used in image completion (`nil`). See '`yatexmth.el`' for the information about how to define a completion alist.

`YaTeX-default-pop-window-height` [Variable]
: Initial height of typesetting buffer when one-window. Number for the lines of the buffer, numerical string for the percentage of the screen-height. `nil` for half height (10)

`YaTeX-help-file` [Variable]
: Global online help file name (‘`$doc-directory/../../site-lisp/YATEXHLP.eng`’)

`YaTeX-help-file-private` [Variable]
: Private online help file name (‘`"~/YATEXHLP.eng"`’)

`YaTeX-no-begend-shortcut` [Variable]
: Disable [prefix] b ?? shortcut (`nil)`

`YaTeX-hilit-pattern-adjustment-private` [Variable]
: List of the list that contain the regular expression and the symbol of logical meaning of the string that matches the pattern. See also the value from `(assq 'yatex-mode hilit-patterns-alist)` and the value of `YaTeX-hilit-pattern-adjustment-default` (and even the document of hilit19.el).

`YaTeX-sectioning-level` [Variable]
: Alist of LaTeX's sectioning command vs its height.

`YaTeX-hierarchy-ignore-heading-regexp` [Variable]
: `YaTeX-display-hierarchy` searches for sectioning command first, and comment line secondary as a file headings. In latter case, ignore lines that match with regular expression of this variable. Default value of this variable is RCS header expressions and mode specifying line ‘-*- xxxx -*’.

`YaTeX-skip-default-reader` [Variable]
: Non-nil for this variable skips the default argument reader of section-type command when add-in function for it is not defined (`nil`)

`YaTeX-create-file-prefix-g` [Variable]
: When typing *`prefix` g* on the `\include` line, open the target file even if the file doesn't exist (`nil`)

`YaTeX-simple-messages` [Variable]
: Simplyfy messages of various completions (`nil`)

`YaTeX-hilit-sectioning-face` [Variable]
: When hilit19 and yatex19 is active, YaTeX colors the sectioning commands. This variable specifies the foreground and background color of `\part` macro. The default value is `'(yellow/dodgerblue yellow/slateblue)`. The first element of this list is for the screen when `hilit-background-mode` is `'light`, and the second element is for `'dark`. You should specify both color as ‘forecolor/backcolor’.

`YaTeX-hilit-sectioning-attenuation-rate` [Variable]
: When color mode, this variable specifies how much attenuate the color density of `\subparagraph` compared with that of `\chapter` (`'(15 40)`) See also `YaTeX-hilit-sectioning-face`.

`YaTeX-use-AMS-LaTeX` [Variable]
: If you use AMS-LaTeX, set to `t` (`nil`)

`YaTeX-use-LaTeX2e` [Variable]
: If you use LaTeX2e, set to `t` (`t`)

`YaTeX-template-file` [Variable]
: File name which is automatically inserted at creation (`~/work/template.tex`)

`YaTeX-search-file-from-top-directory` [Variable]
: Non-nil means to search input-files from the directory where main file exists (`t`)

`YaTeX-use-font-lock` [Variable]
: Use font-lock to fontify buffer or not (`(featurep 'font-lock)`

`YaTeX-use-hilit19` [Variable]
: Use hilit19 to highlight buffer or not (`(featurep 'hilit19)`

`YaTeX-use-italic-bold` [Variable]
: YaTeX tries to search italic, bold fontsets or not (`t` if Emacs-20 or later). This variable is effective only when font-lock is used. (`(featurep 'hilit19)`

`YaTeX-singlecmd-suffix` [Variable]
: Suffix which is always inserted after maketitle-type macros. `"{}"` is recommended.

`YaTeX-package-alist-private` [Variable]
: Alist of LaTeX2e-package name vs. lists of macros in it. Set this alist properly and YaTeX automatically check the declaratiion of 'usepackage' for corresponding macro, when you input that macro with completion. If required 'usepackage' is not found, YaTeX also automatically inserts '\usepackage'. Alist is as follows;

```
'((PackageName1
     (completionType ListOfMacro)
     (completionType ListOfMacro))
  (PackageName2
     (completionType ListOfMacro)
     (completionType ListOfMacro...))....)
```

completionType is one of `env, section, maketitle`. Consult the value of `YaTeX-package-alist-default` as an example.

`YaTeX-tabular-indentation` [Variable]
: At indentation by *C-i* in tabular or array environment, YaTeX put the additional spaces to the normail indentation depth. The number of additional spaces is the product of YaTeX-tabular-indentation and the number of column position in tabular.

`YaTeX-noindent-env-regexp` [Variable]
: Regexp of environment names that should begin with no indentation. All verbatime-like environment name should match with.

`YaTeX-electric-indent-mode` [Variable]
: Emacs 24.4 introduces automatic indentation of current and new lines. This might be annoying for some people. Pass this value to the function 'electric-indent-local-mode. If you prefer to stop electric-indent-mode in yatex-mode, set '-1' to this variable.

`YaTeX-ref-default-label-string` [Variable]

Default \\ref time string format. This format is like strftime(3) but allowed conversion char are as follows; %y -> Last 2 digit of year, %b -> Month name, %m -> Monthe number(1-12), %d -> Day, %H -> Hour, %M -> Minute, %S -> Second, %qx -> alphabetical-decimal conversion of yymmdd. %qX -> alphabetical-decimal conversion of HHMMSS. Beware defualt label-string should be always unique. So this format string should have both time part (%H+%M+%S or %qX) and date part (%y+(%b|%m)+%d or %qx).

`YaTeX-ref-generate-label-function` [Variable]

Function to generate default label string for unnamed \\label{}s. The function pointed to this value should take two arguments. First argument is LaTeX macro's name, second is macro's argument. Here is an example for using this value.

```
(setq YaTeX-ref-generate-label-function 'my-yatex-generate-label)
(defun my-yatex-generate-label (command value)
  (and (string= command "caption")
       (re-search-backward "\\\\begin{\\(figure\\|table\\)}" nil t)
       (setq command (match-string 1)))
  (let ((alist '(("chapter" . "chap")
                 ("section" . "sec")
                 ("subsection" . "subsec")
                 ("figure" . "fig")
                 ("table" . "tbl"))))
    (if (setq command (cdr (assoc command alist)))
        (concat command ":" value)
      (YaTeX::ref-generate-label nil nil))))
```

19.1.2 Sample definitions

For instance, to change the prefix key stroke to *ESC*, and name of the user dictionary '`~/src/emacs/yatexrc`', and set `fill-prefix` to single TAB character, add the following `setq` to '`~/.emacs`'.

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

19.1.3 Hook variables

More customizations will be done by the hook-function defined in hook-variable `yatex-mode-hook`. This is useful to define a shortcut key sequence to enter some environments other than `document` and `enumerate` etc. The following statement defines `[prefix] ba` to enter `\begin{abstract}` ... `=end{abstract}` immediately.

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

You should use functions `YaTeX-define-key`, or `YaTeX-define-begend-key` to define all the key sequences of yatex-mode.

19.1.4 Hook file

You can stuff all of YaTeX related expressions into a file named '`yatexhks.el`' if you have a lot of codes. YaTeX automatically load this file at the initialization of itself. Using '`yatexhks.el`' makes `yatex-mode-load-hook` unnecessary.

19.2 Add-in functions

You can easily define a function to input detailed arguments with completion according to LaTeX environments or commands.

19.2.1 What is add-in functions?

When you input `tabular` environment, don't you think "I want YaTeX to complete its argument toward my favorite one such as `{|c|c|c|}`..."? Yes, you can define the function to complete arguments for any environment and any LaTeX commands.

19.2.2 Procedure

Here is the procedure to define add-in functions.

1. Define the function
2. Put the function into '`yatexhks.el`'

19.2.3 How the add-in function works

There are three types of add-in.

1. Option add-in
2. argument add-in
3. enclosing add-in

Option add-in returns the LaTeX's optional parameters such as optional strings after `\begin{ENV}`, optional strings between a section-type command and its first argument, and optional strings just after type maketitle-type command. The following illustrates the name of add-in functions, where underlined strings are generated by add-in functions.

```
\begin{table}[ht] (Function name: YaTeX:table)
             ~~~~

\put(100,200){} (Function name: YaTeX:put)
    ~~~~~~~~~~

\sum_{i=0}^{n} (Function name: YaTeX:sum)
    ~~~~~~~~~~
```

Obviously, the function name is decided by concatenating the prefix 'YaTeX:' and LaTeX command's name.

Another add-in type is *argument add-in*, which completes arguments for section-type commands.

```
\newcommand{\foo}{bar} (Function name: YaTeX::newcommand)
            ~~~~  ~~~
```

When the section-type command is inputted, the function named by concatenating 'YaTeX::' and section-type command, is called automatically with an integer argument which indicates which argument of section-type command is being read. Thus the add-in should determine the job referring the value of its argument.

enclosing add-in is for modifying and/or checking the region that will be enclosed by section-type commands via *[prefix] S*. An enclosing add-in function will be called with two arguments, beginning of the enclosed region and end of the region. Suppose you want to enclose the existing text `(a+b)/c` by `\frac{}`.

```
a/c
| |
A B
```

You do set-mark-command at point A and then move to point B. Typing *[prefix] S* and input `frac` enclose the region like this;

```
\frac{a/c}
```

Normally, the expression `a/c` is translated to `\frac{a}{c}`. An enclosing add-in is useful for modifying `/` to `}{`.

19.2.3.1 Defining 'option add-in'

If you want `{|c|c|c|}` for all `tabular` environment,

```
(defun YaTeX:tabular ()
  "{|c|c|c|}")
```

is enough. If you want more complicated format, define as below.

```
(defun YaTeX:tabular ()
  "{@{\\vrule width 1pt\\ }|||@{\\ \\vrule width 1pt}}")
```

Note that the character \ must be described as \\ in Emacs-Lisp. The next example reads the tabular format from keyboard.

```
(defun YaTeX:tabular ()
  (concat "{" (read-string "Rule: ") "}"))
```

19.2.3.2 Defining 'argument add-in'

This section describes how to define the add-in function for `\newcommand`.

The first argument of `\newcommand` begins always with `\`. The second argument is usually so complex that we can not edit them in the minibuffer. Here is the created function considering this.

```
(defun YaTeX::newcommand (n) ;n is argument position
  (cond
   ((= n 1) ;1st argument is macro name
    (read-string "Command: " "\\")) ;initial input `\'
   ((= n 2) "") ;do nothing when reading arg#2
   (t nil)))
```

Note that when the 'argument add-in' function return 'nil', normal argument reader will be called.

19.2.3.3 Defining 'enclosing add-in'

This section describes how to define the add-in function for text enclosed by `\frac{}`.

When enclosing the text `5/3` by `\frac{}`, you might want to replace `/` with `}{`. Enclosing function `YaTeX::frac-region` is called with two arguments, beginning of enclosed text and end of enclosed text. The function is expected to replace `/` with `}{`. Here is an example expression.

```
(defun YaTeX::frac-region (beg end)
  (catch 'done
    (while (search-forward "/" end t)
      (goto-char (match-beginning 0))
      (if (y-or-n-p "Replace this slash(/) with `}{'")
  (throw 'done (replace-match "}{")))
      (goto-char (match-end 0)))))
```

19.2.4 How the function is called

YaTeX calls the add-in functions for specified begin-type, section-type, and maketitle-type command, if any. 'Option add-in' functions for begin-type are called when `\begin{ENV}` has been inserted, functions for section-type are called just before input of the first argument, and functions for maketitle-type is called after maketitle-type command has been inserted. 'Argument add-in' functions are called at each entry of arguments for section-type commands.

19.2.5 Useful functions for creating add-in

Many add-in functions for typical LaTeX commands are defined in '`yatexadd.el`'. Those are also useful as references. Here are the short descriptions on useful functions, where [F] means function, [A] means arguments, [D] means description.

[F] YaTeX:read-position

[A] Character list which can show up in the brackets

[D] Return the location specifier such as '[htb]'. When nothing is entered, omit [] itself. If the possible characters are "htbp", call this function as `(YaTeX:read-position "htbp")`

[F] YaTeX:read-coordinates

[A] Base prompt, X-axis prompt, Y-axis prompt (each optional)

[D] Read the coordinates with the prompt "BasePrompt X-axisPrompt:" for X-axis, "BasePrompt Y-axisPrompt:" for Y-axis, and return it in the form of "(X,Y)". The default prompts are `Dimension`, `X`, `Y` respectively.

[F] YaTeX:check-completion-type

[A] One of the symbols: 'begin, 'section, or 'maketitle

[D] Check the current completion type is specified one and cause error if not. The variable `YaTeX-current-completion-type` holds the symbol according to the current completion type.

19.2.6 Contribution

If you make your own pretty function and you let it be in public, please send me the function. I'm going to include it in the next release.

19.3 Add-in generator

First, don't forget to read the section of add-in functions Section 19.2 [Add-in functions], page 34. If you easily understand how to define them, there's no need to read this section. But being not familiar with Emacs-Lisp, when you don't have clear idea what to do, this section describes how to get YaTeX make add-in function.

There are two methods of generation. One is for fully interactive generator for beginners and another requires little knowledge of Emacs-Lisp.

19.3.1 Generator for beginners

The former generator is called by

M-x YaTeX-generate

strokes. All you have to do is follow the guidances. Defying them may cases the disaster (I wonder what is it???). So when you make some mistake, it is recommendable to type *C-g* and start afresh.

19.3.2 Simple generator

The latter generator is invoked by the next sequence.

M-x YaTeX-generate-simple

This generator can make both “option add-in” and “argument add-in” (*refer the section add-in functions* Section 19.2.3 [How the add-in function works], page 34), whereas `YaTeX-generate` cannot make “argument addin”.

For example, assume you have the LaTeX command as follows.

```
\epsinput[t](250,50){hoge.eps}{plain}{Picture of foo}
         (A)  (B)     (1)      (2)      (3)
(A)Optional parameter to specify the position
   One of t(top), b(bottom), l(left), r(right)
(B)Maximum size of frame
(1)1st argument is filename of EPS file
(2)2nd argument indicates
plain do nothing
frame make frame around image
dframe make double-frame around image
   for included EPS file.
(3)Caption for the picture
```

Now get start with generation. Typing *M-x YaTeX-generate-simple* brings the prompt:

(O)ption? (A)rgument?

19.3.2.1 Generating “option add-in”

Since (A), (B) above are optional argument, all we have to do to complete them is define the option add-in for them. Let's generate the function to complete (A).

M-x YaTeX-generate-simple RET
epsinput RET
o

Typing as above leads the next prompt.

Read type(1): (S)tring (C)omplete (F)ile ([)option (P)osition co(O)rd. (q)uit

This asks that “Which type is the completion style of 1st argument?”. Here are the possible completion style.

`String`	read plain string
`Complete`	read with completion
`File`	read file name
`Option`	read optional string (if string omitted, omit [] too)
`Position`	read positional option (like [htbp])
`Coord.`	read coordinates
`Quit`	quit from generating

Since (A) is the optional argument to specify the location of included EPS file, the completion style is `Position`, and the possible characters are t, b, l, and r. To tell these information to generator, operate as follows.

```
Read type(1).... p
Acceptable characters: tblr RET
```

(B) is coordinate. So its completion style is coOrd. We want a prompt meaning “Maximum size” when completion.

```
Read type(2).... o
Prompt for coordinates: Max size RET
```

That’s all for optional argument. Select quit.

```
Read type(3).... q
```

Then the generated option add-in function for \epsinput will be shown in the next window.

19.3.2.2 Generating “argument add-in”

Next, create the argument add-in. The arguments for \epsinput are EPS file name, framing style, and caption string in sequence.

```
M-x YaTeX-generate-simple RET
epsinput RET
a
```

Above key strokes bring the prompt that asks the number of argument. Answer it with 3.

```
How many arguments?: 3 RET
```

Then the generator asks the completion style and prompt for completion. Answer them. `f` for FileName and prompt string.

```
Read type(1).... f
Prompt for argument#1 EPS file name RET
```

The second argument is one of selected symbol. So the completion type is `Completion`.

```
Read type(2).... c
Prompt for argument#2 Include style RET
```

Then all the candidates ready to be read. Type single RET after entering all.

```
Item[1](RET to exit): plain RET
Item[2](RET to exit): frame RET
Item[3](RET to exit): dframe RET
Item[4](RET to exit): RET
```

The following prompt asks whether the entered string must belong to candidates or not. In this case, since the argument must be one of `plain`, `frame`, and `dframe`, type `y`.

```
Require match? (y or n) y
```

The last argument is the caption string for which any completion is needed.

```
Read type(3).... s
Prompt for argument#3 Caption RET
default: Figure of RET
```

Finally we'll get the argument add-in in the next window.

19.3.3 Contribution

If you get your own pretty function and you let it be in public, please steel yourself in the happy atmosphere and do not send me the function. I do know it is not fine because it is generated by yatexgen:-p.

20 Etcetera

The standard completion tables provided in '`yatex.el`' contain a few LaTEX commands I frequently use. This is to lessen the key strokes to complete entire word, because too many candidates rarely used often cause too many hits. Therefore always try to use completion in order to enrich your dictionary, and you will also find 'Wild Bird' growing suitable for your LaTEX style.

The package name 'Wild Bird' is the English translation of Japanese title 'Yachou', which is a trick on words of Japanese.

21 Copying

This program is distributed as a free software. You can use/copy/modify/redistribute this software freely but with NO warranty to anything as a result of using this software. Adopting code from this program is also free. But I would not do contract act.

Any reports and suggestions are welcome as long as I feel interests in this software. My possible e-mail address is 'yuuji@yatex.org'. (as of Jan.2004) And there is mailing list for YaTeX. Although the common language is Japanese, questions in English will be welcome. To join the ML, send the mail whose subject is 'append' to the address 'yatex@yatex.org. If you have some question, please ask to 'yatex-admin@yatex.org'.

The specification of this software will be surely modified (depending on my feelings) without notice :-p.

HIROSE Yuuji