The metainfo package*

Jonathan Sauer
jonathan.sauer@gmx.de

2004/11/25

Abstract

This file describes the metainfo package that typesets only special comments of a TEX file.

Contents

1 Introduction **2**

2 Macros **2**

3 Formatting **2**
3.1 Metainfo after a heading 2
3.2 Local metainfo 2

4 Package options **3**

5 Driver files **3**

6 Examples **4**
6.1 Example driver file 4
6.2 Example document 4
6.3 Example local metainfo formatting 5

7 Notes/Limitations **5**

8 Implementation **6**
8.1 Main environments and macros 6
8.2 Package customization 6
8.3 Package options 7
8.3.1 Headings 7
8.3.2 Index shorthand 7
8.3.3 Other options 9

*This document corresponds to metainfo.sty v0.9.1, dated 2004/11/25.

8.4 Internal environments and macros 9
8.4.1 General definitions and flags 9
8.4.2 Modes 10
8.4.3 Gobbling text 14
8.4.4 Catcode changes 15

1 Introduction

Suppose you write a text and include some annotations for yourself that are not printed, i.e. an explanation of a pun or a short summary of the current chapter in order to be able to later create a summary of the whole text. Then it would be nice to be able to extract these annotations and typeset them separately. This package provides the macro `\typesetmetainfo` to typeset only these annotations (herein called 'metainfo', as they are information about the information contained in the text).

2 Macros

`\typesetmetainfo` Usage: `\typesetmetainfo {`⟨*file*⟩`}`.

Typesets the metainfo of the TEX file ⟨*file*⟩. Anything not a metainfo—text, preamble—is skipped.

3 Formatting

3.1 Metainfo after a heading

Any comments following a line with a heading are considered a metainfo and are typeset as if they were normal text. The first line that does not begin with a percent sign finishes the metainfo.

A heading is started by a macro defined in `\\mi@@MIbeginnings`. Normally this macro contains `\chapter`, `\section`, `\subsection`, and `\subsubsection`, but you can of course redefine it.

The heading itself is typeset as well, before the metainfo, resulting all metainfo being typeset using the outline of the normal document. Note that when writing a heading, some restriction apply:

1. The heading macro must be at the beginning of the line.
2. Any parameters of the macro must be on the same line. (parameters split over several lines might work, but they are not guaranteed to)

3.2 Local metainfo

There is another way to include metainfo in a document, that is as a *local metainfo*. A local metainfo is a metainfo not following a heading; instead they can appear

anywhere in the text. They start with a double percent sign at the beginning of the line (`%%`); the line itself is then typeset as a metainfo as well as all the following lines beginning with a `%` (just as a metainfo following a heading). The first line that does not begin with a percent sign finishes the local metainfo.

You can format a local metainfo using three macros. (see section 6.3 on page 5 for an example how to customize them.)

1. `\mi@@firstlocalMItext`: The contents of this macro is inserted before the first local metainfo of a chapter.
2. `\mi@@lastlocalMItext`: The contents of this macro is inserted after the last local metainfo of a chapter.
3. `\mi@@everylocalMItext`: The contents of this macro is inserted before every local metainfo.

Note If you change these macros in your document preamble or your main document opposed to a package file, you must surround them with `\makeatletter` and `\makeatother`, as shown in the example below.

4 Package options

The following package options exist:

compactheadings Changes the section headings to be more compact in order to save some space.

indexshorthand Provides shorthands for indexing. `^{`⟨*text*⟩`}` indexes and typesets ⟨*text*⟩, `^^{`⟨*text*⟩`}` only indexes it.[1] For indexing, the standard index macro `\index` is used.

These macros only work in text mode; in math mode, `^` is a superscript as predefined in LaTeX.

listlocalmetainfo Changes the macros `\mi@@firstlocalMItext`, `\mi@@lastlocalMItext` and `\mi@@everylocalMItext` (described in the section above) to `itemize` the local metainfo.

5 Driver files

In order to typeset only the metainfo of a document, a special driver file is needed. This driver loads the packages necessary for typesetting the metainfo (at least the package metainfo) and inputs the document to be typeset using `\typesetmetainfo`.

[1] These shorthands have been inspired by Donald E. Knuth's own index macros used for the TeXbook.

Note No package of the document processed using `\typesetmetainfo` is loaded, as the `\usepackage` macros are skipped. The same is true for any definitions in the document preamble or elsewhere: They are skipped, so if the metainfo relies on these definitions, they have to be included in the driver as well.

6 Examples

6.1 Example driver file

The following driver file typesets the metainfo of the TEX file 'example.tex' using compact headings:

```
\documentclass{book}
\usepackage[compactheadings]{metainfo}

\begin{document}
\typesetmetainfo{example}
\end{document}
```

6.2 Example document

If we save the following file as 'example.tex' and process it using the driver file in the example above ...

```
\documentclass{minimal}
\usepackage{testpackage}

\begin{document}

\chapter{Chapter 1}
% Metainfo for chapter 'Chapter 1'

\section{Section 1}
% Metainfo for section 'Section 1'

This is some text.

\chapter{Chapter 2}
\section{Section 2}
% Metainfo for section 'Section 2'

This is some more text.

%% Local metainfo. This metainfo is a bit longer, but only
% a little bit.

\section{Section 3}
% Metainfo for section 'Section 3'
```

```
\end{document}
```

... this results in:[2]

Chapter 1
Metainfo for chapter 'Chapter 1'

Section 1
Metainfo for section 'Section 1'

Chapter 2
Section 2
Metainfo for section 'Section 2'

Annotations:
Local metainfo. This metainfo is a bit longer, but only a little bit.

Section 3
Metainfo for section 'Section 3'

6.3 Example local metainfo formatting

The following macros prefix local metainfo with 'Annotations:' in bold typeface and typesets the local metainfo in an `itemize` environment:[3]

```
\makeatletter
\def\mi@@firstlocalMItext{%
  \addvspace{\baselineskip}%
  \noindent\textbf{Annotations:}%
  \begin{itemize}%
}
\def\mi@@lastlocalMItext{%
  \end{itemize}%
}
\def\mi@@everylocalMItext{%
  \item\relax%
}
\makeatother
```

7 Notes/Limitations

- Any text or macros on the same line as a heading are processed as well, immediately after the heading. Thus you can type `\section{foo}\label{sec:foo}`

[2] Approximately, as the real formatting will differ slightly from the text typeset here.
[3] Similar to the package option `listlocalmetainfo`, described in section 4 on page 3 does.

and refer to this section inside a metainfo using the label `sec:foo`.

- The document included using `\typesetmetainfo` must be a valid LaTeX document insofar as that it must contain a `document` environment, because `\end{document}` serves as the ending delimiter for typesetting the metainfo.

  That also means that any metainfo following `\end{document}` is *not* typeset.

8 Implementation

8.1 Main environments and macros

`\typesetmetainfo` Usage: `\typesetmetainfo {`⟨*file*⟩`}`.

Typesets the metainfo of a document.

```
\newcommand{\typesetmetainfo}[1]{%
  \bgroup%
  \mi@activenewline%
  \mi@emptyactivepercent%
  \mi@otherbraces%
  \ifmi@indexmacros\mi@activehat\fi%
```

We provide support for the standard LaTeX `verbatim` environment:

```
  \let\mi@old@verbatim\@verbatim%
  \def\@verbatim{%
    \mi@old@verbatim%
    \mi@verbatimnewline%
  }%
```

We use the original TeX definition of `\input` (saved by LaTeX in `\@@input`), because we must continue with our special processing immediately after ⟨*file*⟩ has been opened. This is a job for the original TeX `\input`, as it simply switches the input stream to ⟨*file*⟩ and expands to nothing.

The `\relax` delimits the filename.

```
  \expandafter\mi@skiplines\@@input#1\relax%
  \egroup%
}
```

8.2 Package customization

`\mi@@MIbeginnings` Stores all the macros that can begin a metainfo.

```
\def\mi@@MIbeginnings{\chapter\section\subsection\subsubsection}
```

`\mi@@firstlocalMItext` Stores the text inserted before the first local metainfo.

```
\def\mi@@firstlocalMItext{\textbf{Annotations:}\par\noindent}
```

`\mi@@lastlocalMItext` Stores the text inserted after the last local metainfo.

```
\def\mi@@lastlocalMItext{}
```

\mi@@everylocalMItext Stores the text inserted before every local metainfo.

```
\def\mi@@everylocalMItext{}
```

8.3 Package options

8.3.1 Headings

We provide the possibility of changing the headings to a more compact formatting:

```
\DeclareOption{compactheadings}{%
  \@ifundefined{thechapter}{}{%
    \def\chapter{\@startsection{chapter}{0}{\z@}{-2\baselineskip}%
          {\baselineskip}{\normalfont\normalsize\bfseries}}%
  }%
  \def\section{\@startsection{section}{1}{\z@}{-2\baselineskip}%
        {\baselineskip}{\normalfont\normalsize\bfseries}}%
  \def\subsection{\@startsection{subsection}{2}{\z@}{-2\baselineskip}%
        {\baselineskip}{\normalfont\normalsize\bfseries}}%
  \def\subsubsection{\@startsection{subsubsection}{3}{\z@}%
        {-2\baselineskip}{\baselineskip}%
        {\normalfont\normalsize\bfseries}}%
}
```

8.3.2 Index shorthand

\ifmi@indexmacros `true`, if index shorthands specified using the package option `indexshorthand` are used, otherwise `false`:

```
\newif\ifmi@indexmacros
\mi@indexmacrosfalse
```

We provide the possibility of using the shorthand `^` for an index entry that is typeset, and `^^` for an index entry that is not typeset:

```
\DeclareOption{indexshorthand}{%
  \mi@indexmacrostrue%
```

\mi@hat We save the original meaning of `^`. We do not simply use `^` in the `\mi@hat...`-macros, because then we would assume that `^` is not already an active character before our catcode change (which we will make in `\typesetmetainfo`). Normally, `^` has catcode 7 (superscript), but if is was active (i.e. because of another package changing `^`), we would store an active `^` in the definition of the `\mi@hat...`-macros. When later used this `^` would point to our macro `\mi@hat`, resulting in an endless loop. So we save the original meaning of `^` instead using `\let`, which, if `^` is active, would save the macro `^` would have been `\let` to.

```
  \let\mi@@@hat^%
```

\mi@hat Main macro for indexing and first step in deciding how to index:

- Math mode: Expand to the original definition of `^`.

- Otherwise: Check the next token using `\mi@hat@`.

```
\def\mi@hat{%
  \ifmmode%
    \expandafter\mi@@@hat%
  \else%
    \expandafter\futurelet\expandafter\@tempa\expandafter\mi@hat@%
  \fi%
}
```

`\mi@hat@` Second step in deciding how to index. We know that we are not in math mode. The token following the hat has been prefetched and stored in `\@tempa`. (it has not been read yet)

- A second hat (`^`): Check the next token using `\mi@hat@twohats`.
- Opening brace: Typeset the index word and index it using it `\mi@indextypeset`.
- Otherwise: Display an error, as `^^` must be followed by a parameter in braces.

```
\def\mi@hat@{%
  \ifx\@tempa\mi@hat%
    \expandafter\mi@hat@twohats%
  \else\ifx\@tempa\bgroup%
    \expandafter\expandafter\expandafter\mi@indextypeset%
  \else%
    \mi@hat@errbrace%
  \fi\fi%
}
```

`\mi@hat@twohats` First step in deciding how to index a word not typeset. We know we have two hats in a row (`^^`), but the second hat is not read yet (only prefetched using `\futurelet`). Therefore we gobble it using `#1`.

```
\def\mi@hat@twohats#1{%
  \futurelet\@tempa\mi@hat@twohats@%
}
```

`\mi@hat@twohats` Second step in deciding how to index a word not typeset. We know we have two hats in a row, both read. The token following the two hats has been prefetched and stored in `\@tempa`.

Note that we could skip this step and simpy expand to `\index`, letting `\index` take care that a proper parameter follows.

```
\def\mi@hat@twohats@{%
  \ifx\@tempa\bgroup%
    \expandafter\index%
  \else%
    \mi@hat@errbrace%
  \fi%
}
```

`\mi@indextypeset` Usage: `\mi@indextypeset {`⟨*word*⟩`}`. Typesets ⟨*word*⟩ and indexes it using `\index`.

```
\def\mi@indextypeset#1{%
  #1\index{#1}%
}
```

`\mi@hat@errbrace` Error handling when a `^` or `^^` is not followed by a left brace.

```
\def\mi@hat@errbrace{%
  \PackageError{metainfo}{`\string^' or `\string^\string^' not %
              followed by a left brace}\@ehc%
}
```

End of `\DeclareOption{index}`:

```
}
```

8.3.3 Other options

```
\DeclareOption{listlocalmetainfo}{%
  \def\mi@@firstlocalMItext{%
    \begin{itemize}%
  }%
  \def\mi@@lastlocalMItext{%
    \end{itemize}%
  }%
  \def\mi@@everylocalMItext{%
    \item\relax%
  }%
}
```

```
\ProcessOptions\relax
```

8.4 Internal environments and macros

8.4.1 General definitions and flags

Strategy for dealing with catcode changes At the beginning of the processing, the catcodes are changed as follows:

`^^M`	13 (active), set in `\mi@activenewline`
`%`	13, set in `\mi@emptyactivepercent`
`{`	12 (other), set in `\mi@otherbraces`
`}`	12, set in `\mi@otherbraces`
`^`	13, optionally set in `\mi@activehat`

If any macro changes the catcodes of these characters, it must reset them to these values after processing. If any macro needs the braces `{` and `}` for parameter grouping, it should use the macro `\mi@normalbraces` to change the catcodes accordingly.

`\ifmi@firstlocalMI` `true` if this is the first local metainfo, otherwise `false`. Used to insert `\mi@@firstlocalMItext` and `\mi@@lastlocalMItext`.

```
\newif\ifmi@firstlocalMI
\mi@firstlocalMItrue
```

8.4.2 Modes

Three modes of processing exist:

Skipping text Text is skipped line after line. If two percent signs are found at the beginning of a new line, a local metainfo is begun and the mode changed to 'Typesetting a metainfo'.

Main macro: `\mi@skiplines`.

Beginning metainfo The appropriate heading is typeset and the remaining line of text skipped. The mode is then changed to 'Typesetting a metainfo'.

Main macro: `\mi@checkbeginMI`.

Typesetting a metainfo The text is typeset. If a line does not start with a percent sign, the metainfo is done. The mode is then changed to 'Skipping text'.

Main macro: `\mi@typesetMI`.

`\mi@skiplines` Main macro. Skips lines and checks the first token of a line (stored in `#1`) for two special cases:

1. A percent sign `%`: We check if it starts a local metainfo using `\mi@checkbeginlocalMI`.
2. A control sequence: We check if it starts a new metainfo using `\mi@checkbeginMI`.

```
\def\mi@skiplines#1{%
  \ifx#1\mi@percentempty%
```

A percent sign: We check if it starts a local metainfo.

```
    \expandafter\mi@checkbeginlocalMI%
  \else\ifcat\noexpand#1\relax%
```

A control sequence: We check if it starts a new metainfo.

```
    \expandafter\expandafter\expandafter\mi@checkbeginMI%
  \else%
```

Anything else: We skip it and the remaining line of text. Note that it could be a line ending (`^^M`), so it may be possible that we gobble only `#1`, which will be re-inserted into the stream below. That way we do not have to check if `#1` is a line ending.

```
    \expandafter\expandafter\expandafter\mi@gobbletolineend%
  \fi\fi%
```

We always insert the first token of a line back into the stream, even though it is not necessary for `\mi@checkbeginlocalMI`. But as normally most of the lines of a text will be skipped, we optimize for this case.

```
  #1%
}
```

The following macros require `^^M` to be active:

```
\bgroup
\catcode`\^^M=\active%
```

`\mi@typesetMI` Prepares the typesetting of a metainfo.

```
\gdef\mi@typesetMI{%
  \mi@normalbraces%
```

Every CR we check if the next line continues the metainfo using `\mi@typesetMI@checkend`.

```
  \let^^M\mi@typesetMI@checkend%
```

Now we start typesetting the text. We let TEX take completely control instead of reading one line after the other; we will regain control at the end of each line using our redefinition of `^^M`.

```
  \ignorespaces%
}
```

`\mi@typesetMI@checkend` Called at a carriage return. If the token after the CR is equals to `\mi@percentempty`, the next line starts with a `%`, thus continuing the metainfo. Otherwise, the metainfo is finished.

```
\long\gdef\mi@typesetMI@checkend#1{%
  \ifx#1\mi@percentempty%
```

The next line continues the comment. We check for an empty line:

```
    \expandafter\mi@typesetMI@checkend@%
  \else%
```

The next line finishes the comment:

```
    \mi@otherbraces%
    \let^^M\@empty%
    \expandafter\mi@skiplines\expandafter#1%
  \fi%
}
```

`\mi@typesetMI@checkend@` Checks if a `^^M` (CR) follows immediately after the `%`. Then we insert a `\par`. Note that any spaces between the `%` and the CR are automatically skipped by using a non-delimited parameter.

```
\long\gdef\mi@typesetMI@checkend@#1{%
  \ifx#1\mi@typesetMI@checkend%
    \par\expandafter\mi@typesetMI@checkend%
  \else%
```

We assume the line is `%` ⟨*text*⟩ instead of `%`⟨*text*⟩, thus we insert back a space that was skipped before:

```
    \space%
    \expandafter#1%
  \fi%
}
```

`\mi@checkbeginMI` Usage: `\mi@checkbeginMI`⟨*cs*⟩.

First step in the check for the begin of a metainfo: Checks if ⟨*cs*⟩ is a control sequence that can begin a metainfo.

```
\gdef\mi@checkbeginMI#1{%
```

Handle `\end`:

```
  \ifx#1\end%
    \expandafter\mi@checkdocumentend%
  \else%
```

We check if ⟨*cs*⟩ is a relevant control sequence. (these are stored in `\mi@@MIbeginnings`) Then we use `\mi@checkbeginMI@` to evaluate the result:

```
    \def\@tempa##1#1##2\@nil{\mi@checkbeginMI@{##2}#1}%
    \expandafter\expandafter\expandafter\@tempa%
      \expandafter\mi@@MIbeginnings\expandafter#1%
      \expandafter\@nil%
  \fi%
}
```

`\mi@checkbeginMI@` Support macro for `\mi@checkbeginMI`. Checks if `#2` is a relevant control sequence by checking if `#1` is empty (`false`) or not (`true`).

```
\gdef\mi@checkbeginMI@#1#2{%
  \ifx^^M#1^^M%
```

Not a relevant control sequence: Skip till the end of the line.

```
    \expandafter\mi@gobbletolineend%
  \else%
```

A relevant control sequence. If there has been any local metainfo before this macro, we finish it by inserting `\mi@@lastlocalMItext`:

```
    \ifmi@firstlocalMI\else\mi@@lastlocalMItext\fi%
```

We restore all character catcodes except for `^^M`, which we use to continue processing after the control sequence in question has been executed: (this is the reason a heading must appear on a single line, as otherweise `^^M` is executed more than once)

```
    \let^^M\mi@checkbeginMI@@%
    \mi@normalbraces%
    \mi@normalpercent%
```

In any case, a local metainfo following this macro is the first, so we set the flag accordingly:

```
    \mi@firstlocalMItrue%
    \expandafter#2%
  \fi%
}
```

`\mi@checkbeginMI@@` Support macro for `\mi@checkbeginMI@`. Is called via `^^M` at the end of a line.

```
\gdef\mi@checkbeginMI@@{%
  \let^^M\@empty%
  \mi@otherbraces%
  \mi@emptyactivepercent%
  \mi@checkbeginMI@@@%
}

\egroup
```

`\mi@checkbeginMI@@@` Support macro for `\mi@checkbeginMI` and the second step in the check for the begin of a metainfo: Checks if the line after the control sequence begins with a percent sign.

```
\def\mi@checkbeginMI@@@#1{%
  \ifx#1\mi@percentempty%
```

A percent sign: It starts a metainfo. Change to metainfo.

```
    \par%
    \expandafter\mi@typesetMI%
  \else%
```

Anything else: No metainfo follows. Ignore it.

```
    \expandafter\mi@skiplines\expandafter#1%
  \fi%
}
```

`\mi@checkbeginlocalMI` Usage: `\mi@checkbeginlocalMI`⟨*dummy*⟩⟨*next token*⟩.
⟨*dummy*⟩ is the result of an optimization of `\mi@skiplines`. (see above)

```
\def\mi@checkbeginlocalMI#1#2{%
  \ifx#2\mi@percentempty%
```

A percent sign: It starts a local metainfo. Change to metainfo.

```
    \par%
    \ifmi@firstlocalMI\mi@@firstlocalMItext\fi%
    \mi@@everylocalMItext%
```

We have begun to typeset the local metainfo, therefore we take note that any local metainfo following this one is not the first.

```
    \mi@firstlocalMIfalse%
    \expandafter\mi@typesetMI%
  \else%
```

Anything else: No, it is just a simple comment. Ignore it.

```
    \expandafter\mi@gobbletolineend\expandafter#2%
  \fi%
}
```

`\mi@checkdocumentend` Checks if the `\end` just read ends the `document` environment.

```
\def\mi@checkdocumentend{%
```

We change the catcodes of braces back to normal in order to get the name of the environment as the only parameter of `\mi@checkdocumentend@` and not only the opening brace.

```
\mi@normalbraces%
\mi@checkdocumentend@%
}
```

`\mi@checkdocumentend@` Support macro for `\mi@checkdocumentend`.

```
\def\mi@checkdocumentend@#1{%
  \def\@tempa{#1}%
  \ifx\@tempa\mi@textdocument%
```

If there has been any local metainfo before this macro, we finish it by inserting `\mi@@lastlocalMItext`:

```
    \ifmi@firstlocalMI\else\mi@@lastlocalMItext\fi%
    \expandafter\mi@gobbletolineend@any\expandafter\endinput%
  \else%
```

As are not done yet, we better set the catcode of braces to 'letter':

```
    \mi@otherbraces%
    \expandafter\mi@skiplines%
  \fi%
}
```

`\mi@textdocument` The text 'document' for `\mi@checkdocumentend`.

```
\def\mi@textdocument{document}
```

`\mi@percentempty`

```
\def\mi@percentempty{\@empty}
```

8.4.3 Gobbling text

```
\bgroup
\catcode`\^^M=\active%
```

`\mi@gobbletolineend` Gobbles up any text till the end of the line. Continues with `\mi@skiplines`. (specialization of `\mi@gobbletolineend@any`)

```
\gdef\mi@gobbletolineend{%
  \mi@gobbletolineend@any\mi@skiplines%
}
```

`\mi@gobbletolineend@any` Usage: `\mi@gobbletolineend@any {`⟨*macro*⟩`}`.

Gobbles up any text till the end of the line. Continues with ⟨*macro*⟩. The macro is `long` in case the line (stored in `#2`) contains a `\par`.

```
\long\gdef\mi@gobbletolineend@any#1#2^^M{%
  #1%
}
\egroup
```

8.4.4 Catcode changes

We define the macros for changing catcodes:

```
\mi@activenewline
\mi@verbatimnewline
\mi@emptyactivepercent
\mi@activehat
```

```
\bgroup
```

We use | as the comment character as we make % active:

```
\catcode`\|=14 %
\catcode`\^^M=\active%
\catcode`\%=\active|
\gdef\mi@activenewline{|
  \catcode`\^^M=\active|
  \let^^M\@empty|
}|
\gdef\mi@verbatimnewline{|
  \catcode`\^^M=\active|
  \def^^M{\par\@gobbletwo}|
}|
\gdef\mi@emptyactivepercent{|
  \catcode`\%=\active|
  \let%\mi@percentempty|
}|
\catcode`\^=\active|
\gdef\mi@activehat{|
  \catcode`\^\active|
  \let^\mi@hat|
}|
\egroup
```

```
\mi@normalpercent
```

```
\def\mi@normalpercent{%
  \catcode`\%=14 %
}
```

```
\mi@otherbraces
```

```
\def\mi@otherbraces{%
  \catcode`\{=12 %
  \catcode`\}=12 %
}
```

```
\mi@normalbraces
```

```
\def\mi@normalbraces{%
  \catcode`\{=1 %
  \catcode`\}=2 %
}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.

Symbols

\% 195, 205, 215
\@verbatim 7, 8
\{ 218, 222
\} 219, 223
\^ 96, 184, 194, 197, 201, 208, 210
\| 193

C

\chapter 15, 21

I

\ifmi@firstlocalMI . . . 83, 133, 159, 174
\ifmi@indexmacros 6, 32
\ignorespaces 100

M

\mi@@@hat 36, 39
\mi@@everylocalMItext 18, 78, 160
\mi@@firstlocalMItext 16, 72, 159
\mi@@lastlocalMItext . . 17, 75, 133, 174
\mi@@MIbeginnings 15, 125
\mi@activehat . . . 6, 192
\mi@activenewline 3, 192
\mi@checkbeginlocalMI 87, 156
\mi@checkbeginMI 89, 119
\mi@checkbeginMI@ 123, 129
\mi@checkbeginMI@@ 134, 141
\mi@checkbeginMI@@@ 145, 148
\mi@checkdocumentend 121, 167
\mi@checkdocumentend@ 169, 171
\mi@emptyactivepercent 4, 144, 192
\mi@firstlocalMIfalse 161
\mi@firstlocalMItrue 84, 137
\mi@gobbletolineend . 91, 131, 164, 185
\mi@gobbletolineend@any 175, 186, 188
\mi@hat . . 36, 37, 45, 211
\mi@hat@ 41, 44
\mi@hat@errbrace 50, 60, 66
\mi@hat@twohats 46, 53, 56
\mi@hat@twohats@ 54, 56
\mi@indexmacrosfalse 33
\mi@indexmacrostrue 35
\mi@indextypeset 48, 63
\mi@normalbraces . . . 98, 135, 168, 221
\mi@normalpercent 136, 214
\mi@old@verbatim . 7, 9
\mi@otherbraces . 5, 106, 143, 177, 217
\mi@percentempty 86, 103, 149, 157, 182, 206
\mi@skiplines 12, 85, 108, 153, 178, 186
\mi@textdocument 173, 181
\mi@typesetMI 97, 151, 162
\mi@typesetMI@checkend . 99, 102, 112, 113
\mi@typesetMI@checkend@ 104, 111
\mi@verbatimnewline 10, 192

S

\section 15, 24
\subsection 15, 26
\subsubsection . . 15, 28

T

\typesetmetainfo . 1, *2*