TEXmate
(comprehensive chess annotation in LATEX)
Implementation

Federico Garcia
federook@gmail.com

2005/03/13

The user's manual and a sample of the package are found as an independent document (it *uses* the package, so it has to be typeset after installation): `texmatesample.tex`. Here is the code, somewhat commented.

1 Prelimitary matters

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesPackage{texmate}[2005/03/12 v1 Chess typesetting (Federico Garcia)]
\newif\if@filling\@fillingfalse
\DeclareOption{filling}{\@fillingtrue}
\DeclareOption{notfilling}{\@fillingfalse}
\DeclareOption*{\typeout{Unknown option (`\CurrentOption')}}
\ExecuteOptions{filling}
\ProcessOptions
```

2 The pieces

By default, the pieces are referred to in the input by their initials in English (Knight is N because K is King). That's standard in the PGN format, which should be importable into TEXmate. But of course a way has to be provided for the user to change it. Six uppercase letters are the argument of `\pieceinitials`, corresponding to Pawn, Rook, kNight, Bishop, Queen, and King.

`\pieceinitials`

```
\def\pieceinitials#1{\@initials#1}
\def\@initials#1#2#3#4#5#6{%
    \def\@Pawn{#1}\def\@Rook{#2}\def\@Knight{#3}%
    \def\@Bishop{#4}\def\@Queen{#5}\def\@King{#6}%
    \lowercase{\def\@pawn{#1}\def\@rook{#2}\def\@knight{#3}%
    \def\@bishop{#4}\def\@queen{#5}\def\@king{#6}}}
```

3 Diagrams

This is the easy part, now that beautiful chess fonts have been created for skak. By the way, let's start by setting the font. The user can decide on the size with
\diagramsize the length \diagramsize. Default is 18pt. Loading the font will be done when the user calls for \diagram, since there could be a need for different-size diagrams in the same document.

```
\newlength\diagramsize
```

Some new variables have to be defined.

```
\newcount\SquareNo
\newcount\@squarecount
\newif\if@blacksq\@blacksqfalse
\def\@togglesq{\if@blacksq\@blacksqfalse\else\@blacksqtrue\fi}
\newcount\@piececode
```

\diagram The only user macro in this section is \diagram. It will first set up the stage, and then read the input with \@diagline.

```
\def\diagram#1{\font\diagramfont=skak10 at \the\diagramsize
    \@squarecount\SquareNo
    \bgroup
    \setlength\fboxsep{.8pt}%
    \fbox{\parbox{\the\SquareNo\diagramsize}{%
        \baselineskip\diagramsize\diagramfont\@diagline#1.}}\egroup}
```

\@diagline Now, \@diagline reads the diagram proper. The input can contain only numbers, the initials of pieces (set by \pieceinitials through \@diaginitials), and /, indicating the end of the board file. The diagram is input from left to right, top to bottom. Consequtive empty squares are taken by numbers, so that 2 means 'next are two empty squares.' First, let's adopt the piece initials:

```
\def\@diagpiece#1{\def\temp{#1}\symbol{%
        \ifx\temp\@pawn\if@blacksq'157\else'160\fi\else
        \ifx\temp\@rook\if@blacksq'163\else'162\fi\else
        \ifx\temp\@knight\if@blacksq'155\else'156\fi\else
        \ifx\temp\@bishop\if@blacksq'141\else'142\fi\else
        \ifx\temp\@queen\if@blacksq'154\else'161\fi\else
        \ifx\temp\@king\if@blacksq'152\else'153\fi\else
        \ifx\temp\@Pawn\if@blacksq'117\else'120\fi\else
        \ifx\temp\@Rook\if@blacksq'123\else'122\fi\else
        \ifx\temp\@Knight\if@blacksq'115\else'116\fi\else
        \ifx\temp\@Bishop\if@blacksq'101\else'102\fi\else
        \ifx\temp\@Queen\if@blacksq'114\else'121\fi\else
        \ifx\temp\@King\if@blacksq'112\else'113\fi\else
        \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi}}
```

\@diagpiece The effect of all that is that \@diagpiece will produce the character with the particular piece indicated by its argument, white if uppercase, black if lowercase, and on the appropriate square (light or black) according to \if@blacksq. Of course, it's \@diagline that calls for \@diagpiece, if it finds that the next character in

the input is *not* a number, '/,' or '.' (this latter is the delimitation to the whole diagram, not by the user, but by `\diagram` above):

```
\def\@diagline#1{\@tempcnta0 \let\next\@diagline
    \ifx#1.\let\next\relax\if@filling\@dospaces{\@squarecount}\fi
    \else\ifx#1/\def\next{\if@filling\@dospaces{\@squarecount}\fi
        \newline\@togglesq\@squarecount\SquareNo\relax
        \@diagline}%
    \else\ifcat1#12\@dospaces{#1}%
    \else\@diagpiece#1\advance\@squarecount-1 \@togglesq
    \fi\fi\fi
    \next}
```

`\@dospaces` The last thing is the interpolation of empty squares through `\@dospaces`. Its argument is either a digit input by the user, or the number of squares remaining in the current board file, which TEXmate has been keeping track of with `\@squarecount` if `filling` is an active option.

```
\def\@dospaces#1{\ifnum#1>0
    \if@blacksq\symbol{'132}\else\hphantom{\symbol{'132}}\fi
    \@togglesq\@tempcntb#1 \advance\@tempcnta1 \advance\@squarecount-1
    \ifnum\@tempcnta<\@tempcntb\let\@next\@dospaces\else
        \let\@next\@gobble\fi
    \@next\@tempcntb\fi}
```

4 Chess notation

4.1 Figurines

`\rook` `\knight` `\bishop` `\queen` `\king` `\@piece` `\@@piece` TEXmate uses the beautiful fonts that come with the skak package. Chess notation is done nowdays with figurines (instead of letters) standing for the pieces. This is the default in TEXmate, but it can be changed. The user-modifiable commands `\rook`, `\knight`, etc., hold the symbol(s) for each piece. By default, they call `\@piece`, which basically prints the character of the chess font in place (which changes with context, from `\textskakbf` to `\textskak`). The business about `x`'s category code is explained below.

```
\newcommand{\skakfamily}{\usefont{U}{skak}{m}{n}}
\DeclareTextFontCommand{\textskakbf}{\skakfamily\bfseries}
\DeclareTextFontCommand{\textskak}{\skakfamily}
\newcount\@rk\@rk'122
\newcount\@kt\@kt'116
\newcount\@bp\@bp'102
\newcount\@qn\@qn'121
\newcount\@kg\@kg'113
\DeclareRobustCommand\rook{\@piece\@rk }
\DeclareRobustCommand\knight{\@piece\@kt }
\DeclareRobustCommand\bishop{\@piece\@bp }
\DeclareRobustCommand\queen{\@piece\@qn }
\DeclareRobustCommand\king{\@piece\@kg }
\def\@piece#1{\bgroup\catcode`\x=11 \textpiece{\symbol{#1}}\egroup}
```

```
\def\@@piece#1{\def\temp{#1}%
    \ifx\temp\@Rook\rook\else
    \ifx\temp\@Knight\knight\else
    \ifx\temp\@Bishop\bishop\else
    \ifx\temp\@Queen\queen\else
    \ifx\temp\@King\king\else
    #1\fi\fi\fi\fi\fi}
```

At any point, then, the commands `\rook`, `\knight`, etc., can be used to produce the desired figurine.

4.2 Captures and checks

`\takes` There are several ways to notate captures. They are universally input with 'x,' but many styles simply do not indicate capture with any particular symbol. Others do, usually with '×,' but also with ':'. The thing is that the `x` has changed into an active character in TEXmate. It is desirable to keep this change at a minimum, because `\relax` and `\ifx` are used all over the place (by all kinds of commands, not only defined by TEXmate: for example, `x` cannot be active when the chess selected is set by `\@piece`, because LATEX runs into all kinds of confusions). So, first realization: the `x` doesn't make sense but after a category-12 character (a piece or a pawn). So it is `\@@piece` that makes the change. All delimiters change it back, to allow for user-created commands that include `x`.

```
\DeclareRobustCommand\takes{\makebox[1.2ex][c]{$\times$}}
{\catcode`\x=13 \gdefx{\takes}}
```

`\checksign` Checks are some times indicated, some times not. They are input, universally again, with a plus sign. So it will be made active too in chess mode. The definitions take place inside chess mode, below, and in the chess symbols section.

4.3 Chess symbols

skak's fonts also provide for the symbols of the *Informator*. TEXmate implements them using those characters, adjusting them for size and position (with the admittedly annoying result that they should be typed manually if the surrounding text is not in normal size). Check and mate signs can be boldface, the others are usually not, even in the main line.

```
\def\@chesssymbol#1{\bgroup\catcode`\x=11
    \smash{\textskak{\symbol{#1}}}\egroup }
\DeclareRobustCommand\checksign{\smash{\@piece{'053}}}
\DeclareRobustCommand\mate{\smash{\@piece{'155}}}
\DeclareRobustCommand\wbetter{\raisebox{-.1ex}{\@chesssymbol{'146}}}
\DeclareRobustCommand\bbetter{\raisebox{-.1ex}{\@chesssymbol{'147}}}
\DeclareRobustCommand\wBetter{\raisebox{-.35ex}{\@chesssymbol{'143}}}
\DeclareRobustCommand\bBetter{\@chesssymbol{'145}}
\DeclareRobustCommand\WBetter{{\large\@chesssymbol{'150}}}
\DeclareRobustCommand\BBetter{{\large\@chesssymbol{'151}}}
\DeclareRobustCommand\equal{=}
\DeclareRobustCommand\unclear{\raisebox{-.5ex}{{\Large\@chesssymbol{'153}}}}
```

```
\DeclareRobustCommand\compensation{{\large\@chesssymbol{'156}}}
\DeclareRobustCommand\development{{\footnotesize\@chesssymbol{'164}}}
\DeclareRobustCommand\spaceadv{{\footnotesize\@chesssymbol{'171}}}
\DeclareRobustCommand\attack{\raisebox{-.3ex}{{\large\@chesssymbol{'101}}}}
\DeclareRobustCommand\initiative{\raisebox{-.2ex}{{\large\@chesssymbol{'103}}}}
\DeclareRobustCommand\counterplay{{\large\@chesssymbol{'126}}}
\DeclareRobustCommand\zugzwang{{\small\@chesssymbol{'104}}}
\DeclareRobustCommand\withidea{{\footnotesize\@chesssymbol{'105}}}
\DeclareRobustCommand\onlymove{{\footnotesize\@chesssymbol{'106}}}
\DeclareRobustCommand\betteris{{\footnotesize\@chesssymbol{'142}}}
\DeclareRobustCommand\boardfile{\@chesssymbol{'110}}
\DeclareRobustCommand\boarddiagonal{{\small\@chesssymbol{'107}}}
\DeclareRobustCommand\boardcenter{{\small\@chesssymbol{'111}}}
\DeclareRobustCommand\kingside{\@chesssymbol{'117}}
\DeclareRobustCommand\queenside{\@chesssymbol{'115}}
\DeclareRobustCommand\weak{{\small\@chesssymbol{'170}}}
\DeclareRobustCommand\ending{{\footnotesize\@chesssymbol{'114}}}
\DeclareRobustCommand\bishops{\@chesssymbol{'141}}
\DeclareRobustCommand\oppositebishops{\@chesssymbol{'157}}
\DeclareRobustCommand\samebishops{\@chesssymbol{'163}}
\DeclareRobustCommand\unitedpawns{\@chesssymbol{'153}}
\DeclareRobustCommand\separatedpawns{\@chesssymbol{'161}}
\DeclareRobustCommand\doubledpawns{\@chesssymbol{'144}}
\DeclareRobustCommand\passedpawn{\@chesssymbol{'162}}
\DeclareRobustCommand\pawnsno{{\small\@chesssymbol{'123}}}
\DeclareRobustCommand\timetrouble{{\small\@chesssymbol{124}}}
\DeclareRobustCommand\with{{\small\@chesssymbol{'166}}}
\DeclareRobustCommand\without{{\small\@chesssymbol{'167}}}
\DeclareRobustCommand\chessetc{\@chesssymbol{'120}}
\DeclareRobustCommand\chesssee{\@chesssymbol{'154}}
```

4.4 Contexts and fonts

`\ifont` `\iifont` `\iiifont` `\ivfont`

An annotated game of chess has several contexts and fonts: the main line is usually boldface, comments are not. Some times different fonts are chosen for different levels of commentary. TEXmate defines four levels: by default, the first is boldface; the second and third are set in regular type; the fourth is italicized. Figurines come in boldface and regular. Since levels three and four will happen only after level two, there is no need for them to define the chess font.

```
\def\ifont{\bfseries\let\textpiece\textskakbf}
\def\iifont{\normalfont\let\textpiece\textskak}
\def\iiifont{\normalfont}
\def\ivfont{\itshape}
```

`\iopen` `\iclose` ⋮

Different contexts are indicated also by delimiters. Square parenthesis [ ] are the usual 2nd-level marker; parenthesis ( ) and *( )* are used for the third and fourth levels. The first level, the actual game, is not delimited. Two sets of delimiters and hooks are provided. The first (accessed with `[` in chess mode, see below) is intended for in-game, not-much-text commentaries. The second (accessed with

`\[`, and with a `t` in command names) is for freer commentary. All delimiters are user-modifiable.

```
\newcount\@commlevel
\let\iopen\relax\let\iclose\relax
\DeclareRobustCommand\iopent{\par\noindent }
\DeclareRobustCommand\icloset{\par}
\DeclareRobustCommand\iiopen{ []
\DeclareRobustCommand\iiclose{\leavevmode\unskip]\textbf{;} }
\DeclareRobustCommand\iiiopen{ (}
\DeclareRobustCommand\iiiclose{\leavevmode\unskip) }
\DeclareRobustCommand\ivopen{ (}
\DeclareRobustCommand\ivclose{\leavevmode\unskip) }
\DeclareRobustCommand\iiopent{}
\DeclareRobustCommand\iicloset{}
\DeclareRobustCommand\iiiopent{}
\DeclareRobustCommand\iiicloset{}
\DeclareRobustCommand\ivopent{}
\DeclareRobustCommand\ivcloset{}
```

4.5 Delimiting the input

There are several of the typical ways of notating moves, such as:

```
1. d4 Nf6; 2. c4 g6
1.e4 e5; 2.f4 exf4
3 Nc3 Bg7 4 e4 d6
```

Of course, we don't want to force the user into any particular one. Indeed, we want the user to be free to change the way moves are notated even within the same game. So TEXmate has to take care of spacing, punctuation, etc. Two or three spaces should behave exactly as one, or as a period.

Moreover, the chess-game typist not always is clear about the move number. Most times it's not a crucial piece of information, so one is not thinking of it; and when many levels of variations are involved, it can get pretty confusing. Later, if you realize a mistake in a move number, you will have to change all move numbers from then on...

So, the move number should be totally optional for the user. An input with at least some move numbers is easier to edit, so it's expectable that users will type move numbers now and then. In addition, games in PGN format do have numbers, so TEXmate needs to understand them.

All that means that something like a 'chess mode' is needed, in which different types of character have special meanings. I chose the character | to delimit the 'chess mode:' all chess-related stuff happens within two of these symbols.

```
\def\makebarother{\catcode`\|=12 }
\def\makebarchess{\catcode`\|=13 }
\makebarchess
\def\@chesscodes{\catcode`\ =\active \catcode`\.=\active
    \catcode`\;=\active \catcode`\[=\active \catcode`\]=\active
    \catcode`\>=\active \catcode`\+=\active }
```

```
\def\@restorecodes{\catcode`\ =10 \catcode`\.=12
    \catcode`\;=12 \catcode`\]=12 \catcode`\[=12
    \catcode`\>=12 \catcode`\x=11 \catcode`\+=12 }
```

TEXmate has to be able to understand who's turn it is at every moment, so that it types the moves, and interpolates the move numbers, properly. So far this is easy enough, it's just the boolean test `\if@white`. But there are different kinds of move, at least of Black's moves: some follow immediately after a White move, so they don't need a move number; others resume a line after a commentary, so it's customary to insert the move number and something like '. . . ' Thus, there is a second test, `\if@resuming`, that holds this information.

`\if@white`

`\if@resuming`

In principle, spaces in the input delimit the moves. But also other characters, such as `|` itself, and the commentary openers, can function as move delimiters. Several consecutive spaces should not delimit several moves. The way this all is handled is through a third test, `\if@delimited`. Each delimiting macro will delimit only if this is `false`, and then will make it `true`.

`\if@delimited`

```
\newif\if@white
\newif\if@resuming
\newif\if@delimited
```

'Delimiting' amounts to advancing the turn: if a white move was found and delimited, the next thing is a black move. The macro `\@turn` toggles `\if@white`, so that TEXmate knows what to expect next. Of course, `\if@delimited` is set to `true`.

`\@turn`

```
\def\@turn{\@delimitedtrue\if@white\@whitefalse\else\@whitetrue\fi}
```

4.6 Formatting the input

TEXmate reads the chess-mode input character by character, and decides what to do with it. If it's a number, it will read it as the move number; if it's a letter, it will interpret it as the beginning of a move, and will set up things to typeset that move, according to whose turn it is, and whether it's resuming or not. Among other things, it will turn `x` into an active character, for captures.

If the next character is not a number or a letter, it must be a command, so it will do nothing.

All of this is coded as `\@execute`. The business about 0 is to provide for castling (usually input as 0-0 or 0-0-0). The repetitious way in which it's handled is not the most elegant, but it is the safest.

`\@execute`

```
\long\def\@execute#1{\let\next\relax
    \ifcat1\noexpand#1%
        \ifnum0=#1
            \if@white
                \if@resuming
                    \def\next{\beforeno\the\move\afterno
                        \catcode`\x=\active\@@piece}%
                \else
                    \def\next{\afterb\beforeno\the\move\afterno
```

```
                        \catcode`\x=\active\@@piece}%
                \fi
            \else
                \if@resuming
                    \def\next{\beforeb
                        \advance\move1
                        \catcode`\x=\active\@@piece}%
                \else
                    \def\next{\afterw
                        \advance\move1
                        \catcode`\x=\active\@@piece}%
                \fi
            \fi
            \@delimitedfalse
            \@resumingfalse
        \else
            \def\next{\move}%
        \fi
    \else
    \ifcat a\noexpand#1%
        \if@white
            \if@resuming
                \def\next{\beforeno\the\move\afterno
                    \catcode`\x=\active\@@piece}%
            \else
                \def\next{\afterb\beforeno\the\move\afterno
                    \catcode`\x=\active\@@piece}%
            \fi
        \else
            \if@resuming
                \def\next{\beforeb
                    \advance\move1
                    \catcode`\x=\active\@@piece}%
            \else
                \def\next{\afterw
                    \advance\move1
                    \catcode`\x=\active\@@piece}%
            \fi
        \fi
        \@delimitedfalse
        \@resumingfalse
    \fi\fi
    \next#1}
```

`\@execute` makes reference to many things. `\@@piece` was defined above. But each type of move (according to `\if@white` and `\if@resuming`) is formatted in different ways, which are user-modifiable macros:

```
\newcount\move
\DeclareRobustCommand\afterno{.~}
\DeclareRobustCommand\afterw{ }
```

```
\DeclareRobustCommand\afterb{; }
\DeclareRobustCommand\beforeb{\the\move.~\dots\ }
\DeclareRobustCommand\beforeno{}
```

4.7 Chess mode

\@openchess Entering chess mode (\@openchess) involves a number of actions. First of all, the next | has to be re-defined to *exit* chess mode.

The main line of a game is usually boldface. Any text that the user wants to introduce between the moves is presumably intended as regular type. So, normal conditions are actually level 2; \@openchess will decrease the level, so that the main game becomes 1, and is typeset boldface. Then the font has to be set accordingly. Category codes are set to chess mode then, and the first character of the input is read.

\@closechess Exiting chess mode (\@closechess) un-does all this, and in addition has to delimit any move that precedes it and has not been delimited. Finally, the next move will always be a resuming one, so \if@resuming is made true.

```
\def\@openchess{\let|\@closechess
    \advance\@commlevel-1\relax
    \csname\@roman{\the\@commlevel}font\endcsname
    \@chesscodes
    \@execute}
\let|\@openchess
\def\@closechess{\let|\@openchess
    \if@delimited\else\@turn\fi
    \@restorecodes\normalfont
    \@resumingtrue\advance\@commlevel1\relax}
```

4.8 Commentary

Commentaries are groups. All font changes, modifications in the move number, the turn, the resuming state, etc., are local, so that when we step back to the previous level, the conditions in which it was left are restored.

A commentary usually mentions a move that was an alternative to the one in the game (or in the superior-level line). So, if the last move was Black's tenth move, the commentary should expect, by default, another move no. 10 by Black.

\@opencomm In (\@opencomm), move number and turn are properly modified. A group is open, the level is increased, the font of the new level is chosen, and the next character

\@closecomm is read. \@closecomm basically closes the group.

But both commands have a further feature. They insert punctuation signs, or whatever is defined in user-modifiable macros, according to the type of commentary being open. The argument to the commands is provided by [and] or by \[and \].

```
\def\@opencomm#1{%
    \@resumingtrue
    \catcode`\x=11
    \bgroup
```

```
    \if@delimited
        \@turn
    \fi
    \@delimitedtrue
    \if@white\else\advance\move-1\fi
    \advance\@commlevel1\relax
    \csname\@roman{\the\@commlevel}font\endcsname
    \csname\@roman{\the\@commlevel}#1\endcsname
    \@execute}
\def\@closecomm#1{\csname\@roman{\the\@commlevel}#1\endcsname
    \egroup}
```

4.9 Additional tools

Sometimes reference to a future move is made without the intervening moves. Macros `\dummy` and `\ddummy`, 'dummy moves,' instruct TEXmate to expect not what it was expecting, but something that comes afterwards. The former thus 'skips' a turn, the latter a complete move (by both sides).

`\dummy` `\ddummy`

```
\def\dummy{\@turn
    \if@white\advance\move1\relax\fi\@execute}
\def\ddummy{\advance\move1\relax\@execute}
```

`\white` `\black` `\white` and `\black` force the next input move to be either a white or a black move. In conjunction with typing the move number (they themselves don't try to guess what move number the user is referring to), they provide complete control as to what TEXmate expects next.

```
\def\black{\@whitefalse\@execute}
\def\white{\@whitetrue\@execute}
```

`\steplevel` `\backlevel` To complete user's control, macros `\steplevel` and `\backlevel` allow him to jump from one level of commentary to another, without having to invoke commentaries and all their side-effects.

```
\def\steplevel{\advance\@commlevel1\relax\csname
    \@roman{\the\@commlevel}font\endcsname}
\def\backlevel{\advance\@commlevel-1\relax\csname
    \@roman{\the\@commlevel}font\endcsname}
```

`\newgame` The macro `\newgame` initializes everything for a new game:

```
\def\newgame{\@whitetrue\@resumingtrue\@commlevel2 \move1 \@delimitedtrue}
```

`\threat` Another very common thing in chess commentary is threats. The `\threat` macro typesets the symbol for 'with the idea of...' (a triangle) and allows any text (its argument, delimited by `<` and `>`) to follow. (It momentarily goes out of chess mode.)

```
\long\def\threat#1{\bgroup\ifcase\@commlevel\or
    \iifont\or\iiifont\or\ivfont\or\ivfont\fi\
    \@restorecodes\catcode`\>\active\withidea}
```

4.10 Inside chess mode

Many of the functions described above will be called by active characters inside chess mode. Spaces cannot be used there, so `\iffalse\fi` is used instead of `%`. Since the period and the semicolon are treated especially by TEXmate in chess mode, `\.` and `\;` are provided to typeset those signs.

```
\def\@chessperiod{. }
\def\@chesssemicolon{; }
{\@chesscodes\iffalse
\fi\global\let\.\@chessperiod\iffalse
\fi\global\let\;\@chesssemicolon\iffalse
\fi\gdef+{\checksign}\iffalse
\fi\gdef\#{\mate}\iffalse
\fi\long\gdef #1{\ifx #1\else\iffalse
    \fi\if@delimited\else\@turn\fi\expandafter\@execute\fi#1}\iffalse
\fi\gdef.#1{ }\gdef;{ }\iffalse
\fi\global\let>\egroup\iffalse
\fi\gdef[{\@opencomm{open}}\iffalse
\fi\gdef\[{\@opencomm{opent}}\iffalse
\fi\gdef]{\@closecomm{close} }\iffalse
\fi\gdef\]{\@closecomm{closet} }}
```

5 Initialization values

```
\setlength\diagramsize{18pt}
\SquareNo8
\pieceinitials{PRNBQK}
\let\textpiece\textskak
\newgame
```
⟨/package⟩