

The package `yagusylo`*

Le T_EXnicien de surface[†]

February 27, 2009

Abstract

This package enables you to obtain a symbol without loading the package which usually provides it in order to avoid name clashes.

It could be considered as an extension of `pifont` in `technicolor`. Its behaviour is controlled by keys using `xkeyval`.

The English documentation for the user is available in `yagusylo-en.pdf`.

Résumé

Cette extension permet d'obtenir un symbole sans avoir à charger l'extension qui le fournit habituellement. Cela permet d'éviter des conflits de noms.

On peut la considérer comme une extension de `pifont` en `technicolor`. Son comportement est contrôlé à l'aide de clés grâce à `xkeyval`.

La documentation en français pour l'utilisateur est `yagusylo-fr.pdf`.

Contents

[1 The code](#)

2

*This document corresponds to the file `yagusylo v1.2, 2009/02/26, 50th anniversary edition`.

[†]le.texnicien.de.surface@wanadoo.fr

1 The code

We begin with the usual salutations and then load some packages

```
1 \RequirePackage{xifthen}
2 \RequirePackage{suffix}
3 \RequirePackage{xargs}
```

xargs requires xkeyval and xifthen requires etex, calc, ifthen, and ifmtarg so no need to ask for them again but they are all available from now on.

Two macros the behaviour of which will be controlled by options on the document body level. For the moment, no info, no warning.

```
4 \newcommand*\Y@GINFO}[2] [] {}
5 \newcommand*\Y@GWARNING}[3] {}
```

The first global key controls the symfam

```
6 \define@choicekey*+[Y@G]{yagusylo.sty}{symfam}[\Y@G@SymFamChoice\nr]%
7 {marvosym,fourier,wasysym,bbding,dingbat,ark,ifsym,ifsymgeo,%
8  ifsymgeonarrow,ifsymgeowide,ifsymclock,ifsymweather,pifont}%
9 [pifont]%
10 {\Y@GINFO{You choose \Y@G@SymFamChoice}}%
11 {\PackageError{yagusylo}%
12  {The symbols family ‘‘\Y@G@SymFamChoice’’ is not yet known.}
13  {I don’t know the family you required.\MessageBreak
14   If it’s not a typo you may consider to contact me
15   (TdS)\MessageBreak to obtain support for it.}}
```

the boolean key color will control the loading of xcolor thereafter

```
16 \define@boolkey[Y@G]{yagusylo.sty}[Y@G@]{color}[false] {}
```

the boolean key configfile will control the inputing of the configuration file, if it exists,

```
17 \define@boolkey[Y@G]{yagusylo.sty}[Y@G@]{configfile}[false] {}
```

The keys info and onerror control the behaviour of yagusylo

```
18 \define@choicekey*+[Y@G]{yagusylo.sty}{info}[\Y@G@Info\nr]%
19 {mute,normal,verbose}[normal] {} {}
20 \define@choicekey*+[Y@G]{yagusylo.sty}{onerror}[\Y@G@OnError\nr]%
21 {nice,tough}[tough] {} {}
```

It is possible to pass options to xcolor with the following key

```
22 \define@key[Y@G]{yagusylo.sty}{XcolorOptions}[!Y@G!]%
23 {\def\Y@GXcolorOptions{#1}}
```

It’s now time to set the default values of the global keys, read the options given by the user and relax the macro \nr which wont be mentioned again.

```
24 \ExecuteOptionsX[Y@G]{symfam,color,XcolorOptions,info,onerror,configfile}
25 \ProcessOptionsX[Y@G]\relax
26 \let\nr\relax
```

Depending of the value given to the key color we load xcolor or not. If the value of XcolorOptions is the default !Y@G! xcolor is loaded without any option, if not xcolor is loaded with the options provided by the user. The control of the validity of those options will be done by xcolor itself.

```
27 \ifY@G@color
28 \ifthenelse{\equal{\Y@GXcolorOptions}{!Y@G!}}
29 {\RequirePackage{xcolor}}
```

```

30 {\RequirePackage[\YGXcolorOptions]{xcolor}
31 \PackageInfo{yagusylo}{Package xcolor loaded with options:
32 \YGXcolorOptions}}
33 \define@key[Y@G]{yagusylo.sty}{symcolor}[red]{\def\YGSymColor{#1}}
34 \setkeys[Y@G]{yagusylo.sty}{symcolor}
35 \newcommand\YG@Couleur[1]{\color{#1}}
36 \else
37 \define@key[Y@G]{yagusylo.sty}{symcolor}[]{}
38 \newcommand\YG@Couleur[1]{\relax}
39 \def\YGSymColor{\relax}
40 \fi

```

We don't want the global options color, info and onerror to be changed afterwards so we disable them.

```

41 \disable@keys[Y@G]{yagusylo.sty}{color,onerror,info}

```

`\setyagusylokeys` The macro requires an argument, if it is a * the default settings of the global keys are redone, in the other case, the user is meant to provide legal pairs of key-value.

```

42 \newcommand\setyagusylokeys[1]{%
43 \ifthenelse{equal{#1}{*}}%
44 {\setkeys[Y@G]{yagusylo.sty}{%
45 symfam,symcolor,%
46 leadtype,symplace,sympos,boxwidth,before,after,%
47 head,tail}}%
48 {\setkeys[Y@G]{yagusylo.sty}{#1}}}

```

Whatever the value of onerror key, yagusylo will give, at the end of its loading, a summary of the global setup which is enforced at the beginning of the document:

```

49 \AtEndOfPackage{%
50 \PackageInfo{yagusylo}{^^J%
51 =====^^J%
52 ^^J%
53 ***** YAGUSYLO GLOBAL SETUPS *****^^J%
54 ^^J%
55 ===== GENERAL OPTIONS =====^^J%
56 ^^J%
57 option ‘‘info’’ ..... is ‘‘\YG@Info’’^^J%
58 option ‘‘onerror’’ ..... is ‘‘\YG@OnError’’^^J%
59 option ‘‘color’’ ..... is \ifY@G@color‘‘true’’\else‘‘false’’\fi^^J%
60 option ‘‘configfile’’ ... is \ifY@G@configfile‘‘true’’\else‘‘false’’\fi^^J%
61 ^^J%
62 ----- CHANGEABLE OPTIONS -----^^J%
63 ^^J%
64 \ifY@G@color
65 option ‘‘symcolor’’ ..... is ‘‘\YGSymColor’’^^J\fi
66 option ‘‘symfam’’ ..... is ‘‘\YG@SymFamChoice’’^^J%
67 option ‘‘leadtype’’ ..... is ‘‘\YG@Gyagfillleadtype’’^^J%
68 option ‘‘symplace’’ ..... is ‘‘\YG@Gyagfillsymplace’’^^J%
69 option ‘‘sympos’’ ..... is ‘‘\YG@Gyagfillsympos’’^^J%
70 option ‘‘boxwidth’’ ..... is ‘‘\YG@Gyagfillboxwidth’’^^J%
71 option ‘‘before’’ ..... is ‘‘\YG@Gyagfillbefore’’^^J%
72 option ‘‘after’’ ..... is ‘‘\YG@Gyagfillafter’’^^J%
73 option ‘‘head’’ ..... is ‘‘\the\YG@Glinehead’’^^J%
74 option ‘‘tail’’ ..... is ‘‘\the\YG@Glinetail’’^^J%

```

```

75 :::::::::::::::::::: OPTIONS FOR YAGENUMERATE ::::::::::::::::::::^^J%
76 option ‘‘firstitemnum’’ ... is ‘‘\number\Y@GEnumFirstItemNum’’^^J%
77 option ‘‘enumlength’’ ..... is ‘‘\number\Y@GEnumLength’’^^J%
78 \ifY@G@color
79 option ‘‘symcolor’’ ..... is ‘‘\Y@GEnumSymColor’’^^J\fi
80 option ‘‘symfam’’ ..... is ‘‘\Y@G@EnumSymFamChoice’’^^J%
81 -----^^J%
82 ^^J%
83 The known patterns for environment ‘‘yagenumerate’’ are:^^J%
84 ‘‘piwcr’’ (default), ‘‘piwcs’’, ‘‘pibcr’’, and ‘‘pibcs’’^^J%
85 ^^J%
86 =====%
87 \@gobbletwo}%

```

Always at the end of the loading, we redefine the info and warning/error commands:

```

88 \ifthenelse{\equal{\Y@G@Info}{verbose}}
89   {\renewcommand*\Y@GINFO}[2][0]{\PackageInfo{yagusylo}{#2}}}{%
90 \ifthenelse{\equal{\Y@G@Info}{normal}}
91   {\renewcommand*\Y@GINFO}[2][0]{%
92     \ifthenelse{#1>0}{\PackageInfo{yagusylo}{#2}}}{}}{%
93 \ifthenelse{\equal{\Y@G@OnError}{tough}}
94   {\renewcommand*\Y@GWARNING}[3]{\PackageError{yagusylo}{#1}{#3}}}{%
95 \ifthenelse{\equal{\Y@G@OnError}{normal}}
96   {\renewcommand*\Y@GWARNING}[3]{\PackageError{yagusylo}{#1}{#3}}}{%
97 \ifthenelse{\equal{\Y@G@OnError}{nice}}
98   {\renewcommand*\Y@GWARNING}[3]{\PackageWarning{yagusylo}{#1#2}}}{%

```

This is the first utility macro which is not really more than an abbreviation. Y@G is the prefix of all internal names.

```

99 \newcommand\Y@G@U@FamilyDef[1]{\fontencoding{U}\fontfamily{#1}}

```

And then two other shortcuts.

```

100 \newcommand\Y@Gif[3]{%
101   \ifthenelse{\equal{#1}{#2}}{\Y@G@U@FamilyDef{#3}}}{%
102 \newcommand\Y@GifE[4]{%
103   \ifthenelse{\equal{#1}{#2}}{\Y@G@U@FamilyDef{#3}#4}}}{%

```

The internal macro.

```

104 \newcommand\Y@Gyagding[3]{%
105   \Y@G@GetSymb{#1}\selectfont\Y@G@Couleur{#3}\char#2}}

```

\yagding This user-level macro is defined taking advantage of the features of `\newcommandx` of `xargs` which enable us to have an optional argument before and another after the mandatory one. The default values of 1st and 3rd arguments ensure that the produced ding will be chosen in current default symfam with current default colour, if there is any.

```

106 \newcommandx\yagding[3][1=\Y@G@SymFamChoice,3=\Y@G@SymColor]{%
107   \Y@G@GetSymb{#1}\selectfont\Y@G@Couleur{#3}\char#2}}

```

The `\Y@G@GetSymb` macro selects the correct font setting from the symbolic name of the symfam. There is no check mechanism for the argument is passed through the key-value mechanism and so checked at giving time.

```

108 \newcommand\Y@G@GetSymb[1]{%
109   \Y@GifE{#1}{marvosym}{mvs}{\fontseries{m}\fontshape{n}}%

```

```

110 \Y@Gif{#1}{fourier}{futs}%
111 \Y@Gif{#1}{wasysym}{wasy}%
112 \Y@Gif{#1}{bbding}{ding}%
113 \Y@Gif{#1}{dingbat}{dingbat}%
114 \Y@Gif{#1}{ark}{ark}%
115 \Y@Gif{#1}{ifsym}{ifsym}%
116 \Y@Gif{#1}{ifsymgeo}{ifgeo}%
117 \Y@GifE{#1}{ifsymgeonarrow}{ifgeo}{\fontshape{na}}%
118 \Y@GifE{#1}{ifsymgeowide}{ifgeo}{\fontshape{w}}%
119 \Y@Gif{#1}{ifsymclock}{ifclk}%
120 \Y@Gif{#1}{ifsymweather}{ifwea}%
121 \Y@Gif{#1}{pifont}{pzd}}

```

`\defdingname` Next user-level macro relies on the `\newcommandx` to enable 3 optional arguments, the last one placed after the mandatory ones. The 4th argument is used to build the name of the internal macro with `\edef` or `\xdef` depending of the value of the 2nd (optional) argument which defaults to `local`.

This macro issues an info message if `info` key has value `verbose`.

To facilitate the use of the 2nd argument we provide a shortcut for the 1st one. It's because you have to give the 1st explicitly to be able to give the 2nd explicitly as well.

```

122 \newcommandx\defdingname[5][1=*,2=local,5=\Y@GSymColor]{%
123 \Y@GINF0{You define ‘‘#4’’ with symbols family ‘‘#1’’ and color
124 ‘‘#5’’}%

```

First we treat the default, and commonest (?), case for 1st argument

```

125 \ifthenelse{\equal{#1}{*}}
in which case we do the same for the 2nd argument
126 {\ifthenelse{\equal{#2}{local}}
and then we look at the last, about color.
127 {\ifthenelse{\equal{#5}{*}}
128 {\expandafter\edef\csname Y@G@@ #4\endcsname
129 {\yagding[\Y@G@SymFamChoice]{#3}[\noexpand\Y@GSymColor]}}
130 {\expandafter\edef\csname Y@G@@ #4\endcsname
131 {\yagding[\Y@G@SymFamChoice]{#3}[#5]}}}%

```

Here the ding name will be globally defined.

```

132 {\ifthenelse{\equal{#2}{global}}
133 {\ifthenelse{\equal{#5}{*}}
134 {\expandafter\xdef\csname Y@G@@ #4\endcsname
135 {\yagding[\Y@G@SymFamChoice]{#3}[\noexpand\Y@GSymColor]}}
136 {\expandafter\xdef\csname Y@G@@ #4\endcsname
137 {\yagding[\Y@G@SymFamChoice]{#3}[#5]}}}%

```

And we come here in case the 2nd argument is neither `local` nor `global`.

```

138 {\Y@GWARNING{Value #2 not recognised} {\.\MessageBreak The possible
139 values are ‘‘local’’ (default) and ‘‘global’’. \MessageBreak I
140 will assume you wanted ‘‘local’’} {The possible values are
141 ‘‘local’’ (default) and ‘‘global’’}}}%

```

We use the same pattern of nested `\ifthenelse` when the 1st argument is explicitly provided.

```

142 {\ifthenelse{\equal{#2}{local}}

```

```

143 {\ifthenelse{\equal{#5}{*}}
144   {\expandafter\edef\csname Y@G@@ #4\endcsname
145     {\yagding[#1]{#3}[\noexpand\Y@GSymColor]}}
146   {\expandafter\edef\csname Y@G@@ #4\endcsname
147     {\yagding[#1]{#3}{#5}}}}%
148 {\ifthenelse{\equal{#2}{global}}
149   {\ifthenelse{\equal{#5}{*}}
150     {\expandafter\xdef\csname Y@G@@ #4\endcsname
151       {\yagding[#1]{#3}[\noexpand\Y@GSymColor]}}
152     {\expandafter\xdef\csname Y@G@@ #4\endcsname
153       {\yagding[#1]{#3}{#5}}}}%
154   {\Y@GWARNING{Value #2 not recognised} {.\MessageBreak The possible
155     values are ‘‘local’’ (default) and ‘‘global’’.\MessageBreak I
156     will assume you wanted ‘‘local’’} {The possible values are
157     ‘‘local’’ (default) and ‘‘global’’}}}}

```

`\yagding+` First appearance of `\WithSuffix` from suffix. The default font encoding is still U but it can be given explicitly, enabling the user to pick something in a font with different encoding. Here once again, * has a special meaning for 3rd and 4th arguments.

```

158 \WithSuffix\newcommandx\yagding+[6] [1=U,6=\Y@GSymColor] {%
159   \fontencoding{#1}\fontfamily{#2}%
160   \ifthenelse{\equal{#3}{*}}{\fontseries{#3}}%
161   \ifthenelse{\equal{#4}{*}}{\fontshape{#4}}%
162   \selectfont\Y@G@Couleur{#6}\char#5}}

163 \WithSuffix\newcommand\yagding*[1] {%
164   \ifthenelse{\isnamedefined{Y@G@@ #1}}
165     {\csname Y@G@@ #1\endcsname}
166     {[\string? #1 \string?]%
167     \Y@GWARNING{The name #1 is not defined}
168     {.\MessageBreak
169     Or you forgot to define it\MessageBreak
170     or you defined it inside an alien group.\MessageBreak
171     In any case you used it}{}}

```

`\defdingname+` And now the “grown-up” version of `\defdingname` which mixes the syntax of both `\defdingname` and `\yagding+`.

```

172 \WithSuffix\newcommandx\defdingname+[8] [1=U,2=local,8=\Y@GSymColor] {%
173   \ifthenelse{\equal{#2}{local}}%
174   {\ifthenelse{\equal{#8}{*}}%
175     {\expandafter\edef\csname Y@G@@ #7\endcsname
176       {\yagding+[#1]{#3}{#4}{#5}{#6}[\noexpand\Y@GSymColor]}}%
177     {\expandafter\edef\csname Y@G@@ #7\endcsname
178       {\yagding+[#1]{#3}{#4}{#5}{#6}{#8}}}}%
179   {\ifthenelse{\equal{#2}{global}}%
180     {\ifthenelse{\equal{#8}{*}}%
181       {\expandafter\xdef\csname Y@G@@ #7\endcsname
182         {\yagding+[#1]{#3}{#4}{#5}{#6}[\noexpand\Y@GSymColor]}}%
183       {\expandafter\xdef\csname Y@G@@ #7\endcsname
184         {\yagding+[#1]{#3}{#4}{#5}{#6}{#8}}}}%
185     {\Y@GWARNING{Value #2 not recognised} {.\MessageBreak The possible
186       values are ‘‘local’’ (default) and ‘‘global’’.\MessageBreak I
187       will assume you wanted ‘‘local’’} {The possible values are

```

188 ‘‘local’’ (default) and ‘‘global’’}}}

The `\YGfill` macro does the internal work when one of the `\yagfill(*/+)` is called. There is generally no error control for the arguments are passed by keys.

It is heavily indebted to the code of the `\Pifill` of `pifont`.

```
189 \newcommand{\YGfill}[7]%
190 {\leavevmode
191   \dimendef\YGlongi=255\relax
192   \dimendef\YGBoxActualWidth=254\relax
193   \ifthenelse{\equal{#2}{l}}{\let\Leaders=\leaders}{}%
194   \ifthenelse{\equal{#2}{x}}{\let\Leaders=\xleaders}{}%
195   \ifthenelse{\equal{#2}{c}}{\let\Leaders=\cleaders}{}%
196   \ifthenelse{\equal{#3}{a}}
197   {\def\LaBoite{\hbox{\makebox{\hspace{#6}#1\hspace{#7}}}}
198   {%
```

Now we can check the required length against the actual length of the material to be typeset.

```
199   \settowidth{\YGlongi}{#1}%
200   \ifdim#5>\YGlongi%
201   \setlength{\YGBoxActualWidth}{#5}%
202   \else
203   \setlength{\YGBoxActualWidth}{\YGlongi}%
204   \YGINFO{The choosen value for ‘‘boxwidth’’ is too short,\MessageBreak
205   I will use the natural width of the symbol: \the\YGBoxActualWidth}\fi
206   \ifthenelse{\equal{#3}{n}}
207   {\ifthenelse{(#4<0)\or{(#4>1000)}}
208   {\YGWARNING{I don’t understand what you want}
209   {\MessageBreak I assume the default}
210   {You should read the documentation [[SECyagfillboxwidth]].}%
211   \def\LaBoite{\hbox{\makebox[\YGBoxActualWidth][c]{#1}}}}%
212   {\setlength{\YGlongi}{\YGBoxActualWidth-\YGlongi}%
213   \setlength{\YGlongi}{\YGlongi/1000*#4}%
214   \def\LaBoite{\hbox{\makebox%
215   [\YGBoxActualWidth][l]{\hspace*{\YGlongi}#1\hfill}}}}}%
216   {\def\LaBoite{\hbox{\makebox[\YGBoxActualWidth][#3]{#1}}}}%
217   \Leaders\LaBoite\hfill\kern\z@}}
```

We then define the keys governing the behaviour of `\yagfill(*/+)`.

```
218 \define@choicekey*+[YG]{yagusylo.sty}{leadtype}[\YGyagfillsleadtype\nr]%
219 {l,c,x}[l]%
220 {\YGINFO{Key ‘‘leadtype’’ is ‘‘\YGyagfillsleadtype’’}}%
221 {\PackageError{yagusylo}%
222   {Possible choices for key ‘‘leadtype’’: l (default), c or x}
223   {I don’t know the type you required.\MessageBreak
224   Read the doc, please}}
225 \define@choicekey*+[YG]{yagusylo.sty}{symplace}[\YGyagfillsymplace\nr]%
226 {c,r,l,a,n}[c]%
227 {\YGINFO{Key ‘‘symplace’’ is ‘‘\YGyagfillsymplace’’}}%
228 {\PackageError{yagusylo}%
229   {Possible choices for key ‘‘symplace’’: c (default), l, r, a or n}
230   {I don’t know the place you required.\MessageBreak
231   Read the doc, please}}
```

```

232 \define@key[Y@G]{yagusylo.sty}{sympos}[0]{%
233   \ifthenelse{(#1<0)\or(#1>1000\)}
234   {\PackageError{yagusylo}
235     {Your choice ‘#1’ for key ‘sympos’ is out of bound}
236     {You should chose a interger between 0 and 1000.\MessageBreak
237       You should read the documentation}}
238   {\Y@GINFO{Keys ‘sympos’ is ‘#1’}%
239     \def\Y@Gyagfillsympos{#1}}
240 \define@key[Y@G]{yagusylo.sty}{boxwidth}[0.2in]{%
241   \ifthenelse{\dimtest{#1}<{0pt} \or \dimtest{#1}={0pt}}
242   {\PackageError{yagusylo}
243     {Your choice ‘#1’ for key ‘boxwidth’ is out of bound}
244     {You should chose a positive length.\MessageBreak
245       You should read the documentation}}
246   {\Y@GINFO
247     {You’ve asked for ‘#1’ as box width.\MessageBreak
248       In any case the box will be at least\MessageBreak
249       as large as the symbol itself}}%
250   \def\Y@Gyagfillboxwidth{#1}}
251 \define@key[Y@G]{yagusylo.sty}{before}[0pt]{%
252   \ifthenelse{\dimtest{#1}<{0pt}}
253   {\PackageError{yagusylo}
254     {Your choice ‘#1’ for key ‘before’ is out of bound}
255     {You should chose a non-negative length.\MessageBreak
256       You should read the documentation}}
257   {\Y@GINFO{Key ‘before’ is ‘#1’}%
258     \def\Y@Gyagfillbefore{#1}}

```

The key `after` has a special behaviour. If it has the default value, the length `\Y@Gyagfillafter` is made equal to the length `\Y@Gyagfillbefore`, else it is checked and an error is issued if it is negative.

```

259 \define@key[Y@G]{yagusylo.sty}{after}[!Y@G!]{%
260   \ifthenelse{\equal{#1}{!Y@G!}}{%
261     {\def\Y@Gyagfillafter{\Y@Gyagfillbefore}}%
262     {\ifthenelse{\dimtest{#1}<{0pt}}
263       {\PackageError{yagusylo}
264         {Your choice ‘#1’ for key ‘after’ is out of bound}
265         {You should chose a non-negative length.\MessageBreak
266           You should read the documentation}}
267       {\Y@GINFO{Key ‘after’ is ‘#1’}%
268         \def\Y@Gyagfillafter{#1}}}

```

And now we set the keys just defined.

```

269 \setkeys[Y@G]{yagusylo.sty}{leadtype,symplace,sympos,boxwidth,before,after}

```

`\yagfill` Here comes the first user macro to fill with dings. If the optional argument is given it must be a list of key-value pairs.

```

270 \newcommand\yagfill[2][!Y@G!]{%
271 \begingroup
272 \ifthenelse{\equal{#1}{!Y@G!}}{%
273   {\setkeys[Y@G]{yagusylo.sty}{#1}}%
274   \Y@Gfill{\yagding{#2}}%
275   {\Y@Gyagfillleadtype}{\Y@Gyagfillsymplace}%
276   {\Y@Gyagfillsympos}{\Y@Gyagfillboxwidth}%

```



```

277 {\Y@Gyagfillbefore}{\Y@Gyagfillafter}%
278 \endgroup}

```

And then its friends,

```

\yagfill* first the starred version,
279 \WithSuffix\newcommand\yagfill*[2][!Y@G!]{%
280 \begingroup
281 \ifthenelse{\equal{#1}{!Y@G!}}{%
282   }%
283   {\setkeys[Y@G]{yagusylo.sty}{#1}}%
284 \Y@Gfill{\yagding*{#2}}%
285   {\Y@Gyagfillleadtype}{\Y@Gyagfillsymplace}%
286   {\Y@Gyagfillsympos}{\Y@Gyagfillboxwidth}%
287   {\Y@Gyagfillbefore}{\Y@Gyagfillafter}%
288 \endgroup}

```

\yagfill+ then the plussed version.

```

289 \WithSuffix\newcommand\yagfill+[2][!Y@G!]{%
290 \begingroup
291 \ifthenelse{\equal{#1}{!Y@G!}}{%
292   {\setkeys[Y@G]{yagusylo.sty}{#1}}%
293 \Y@Gfill{#2}}%
294   {\Y@Gyagfillleadtype}{\Y@Gyagfillsymplace}%
295   {\Y@Gyagfillsympos}{\Y@Gyagfillboxwidth}%
296   {\Y@Gyagfillbefore}{\Y@Gyagfillafter}%
297 \endgroup}

```

Some lengths and some keys

```

298 \newlength{\Y@Glinehead}
299 \newlength{\Y@Glinetail}
300 \define@key[Y@G]{yagusylo.sty}{head}[0.5in]{%
301   \setlength{\Y@Glinehead}{#1}}
302 \define@key[Y@G]{yagusylo.sty}{tail}[0.5in]{%
303   \setlength{\Y@Glinetail}{#1}}
304 \setkeys[Y@G]{yagusylo.sty}{head,tail}
  which can be set up with the next macro
305 \newcommand\setyagline[2][2=!Y@G!]{%
306   \ifthenelse{\equal{#2}{!Y@G!}}{%
307     {\setkeys[Y@G]{yagusylo.sty}{head=#1,tail=#1}}%
308     {\setkeys[Y@G]{yagusylo.sty}{head=#1,tail=#2}}}

```

\yagline and are used by

```

309 \newcommand{\yagline}[2][!Y@G!]{%
310 {\begingroup
311   \ifthenelse{\equal{#1}{!Y@G!}}{%
312     {\setkeys[Y@G]{yagusylo.sty}{#1}}%
313     \par\noindent\hspace{\Y@Glinehead}%
314     \Y@Gfill{\yagding[\Y@G@SymFamChoice]{#2}}%
315     {\Y@Gyagfillleadtype}{\Y@Gyagfillsymplace}%
316     {\Y@Gyagfillsympos}{\Y@Gyagfillboxwidth}%
317     {\Y@Gyagfillbefore}{\Y@Gyagfillafter}%
318     \hspace{\Y@Glinetail}\kern\z@\par
319 \endgroup}

```

and his friends

```

320 \WithSuffix\newcommand\yagline*[2][!Y@G!]%
321 {\begingroup
322   \ifthenelse{\equal{#1}{!Y@G!}}{ }%
323   {\setkeys[Y@G]{yagusylo.sty}{#1}}%
324   \par\noindent\hspace{\Y@Glinehead}%
325   \Y@Gfill{\yagding*{#2}}%
326     {\Y@Gyagfillleadtype}{\Y@Gyagfillsymplace}%
327     {\Y@Gyagfillsympos}{\Y@Gyagfillboxwidth}%
328     {\Y@Gyagfillbefore}{\Y@Gyagfillafter}%
329   \hspace{\Y@Glinetail}\kern\z@\par
330 \endgroup}

331 \WithSuffix\newcommand\yagline+[2][!Y@G!]%
332 {\begingroup
333   \ifthenelse{\equal{#1}{!Y@G!}}{ }%
334   {\setkeys[Y@G]{yagusylo.sty}{#1}}%
335   \par\noindent\hspace{\Y@Glinehead}%
336   \Y@Gfill{#2}%
337     {\Y@Gyagfillleadtype}{\Y@Gyagfillsymplace}%
338     {\Y@Gyagfillsympos}{\Y@Gyagfillboxwidth}%
339     {\Y@Gyagfillbefore}{\Y@Gyagfillafter}%
340   \hspace{\Y@Glinetail}\kern\z@\par
341 \endgroup}

```

We need a macro to check the number of nested levels we are at in the list-type environments. The six arguments are: ItemLevel, YagitemizeMaxDepth, ItemMark, setyagitemize, yagitemize, the number of the section in the documentation of this package.

The warning or error — depending of the key `onerror` — is issued once when the level reached has number YagitemizeMaxDepth.

```

342 \newcommand{\Y@GLevelTest}[6]{%
343   \ifthenelse%
344     {\cnttest{\value{Y@G#1}}<{\value{Y@G#2}}}%
345     {\setcounter{Y@G#3}{\value{Y@G#1}}}%
346     {\setcounter{Y@G#3}{\value{Y@G#2}}}%
347     \ifthenelse{\cnttest{\value{Y@G#1}}={\value{Y@G#2}}}%
348     {\Y@GWARNING{Too deeply nested for your setup}
349       {\.\MessageBreak
350         I keep on using the last symbol.\MessageBreak
351         You could have a look at your last\MessageBreak
352         ‘#4’\MessageBreak First ‘#5’ too many}%
353       {You could have a look at your last\MessageBreak
354         ‘#4’\MessageBreak First ‘#5’ too many.\MessageBreak
355       You should read the documentation [[#6]].}}{}}

```

We need some global counters.

```

356 \newcounter{Y@GItemLevel}
357 \newcounter{Y@GItemMark}
358 \newcounter{Y@GYagitemizeMaxDepth}

```

`yagitemize` The first list-like environment is a yagusylo version of itemize, hence its name.

```

359 \newenvironment{x[yagitemize]}[3][1=\Y@G@SymFamChoice,3=\Y@G@SymColor]
360 {\stepcounter{Y@GItemLevel}%
361   \ifthenelse{\equal{#2}{*}}

```

The second argument is `*`.

```

362 {\YGLevelTest{ItemLevel}{YagitemizeMaxDepth}{ItemMark}
363   {setyagitemize}{yagitemize}{5.1.2}%
364   \edef\numero{\roman{YGItemMark}}%
365   \begin{list}{%
366     \yagding%
367     [\csname Y@G@symfam@niveau\numero\endcsname]%
368     {\csname Y@G@symfam@numero\numero\endcsname}%
369     [\csname Y@G@symfam@couleur\numero\endcsname]}{}}

```

The second argument is explicitly given.

```

370   {\begin{list}{\yagding[#1]{#2}{#3}}{}}%
371   {\end{list}\addtocounter{YGItemLevel}{-1}}

```

`\setyagitemize` A macro to set up the `yagitemize` environment. The argument of `\setyagitemize` must be a list of triples separated by periods. Each triple defines a pattern with values separated by comas: firstly the `symfam`, secondly the number and thirdly the color. It uses `\YGsetyagitemize`

```

372 \newcommand{\setyagitemize}[1]{%
373   \setcounter{YG@YagitemizeMaxDepth}{0}%
374   \YG@Gsetyagitemize #1.Y@G@.\@nil}

```

which is defined here with the help of `\YG@Gsetyagitemizeaux`.

```

375 \def\YG@Gsetyagitemizeaux #1\fi{%
376   \fi \YG@Gsetyagitemize #1}%
377 \def\YG@Gsetyagitemize #1.#2{%
378   \def\YG@Gfairemotif ##1,##2,##3\@YG@Gnil{%
379     \edef\YG@Gpremier{##1}\edef\YG@Gdeuxieme{##2}\edef\YG@Gtroisieme{##3}%
380   }
381   \ifx#2\@nil\relax
382   \else
383     \YG@Gfairemotif#1\@YG@Gnil
384     \stepcounter{YG@YagitemizeMaxDepth}%
385     \edef\numero{\roman{YG@YagitemizeMaxDepth}}%
386     \expandafter\edef\csname Y@G@symfam@niveau\numero\endcsname{\YG@Gpremier}
387     \expandafter\edef\csname Y@G@symfam@numero\numero\endcsname{\YG@Gdeuxieme}
388     \expandafter\edef\csname Y@G@symfam@couleur\numero\endcsname{\YG@Gtroisieme}
389     \YG@Gsetyagitemizeaux #2\fi}

```

At the end, the counter `YG@YagitemizeMaxDepth` has value the number of levels defined which is also the deepest level of nesting available before a warning — if `onerror` is `nice` — or an error.

```

390 \newcounter{YG@YagitemizeStarMaxDepth}
391 \WithSuffix\newcommand\setyagitemize*[1]{%
392   \setcounter{YG@YagitemizeStarMaxDepth}{0}%
393   \YG@Gsetyagitemizestar #1.Y@G@.\@nil}
394 \def\YG@Gsetyagitemizestaraux #1\fi{%
395   \fi \YG@Gsetyagitemizestar #1}%
396 \def\YG@Gsetyagitemizestar #1.#2{%
397   \ifx#2\@nil%
398     \relax
399   \else
400     \def\motif{#1}%
401     \stepcounter{YG@YagitemizeStarMaxDepth}%

```

```

402 \edef\numero{\roman{Y@GyagitemizeStarMaxDepth}}%
403 \expandafter\edef%
404 \csname Y@Gsymbol@listdepth\numero\endcsname{%
405 \yagding*{\motif}}%
406 \Y@Gsetyagitemizestaraux #2\fi}

407 \newcounter{Y@GItemStarLevel}
408 \newenvironment{yagitemize*}[1][!Y@G!]%
409 {\stepcounter{Y@GItemStarLevel}
410 \ifthenelse{\equal{#1}{!Y@G!}}
411 {\Y@GLevelTest{ItemStarLevel}{YagitemizeStarMaxDepth}{ItemMark}
412 {setyagitemize*}{yagitemize*}{5.2.2}}%
413 \edef\Y@GActualSymbol{\csname
414 Y@Gsymbol@listdepth\roman{Y@GItemMark}\endcsname}}
415 {\edef\Y@GActualSymbol{\yagding*{#1}}}%
416 \begin{list}{\Y@GActualSymbol}{}
417 {\end{list}\addtocounter{Y@GItemStarLevel}{-1}}

418 \newcommand\yagnumber[3]{\protect\Y@Gyagding{#1}{\arabic{#2}}{#3}}

The boolean is used to control the redefinition of \item.
419 \newboolean{Y@GitemRedefined}
Here are the keys of the enum bunch.

420 \define@choicekey*+{Y@G}{y@genum}{symfam}{\Y@G@EnumSymFamChoice\nr}%
421 {marvosym,fourier,wasysym,bbding,dingbat,ark,ifsym,ifsymgeo,%
422 ifsymgeonarrow,ifsymgeowide,ifsymclock,ifsymweather,pifont}%
423 [pifont]%
424 {\Y@GINFO{You choose \Y@G@EnumSymFamChoice}}%
425 {\PackageError{yagusylo}%
426 {The symbols family ‘‘\Y@G@EnumSymFamChoice’’ is not yet known.}
427 {I don’t know the family you required.\MessageBreak
428 If it’s not a typo you may consider to contact me
429 (TdS)\MessageBreak to obtain support for it.}}

430 \define@key[Y@G]{y@genum}{symcolor}[blue]{\def\Y@GEnumSymColor{#1}}

431 \define@key[Y@G]{y@genum}{firstitemnum}[’254]{%
432 \ifthenelse{(#1<0)\or \(#1>255\)}
433 {\PackageError{yagusylo}
434 {‘‘#1’’ is out of bound or not a number}
435 {‘‘firstitemnum’’ requires a natural number\MessageBreak
436 between 0 and 255}}%
437 {\def\Y@GEnumFirstItemNum{#1}%
438 \Y@GINFO{Key ‘‘firstitemnum’’ is
439 ‘‘\number\Y@GEnumFirstItemNum’’}}}

440 \define@key[Y@G]{y@genum}{enumlength}[10]{%
441 \ifthenelse{(#1<1)\or \(#1>255\)}
442 {\PackageError{yagusylo}
443 {‘‘#1’’ is out of bound or not a number}
444 {‘‘enumlength’’ requires a natural number\MessageBreak
445 between 1 and 255}}%
446 {\def\Y@GEnumLength{#1}%
447 \Y@GINFO{Key ‘‘enumlength’’ is ‘‘\number\Y@GEnumLength’’}}}

```

`\newenumpattern` The macro to create patterns for the `yagenumerate` environment. The macro where the pattern `foo` is kept is `\Y@GEnumPattern@foo`.

```

448 \newcommand*\newenumpattern[2]{%
449   \ifthenelse{\isnamedefined{Y@GEnumPattern@#1}}
450   {\PackageError{yagusylo}{Pattern ‘#1’ already defined}{%
451     You should choose another name\MessageBreak
452     this version does NOT provide a ‘renewenumpattern’ macro}}
453   {{\renewcommand*\Y@GINFO}[1]{%
454     \expandafter\gdef\csname Y@GEnumPattern@#1\endcsname{%
455       \setkeys[Y@G]{y@genum}{#2}}}
456     \Y@GINFO[1]{Pattern ‘#1’ defined with\MessageBreak
457       symfam=\Y@G@EnumSymFamChoice\MessageBreak
458       symcolor=\Y@G@EnumSymColor\MessageBreak
459       firstitemnum=\number\Y@G@EnumFirstItemNum\MessageBreak
460       enumlength=\Y@G@EnumLength\MessageBreak}}}

```

Here we define the 4 default patterns.

```

461 \newenumpattern{piwcr}{symfam=pifont,firstitemnum='254,enumlength=10}
462 \newenumpattern{piwcs}{symfam=pifont,firstitemnum='300,enumlength=10}
463 \newenumpattern{pibcr}{symfam=pifont,firstitemnum='266,enumlength=10}
464 \newenumpattern{pibcs}{symfam=pifont,firstitemnum='312,enumlength=10}
465 \newcommand{\Y@GSetYagEnumerate}[1]{%
466   \ifthenelse{\isnamedefined{Y@GEnumPattern@#1}}{%
467     {\csname Y@GEnumPattern@#1\endcsname}%
468     {\Y@GWARNING{The pattern ‘#1’ is not know}
469       {.\MessageBreak I select ‘piwcr’ instead}
470       {You should read the documentation [[SECyagpattern]].}%
471       \csname Y@GEnumPattern@piwcr\endcsname}}
472 \define@key[Y@G]{y@genum}{enumpattern}[piwcr]{%
473   \edef\Y@G@EnumPatternChoice{#1}%
474   \Y@GSetYagEnumerate{#1}}
475 \setkeys[Y@G]{y@genum}{symfam,symcolor,firstitemnum,enumlength}
476 \newcommand\setyagenumeratekeys[1]{%
477   \ifthenelse{\equal{#1}{*}}{%
478     {\setkeys[Y@G]{y@genum}
479       {symfam,symcolor,firstitemnum,enumlength,enumpattern}}%
480     {\setkeys[Y@G]{y@genum}{#1}}}

```

yagenumerate To create the following macros, I have resorted to already written code from the $\text{\LaTeX} 2_{\epsilon}$ kernel and the `pifont` package. I am also indebted to Arnaud SCHMIT-TBUHL on `fr.comp.text.tex` for invaluable pieces of advice.

```

481 \newcounter{Y@Gmaxitem}
482 \newcounter{Y@Gcitem}
483 \newenvironment{yagenumerate}[1][!Y@G!]{%
484   {%
485     \ifthenelse{\equal{#1}{!Y@G!}}
486     {\setkeys[Y@G]{y@genum}{symfam,symcolor,firstitemnum,enumlength}}
487     {\ifthenelse{\equal{#1}{*}}
488       {\setkeys[Y@G]{y@genum}{enumpattern=\Y@G@EnumPatternChoice}}
489       {\setkeys[Y@G]{y@genum}{#1}}}%
490   \ifnum\@enumdepth>\thr@@\toodeep\else
491   \advance\@enumdepth \@ne
492   \def\STOP{\PackageError{yagusylo}
493     {at least one item too many}

```

```

494     {you can't use more than ‘\number\Y@GEnumLength’ items}}%
495 \setcounter{Y@Gmaxitem}{\Y@GEnumFirstItemNum}
496 \addtocounter{Y@Gmaxitem}{\Y@GEnumLength}
497 \setcounter{Y@Gcitem}{\Y@GEnumFirstItemNum}
498 \ifthenelse{\boolean{Y@GitemRedefined}}{ }{
499   {\let\Y@Golditem\item
500     \def\item{\stepcounter{Y@Gcitem}
501       \ifnum\theY@Gcitem>\theY@Gmaxitem\expandafter\STOP
502       \else\expandafter\Y@Golditem\fi}
503     \setboolean{Y@GitemRedefined}{true}}
504 \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
505 \expandafter\def\csname p@enum\romannumeral\the\@enumdepth\endcsname{%
506 \expandafter\def\csname labelenum\romannumeral\the\@enumdepth\endcsname{%
507   \csname theenum\romannumeral\the\@enumdepth\endcsname}%
508 \expandafter\def\csname theenum\romannumeral\the\@enumdepth\endcsname{%
509   \yagnumber{\Y@G@EnumSymFamChoice}%
510   {enum\romannumeral\the\@enumdepth}{\Y@GEnumSymColor}}%
511 \list{\csname label\@enumctr\endcsname}{%
512   \nmbbrlisttrue
513   \def\@listctr{\@enumctr}%
514   \setcounter{\@enumctr}{\Y@GEnumFirstItemNum}%
515   \addtocounter{\@enumctr}{-1}%
516   \def\makelabel##1{\hss\llap{##1}}}}
517 \fi}{\endlist}

```

The following environment is a wrapper which enables to go back to a “classical” enumerate. It redefines `\item`

```

518 \newenvironment{notyagenum}
519   {\let\item\Y@Golditem
   and takes care of the boolean Y@GitemRedefined just in case a yagenumerate
   would be nested in the enumerate.
520   \setboolean{Y@GitemRedefined}{false}}{ }

```

If the config-file is asked for, we check that it exists before loading it.

```

521 \AtBeginDocument{%
522   \ifY@G@configfile
523   \InputIfFileExists{yagusylo.cfg}%
524   {\PackageInfo{yagusylo}{Configuration file found and read\@gobble}}%
525   {\PackageErrorNoLine{yagusylo}
526     {No configuration file found}
527     {Or yagusylo.cfg does not exist\MessageBreak
528     or it is in some place unknown of TeX.}}%
529   \fi}

```

Change History

v1.1		v1.2	
General: first version on personal		General: first version on CTAN	.. 1
site	1		

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

	Symbols	364, 379, 385–	<code>\leavevmode</code> 190
<code>\(</code>	207, 233, 432, 441	388, 402, 403,	<code>\list</code> 511
<code>\)</code>	207, 233, 432, 441	413, 415, 473, 504	<code>\llap</code> 516
<code>\@YGnil</code>	378, 383	<code>\end</code> 371, 417	
<code>\@enumctr</code>		<code>\endgroup</code> . 278, 288,	M
	504, 511, 513–515	297, 319, 330, 341	<code>\makebox</code>
<code>\@enumdepth</code>	490,	<code>\endlist</code> 517	. 197, 211, 214, 216
	491, 504–508, 510	environments:	<code>\makelabel</code> 516
<code>\@gobbletwo</code>	87	<code>yagenumerate</code> . . . <u>481</u>	<code>\motif</code> 400, 405
<code>\@listctr</code>	513	<code>yagitimize</code> <u>359</u>	
<code>\@ne</code>	491	<code>\ExecuteOptionsX</code> . . . 24	N
<code>\@nmbbrlisttrue</code>	512		<code>\newboolean</code> 419
<code>\@toodeep</code>	490		<code>\newcommandx</code> . . . 106,
		F	122, 158, 172, 305
A		<code>\fontencoding</code> . . . 99, 159	<code>\newcounter</code> 356–358,
<code>\addtocounter</code>		<code>\fontfamily</code> 99, 159	390, 407, 481, 482
	371, 417, 496, 515	<code>\fontseries</code> 109, 160	<code>\newenumpattern</code> . . .
<code>\advance</code>	491	<code>\fontshape</code> <u>448</u> , 461–464
<code>\arabic</code>	418	. 109, 117, 118, 161	<code>\newenvironmentx</code> . . . 359
			<code>\newlength</code> 298, 299
B		G	<code>\noexpand</code> . 129, 135,
<code>\begin</code>	365, 370, 416	<code>\gdef</code> 454	145, 151, 176, 182
<code>\begingroup</code>	271, 280,		<code>\noindent</code> . 313, 324, 335
	290, 310, 321, 332	H	<code>\nr</code> 6, 18,
<code>\boolean</code>	498	<code>\hbox</code> . 197, 211, 214, 216	20, 26, 218, 225, 420
		<code>\hfill</code> 215, 217	<code>\number</code> 76, 77,
C		<code>\hspace</code> 197,	439, 447, 459, 494
<code>\char</code>	105, 107, 162	215, 313, 318,	<code>\numero</code> 364, 367–369,
<code>\cleaders</code>	195	324, 329, 335, 340	385–388, 402, 404
<code>\cnttest</code>	344, 347	<code>\hss</code> 516	
<code>\color</code>	35		P
		I	<code>\PackageErrorNoLine</code> 525
D		<code>\ifdim</code> 200	<code>\PackageWarning</code> 98
<code>\defdingname</code>	<u>122</u> , 172	<code>\ifYG@color</code>	<code>\par</code> 313, 318,
<code>\defdingname+</code>	<u>172</u> 27, 59, 64, 78	324, 329, 335, 340
<code>\define@boolkey</code>	16, 17	<code>\ifYG@configfile</code>	<code>\ProcessOptionsX</code> . . . 25
<code>\define@choicekey</code>	6, 60, 522	<code>\protect</code> 418
	18, 20, 218, 225, 420	<code>\isnamedefined</code>	
<code>\define@key</code>	22, 33, 37, 164, 449, 466	R
	232, 240, 251,	<code>\item</code> 499, 500, 519	<code>\roman</code> 364, 385, 402, 414
	259, 300, 302,		<code>\romannumeral</code>
	430, 431, 440, 472	K 504–508, 510
<code>\dimendef</code>	191, 192	<code>\kern</code> . 217, 318, 329, 340	
<code>\dimtest</code>	241, 252, 262		S
<code>\disable@keys</code>	41	L	<code>\selectfont</code> 105, 107, 162
		<code>\LaBoite</code> 197,	<code>\setboolean</code> 503, 520
E		211, 214, 216, 217	<code>\setcounter</code>
<code>\edef</code>	128, 130, 144,	<code>\Leaders</code> 193–195, 217	. 345, 346, 373,
	146, 175, 177,	<code>\leaders</code> 193	392, 495, 497, 514

<code>\setkeys</code>	<code>\YG@BoxActualWidth</code> .	<code>\YG@Troisieme</code> . 379, 388
34, 44, 48, 269, 192,	<code>\YG@GWARNING</code>
273, 283, 292,	201, 203, 205,	. . . 5, 94, 96, 98,
304, 307, 308,	211, 212, 215, 216	138, 154, 167,
312, 323, 334,	<code>\YG@Gdeuxieme</code> . . 379, 387	185, 208, 348, 468
455, 475, 478,	<code>\YG@GEnumFirstItemNum</code>	<code>\YG@GXcolorOptions</code> .
480, 486, 488, 489 76, 437, 439, 23, 28, 30, 32
<code>\setlength</code> 201, 203,	459, 495, 497, 514	<code>\YG@Gyagding</code> . . . 104, 418
212, 213, 301, 303	<code>\YG@GEnumLength</code>	<code>\YG@Gyagfillafter</code> . .
<code>\settowidth</code> 199 77, 446, 72, 261,
<code>\setyagenumeratekeys</code>	447, 460, 494, 496	268, 277, 287,
. 476	<code>\YG@GEnumPatternChoice</code>	296, 317, 328, 339
<code>\setyagitemize</code> 372, 391 473, 488	<code>\YG@Gyagfillbefore</code> .
<code>\setyagline</code> 305	<code>\YG@GEnumSymColor</code> 71, 258,
<code>\setyagusylokeys</code> . . 42 79, 430, 458, 510	261, 277, 287,
<code>\stepcounter</code> . . 360,	<code>\YG@Gfairemotif</code> 378, 383	296, 317, 328, 339
384, 401, 409, 500	<code>\YG@Gfill</code> 189, 274, 284,	<code>\YG@Gyagfillboxwidth</code>
<code>\STOP</code> 492, 501	293, 314, 325, 336 70,
	<code>\YG@Gif</code> 100,	250, 276, 286,
	110–116, 119–121	295, 316, 327, 338
	<code>\YG@GifE</code> 102, 109, 117, 118	<code>\YG@Gyagfillleadtype</code>
	<code>\YG@GINFO</code> 4, 10, 89, 91, 67, 218,
	123, 204, 220,	220, 275, 285,
	227, 238, 246,	294, 315, 326, 337
	257, 267, 424,	<code>\YG@Gyagfillsymplace</code>
	438, 447, 453, 456 68, 225,
	<code>\YG@GLevelTest</code>	227, 275, 285,
 342, 362, 411	294, 315, 326, 337
	<code>\YG@Glinehead</code> 73, 298,	<code>\YG@Gyagfillsympos</code> 69,
	301, 313, 324, 335	239, 276, 286,
	<code>\YG@Glinetail</code> 74, 299,	295, 316, 327, 338
	303, 318, 329, 340	<code>\yagding</code> 106,
	<code>\YG@Glongi</code>	129, 131, 135,
 191, 199, 200,	137, 145, 147,
	203, 212, 213, 215	151, 153, 158,
<code>\YG@G@Couleur</code>	<code>\YG@GGolditem</code> 499, 502, 519	163, 176, 178,
35, 38, 105, 107, 162	<code>\YG@Gpremier</code> . . . 379, 386	182, 184, 274,
<code>\YG@G@EnumSymFamChoice</code>	<code>\YG@GSetYagEnumerate</code>	284, 314, 325,
. 80, 420, 465, 474	366, 370, 405, 415
424, 426, 457, 509	<code>\YG@Gsetyagitemize</code> .	<code>\yagding+</code> 158
<code>\YG@G@GetSymb</code> 374, 376, 377	<code>yagenumerate</code> (environ-
. 105, 107, 108	<code>\YG@Gsetyagitemizeaux</code>	ment) 481
<code>\YG@G@Info</code> 18, 57, 88, 90 375, 389	<code>\yagfill</code> . . 270, 279, 289
<code>\YG@G@OnError</code>	<code>\YG@Gsetyagitemizestar</code>	<code>\yagfill*</code> 279
. 20, 58, 93, 95, 97 393, 395, 396	<code>\yagfill+</code> 289
<code>\YG@G@SymFamChoice</code> .	<code>\YG@Gsetyagitemizestaraux</code>	<code>yagitemize</code> (environ-
. 6, 10, 12, 66, 394, 406	ment) 359
106, 129, 131,	<code>\YG@GSymColor</code>	<code>\yagline</code> 309
135, 137, 314, 359 33, 39, 65, 106,	<code>\yagnumber</code> . . . 418, 509
<code>\YG@G@U@FamilyDef</code> . .	122, 129, 135,	
. 99, 101, 103	145, 151, 158,	
<code>\YG@GActualSymbol</code> . .	172, 176, 182, 359	
. 413, 415, 416		
		Z
		<code>\z@</code> . . 217, 318, 329, 340