The English documentation of the package yagusylo*

Le TEXnicien de surface
le.texnicien.de.surface@wanadoo.fr

2009/02/26

Abstract

This package enables you to obtain a symbol whithout loading the package which usually provides it in order to avoid name clashes.

It could be considered as a extended version of pifont gone technicolor.

Résumé

La documentation de yagusylo est disponible en français sous le nom de `yagusylo-fr` et `yagusylo-fr.pdf` devrait être disponible avec extension.

Contents

*This document corresponds to the file yagusylo v1, dated 2009/02/26.

1 Introduction

I began to write this package when, for I wanted this symbol ☎ in front of my phone number in a document using the class `lettre`, I realised that both marvosym and `lettre.cls` define a macro `\fax`. Because of this name clash I was pretty disappointed.

In fact, very often, we use just some symbols of all those provided by a package such as `marovsym`. So a lot of macros are defined and loaded for pretty nothing. yagusylo limits the number of macros but cannot prevent TEX to load all the necessary fonts, which is rather good.

Actually, after some thinking, the shortest and quickest way out of the quagmire in which I was stuck was certainly not to write this package but for it was written I would have loathed to let it gather dust in some remote part of my hard disk and I uploaded to my site as a sort of pre-CTAN version.

There I rest for a while before uploading the then version — after a bit of beautification perhaps — on CTAN and I happen to reread a part of the LaTeX Companion 2nd edition and to stumble upon the pifont package. And I was appaled for it seemed that my package did not qualify any more for a quick CTANification.

I decided to add some new capabilities to yagusylo and first of all some equivalent of the environments provided by pifont. And now, after some work and not a few hints from the usual suspects of `fr.comp.text.tex` — Jean-Côme Charpentier, Arnaud Schmittbuhl, Manuel Pégourié-Gonnard — I can without too much blushing upload the true first version of my tiny work.

Before going on hard facts about the usage, just a word about its name. yagusylo is the acronym of *Yet another grand unified symbols loader*. This is kind of ironic for I think there are not so many packages entitled *unified symbols loader* if any. I leave the reader decide if "grand" is the appropriate adjective `;-)`

2 Special names and general conventions

Hereafter a "symbols family" is a set of glyphs which is, in terms of NFSS, defined by the encoding U, a family, and possibly a series, and a shape. yagusylo provides names for these symbols families which you could read in table 1, page 13.

It is often the name of the package which provides the symbols, e. g. *fourier* or *marvosym* but, for some packages, there are many families: with dingbat there are *dingbat* and *ark*.

So when you require the symbols family `marvosym`, it boils down to something as `\fontencoding{U}\fontfamily{mvs}\fontseries{m}\fontshape{n}\selectfont` plus a tiny bit of code to put some color, if the option `color` is enabled, and the fact that it is done in a group to limit the font change.

For sake of laziness, "symfam" will be an abbreviation of "symbols family". I will sometimes refer to a symbol as a *ding*.

This package uses xkeyval to handle its options. So an option is in fact a key and its value. There are global options wich can be set in the preamble within the optional argument of `\usepackage` for instance

```
\usepackage[onerror=nice, info=mute]{yagusylo}
```

which sets two global options viz. `onerror` and `mute`. The global option keys are disabled at the end of the preamble so you can't change these options in the middle of the document.

The other keys are still active at the beginning of the document and are used intensively to change the behaviour of the macros of yagusylo. They will be refered at as "local options".

They are local in that first, internally, I do not use `\gsetkeys` to set them and, secondly, in that they "respect the group limits". So, if you set some local option inside an environment, the setup will be confined to that environment and the option will find its previous value after the environment.

All the local keys affect the behaviour of almost all the commands but *not* the behaviour of the environment `yagenumerate`. There are special keys for it.

Some macros have a starred and some even a plussed version. That is that you will find along with `\somemacro` a `\somemacro*` and perhaps also a `\somemacro+`. Where the bare version `\somemacro` expects a ⟨*char-num*⟩ — an integer between 0 and 255 — the

starred version `\somemacro*` awaits a ⟨*ding-name*⟩ as defined by macros `\defdingname` or `\defdingname+`, see page 6.

The plus versions are not for the fainthearted and require a lot. Be aware and shun them accordingly `;-)`

For sake of convenience I will write `\somemacro(*)` to refer to both the macros `\somemacro` and `\somemacro*`; I will write `\somemacro(*/+)` to refer to the macros `\somemacro`, `\somemacro*`, and `\somemacro+` together.

Some macros accept a argument which can have a "normal" value or the *special* value `*`. I will then write that argument as ⟨*normal*/`*`⟩.

Part I
Usage

Remark In this document yagusylo is loaded with `\usepackage[color=true, onerror=nice]{yagusylo}`.

3 One Glyph Macros

symfam
symcolor

The first three macros provide a means of obtaining *one* glyph. Their behaviour depends, in some way, on two keys `symfam` and `symcolor` that is why I first explain how to set keys with yagusylo.

3.1 Of Keys and How to Set Them

\setyagusylokeys

You can use `\setyagusylokeys` with as its one and only one mandatory argument a list of pairs of the form `key=value`:

`\setyagusylokeys{`⟨*list of key-value pairs/*`*`⟩`}`.

To set e. g. the key `symfam` to the value ***marvosym*** and the key `symcolor` to ***gray*** you will type

```
\setyagusylokeys{symfam=marvosym, symcolor=gray}
```

and the values will be set up until the end of the group in which the command is given or until the next use of `\setyagusylokeys`.

You can use the `\setyagusylokeys` macro with the special argument `*` to return to the default values of the local keys of the package.

After `\setyagusylokeys{*}`, `symfam` has value ***pifont*** and `symcolor` ***red***, see table 2 on page 13, for a complete list of local keys and their default values.

The macro `\setyagusylokeys` does not affect the behaviour of `yagenumerate` environment.

3.2 The Macros Proper

\yagding

yagusylo provides the macro `\yagding` the syntax of which is:

`\yagding[`⟨*family*⟩`]{`⟨*char-num*⟩`}[`⟨*colour*⟩`]`

where ⟨*family*⟩ is one of the symfams. By default, ⟨*family*⟩ equals `pifont` unless you have given an other value to the key `symfam` before using `\yagding` either in the preamble or the document body.

The ⟨*char-num*⟩ is the number of the symbol in the font file which "describes" it. You can look at the section IV on page 21, to find the list of all available symbols with their families and numbers. In any case ⟨*char-num*⟩ is an integer between 0 and 255 inclusive.

The ⟨*colour*⟩ is the name of a colour known by xcolor which deals with all the gory details. By default the colour is the value of `symcolor` which is itself `red` by default.

For example, I get "✳" with `\yagding[fourier]{88}[blue]`. With `\yagding{88}` I obtain "✷", symbol defined in the `pifont` package with the default colour `red`. With `\yagding{88}[green]` I obtain "✷".

Thanks to xargs, yagusylo provides macros which accept more than one optional argument.

\defdingname

With `\defdingname` you can give a *local* or *global* name to a symbol:

`\defdingname[`⟨*family/*`*`⟩`][`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour/*`*`⟩`]`

where ⟨*family*⟩, ⟨*char-num*⟩, and ⟨*colour*⟩ have the same meanings than above. Moreover ⟨*family*⟩ and ⟨*colour*⟩ have also the same default values than above.

If ⟨*defext*⟩ has value `local` — which is the default value — the name is *local* in the sense that its existence is limited to the englobing group. To obtain a *global* definition you have to give ⟨*defext*⟩ the value `global`. Other value will result in an error, if `onerror` has value ***tough*** or in a warning and then in a *local* definition.

To be able to use the 2nd optional argument ⟨*defext*⟩ you must provide the first one. So you can define a name like this:

```
\defdingname[fourier][global]{116}{rhand}[red]
```
and then, even if that definition is made in a group, everywhere in the following part of the document you will obtain "☞" with `\yagding*{rhand}`. To help the user to obtain the usual behaviour of `\defdingname` with 2nd argument `global`, yagusylo enables the use of `*` as value of the first (optional) argument. So `\defdingname[*][global]{75}` will create, globally, the same macro as `\defdingname{75}` does locally.

The last (optional) argument ⟨*colour*/`*`⟩ leads to a special behaviour when its value is `*`. In that case, the colour of the ding will be the colour which is current at the time of use of `\yagding*` and not, as usual, if ⟨*colour*/`*`⟩ is not explicitly given, at the time of definition.

To be clearer, let's assume that at a certain time, the current `symcolor` has value ***red*** and then you write

```
\defdingname[fourier]{116}{hand}\defdingname[fourier]{116}{handvar}[*]
```
then, as long as `symcolor` is not redefined, `\yagding*{hand}` and `\yagding*{handvar}` give the same glyph ☞ *but* after a
```
\setyagusylokeys{symcolor=blue}
```
if `\yagding*{hand}` gives the same ☞, `\yagding*{handvar}` gives ☞.

Remark The yagusylo macros the name of which begins with `def`, as `\defdingname`, do not check previous existence and enable redifinition.

`\yagding*` You will then be able to use `\yagding*` to obtain the named symbol with
```
\yagding*{⟨ding-name⟩}
```
and the obtained glyph cannot be affected by the settings of the local keys except `symcolor` in the special case of a definition using `*` as 4th argument as explained above, see page 7.

For example, with `\defdingname[fourier]{116}{finger}[gray]` I define the name of the symbol "☞" and I can then get it with `\yagding*{finger}`.

In fact, the **true** name of the macro which is used internally by yagusylo is `\Y@G@@␣finger`. If with such a name there is still a clash it's because somebody did it on purpose!

3.3 For Those in the Know

`\yagding+` `\yagding+` has syntax:
```
\yagding+[⟨encoding⟩]{⟨family⟩}{⟨series⟩}{⟨shape⟩}{⟨char-num⟩}[⟨colour⟩]
```
where ⟨*char-num*⟩ and ⟨*colour*⟩ have the meaning defined above. The default value of the optionnal argument ⟨*colour*⟩ is the current colour, as above once again.

All the other arguments refer to the NFSS specification: ⟨*encoding*⟩ is the font encoding, which defaults to `U`, ⟨*family*⟩ is the font family, ⟨*series*⟩ is the font series, and ⟨*shape*⟩ is the font shape. If you want to use a glyph for which series or shape is undefined, just give `*` as argument.

So `\yagding+{futs}{*}{*}{84}[blue]` gives "⚔", `\yagding+{futs}{*}{*}{85}` gives "⚔".

`\defdingname+` `\defdingname+` has syntax:
```
\defdingname+[⟨encoding⟩][⟨defext⟩]{⟨family⟩}{⟨series⟩}
      {⟨shape⟩}{⟨char-num⟩}{⟨ding-name⟩}[⟨colour/*⟩]
```
where the arguments ⟨*ding-name*⟩, ⟨*defext*⟩, and ⟨*colour*/`*`⟩ have the same role as in `\defdingname`.

By default ⟨*encoding*⟩ which awaits a font encoding has value `U`. By default ⟨*series*⟩ — a font series — and ⟨*shape*⟩ — a font shape — have a value which leads to their getting lost. ⟨*family*⟩ would like to be set to a legal font family name.

Once the symbol name defined, you will use it with the usual `\yagding*`.

Be aware that in order to use a font encoding different from `U`, you have to declare it in the preamble as an argument of fontenc. So to be able to show you what follows I have this `\usepackage[T2C,T1]{fontenc}` in the preamble of this document and then `\yagding+[T2C]{cmr}{m}{n}{128}[blue]` gives Пъ.

4 Filling and Line

The macros `\yagline(*/+)` use `\yagfill(*/+)` to place the dings so I begin with filling.

4.1 Filling

The filling mechanism is based on the TEX commands `\leaders`, `\xleaders` and `\cleaders`. Some keys are reserved for setting the behaviour of `\yagfill(*/+)` and then govern also the behaviour of `\yagline(*/+)`.

4.1.1 The Keys

Six option keys govern the behaviour of the macros `\yagfill(*/+)`. They are `leadtype`, `symplace`, `sympos`, `boxwidth`, `before`, and `after`.

`leadtype` The key `leadtype` has default value *l* and can also have value *x* or *c*. With *l* the TEX macro `\leaders` is used, with *c* it is `\cleaders` and with *x* it is `\xleaders`. Some examples below will show the different looks, see page 17.

`symplace` The key `symplace` takes its value among *c* — default —, *r*, *l*, *a*, and *n*.

`sympos` If *n* is chosen, then the key `sympos` must be set, `sympos` requires an integer between 0 and 1,000 inclusive.

`before` `after` If *a* is chosen, then the keys `before` and `after` must be set. Both those options require a non-negative LATEX length but if you do not set `after` explicitly it will take the same value as `before`.

`boxwidth` Except when *a* is chosen, `boxwidth` must be a non-negative LATEX length. If `boxwidth` has value *0 pt* — or any other null length — then the actual box width will be the natural width of the symbol[1] used by the macros `\yagfill(*)`. In fact, that will be the case whenever the value given to `boxwidth` is less than the natural width of the symbol.

4.1.2 The Macros for Filling

`\yagfill` The macro `\yagfill` has the following syntax:

`\yagfill[`⟨*list of key-value pairs*⟩`]{`⟨*char-num*⟩`}`

where the ⟨*list of key-value pairs*⟩, if provided, sets up the value of the keys listed. In case a key is not explicitly set, it keeps its current value. So if the optional argument is not given, all the following keys have their current value: `symfam`,`symcolor`, `leadtype`, `symplace`, `sympos`, `boxwidth`, `before` , and `after`.

⟨*char-num*⟩ has the same meaning as it has for `\yagding` above.

`\yagfill*` The macro `\yagfill*` has the following syntax:

`\yagfill*[`⟨*list of key-value pairs*⟩`]{`⟨*ding-name*⟩`}`

where the ⟨*list of key-value pairs*⟩ has the same usage as with `\yagfill` but where ⟨*ding-name*⟩ must be the name of a symbol previously defined via `\defdingname(+)`.

`\yagfill+` The macro `\yagfill+` has the following syntax:

`\yagfill+[`⟨*list of key-value pairs*⟩`]{`⟨*material*⟩`}`

where ⟨*material*⟩ is something wich can be typeset and has a positive width.

You can look at some examples in the section 11, on page 17.

4.1.3 The Making-Of

The three macros use the same internal code to build the boxes used by the TEX macro `\leaders`, `\cleaders`, or `\xleaders`. The more general is clearly the `+`-ed version.

First in case `symplace` is not *a*, the box has width the value of `boxwidth` unless that value is less than the natural width of the box containing the material to be typeset in which case the natural width is assumed.

With `symplace` equals *a*, the width of the box equals the sum of the values of `before`, `after`, and the natural width of the material. In this case the material begins at distance the value of `before` from the beginning of the box.

[1] That natural width is obtained with `\settowidth`.

When `symplace` doesn't equal *a*, you have to provide a means of positionning the material.

The more general case is then to use of *n* as value of `symplace` together with a integer between 0 and 1,000 as value of `sympos`. In that case, the distance between the left end of the box and the left end of the material is given by $bw \times n/1{,}000 - 0.5mw$ where bw is the actual boxwidth, n the value of `sympos`, and mw the width of the material. If n is not an integer or is out of bound you will have an error.

You can consider "`symplace=c`", "`symplace=l`", and "`symplace=r`" as abbreviations for "`symplace=n, sympos=500`", "`symplace=n, sympos=0`", and "`symplace=n, sympos=1000`" respectively, to achieve a placement centered, on the left, and on the right respectively. I have kept the mnemomics of e. g. `\makebox`.

It could seem obvious but it costs nothing to repeat it: you can set the options governing `\yagfill` and friends with `\setyagusylokeys`, see page 18, example 5.

4.2 To Draw a Line

4.2.1 Special Keys for Lines

`head` The macros `\yagline(*/+)` use internally the equivalent of `\yagfill+` and so are subjects to the same keys but there are two extra keys to set the behaviour of `\yagline` and friends. The key `head` must be a LaTeX length, which can be negative. It fixes the distance between the left margin and the left end of the first box of the line with the proviso that the `leadtype` may make some differences in this respect.

`tail` The option key `tail` fixes the distance between the right margin and the right end of the last box of the line with the same remark about the behaviour of `\leaders`. If `tail` is not explicitly given a value, which must be a legal LaTeX length, the mecanism assumes that it has the same value as `head`.

To set both those keys, yagusylo provides the macro `\setyagline` the syntax of which is the rather unusual:

`\setyagline{`⟨*head-value*⟩`}[`⟨*tail-value*⟩`]`

where both values must be lengths. I have decided for this syntax because then head and tail are in the *natural*[2] order.

By default, `head` and `tail` have value *0.5in* and `\setyagusylokeys{*}` set them back to that value.

4.2.2 The Macros for Drawing a Line

`\yagline` The syntax of `\yagline` is

`\yagline[`⟨*list of key-value pairs*⟩`]{`⟨*char-num*⟩`}`

where ⟨*list of key-value pairs*⟩ and ⟨*char-num*⟩ have their usual meanings.

`\yagline*` The syntax of `\yagline*` is

`\yagline*[`⟨*list of key-value pairs*⟩`]{`⟨*ding-name*⟩`}`

where ⟨*list of key-value pairs*⟩ and ⟨*ding-name*⟩ have their usual meanings once again.

`\yagline+` The syntax of `\yagline+` is

`\yagline+[`⟨*list of key-value pairs*⟩`]{`⟨*material*⟩`}`

where all the arguments have the same meanings as in `\yagfill+`.

All three macros begin and end with a `\par` so `some text\yagline{40}and text` results in "some text

✈ ✈

and text". Perhaps in such a case it would be better to code `some text\yagline{41}\noindent and text` which gives "some text

✉ ✉

and text" and I was *not* refering to the chosen glyph.

[2] Natural at least for left-to-right writing!

5 Itemize

yagusylo provide two environments `yagitemize` and `yagitemize*` and two macros `\setyag itemize(*)` to set the default behaviour of each environment respectively.

5.1 Environment yagitemize

5.1.1 The Environment yagitemize...

yagitemize

Here comes the first environment of yagusylo. As it names could suggest it is a kind of `itemize` environment. It has the following syntax:

`\begin{yagitemize}[`⟨*symfam*⟩`]{`⟨*char-num*/`*`⟩`}[`⟨*colour*⟩`]`

then a certain number of `\item`s and

`\end{yagitemize}`

as usual.

The mandatory argument ⟨*char-num*⟩ must be a number in the above defined meaning or a star `*` in which case the behaviour of the environment changes a little: in such a case, `yagitemize` relies on the default values you will have already provided via `\setyagitemize` at which we will look below.

You can nest as many `yagitemize` as you want but do not complaint if too much results in too ugly. Moreover, because `yagitemize` rests on the well known and ubiquitous `list` environment, LaTeX could complain about too many nested lists at a certain time and don't forget that a `quote` environment is also a `list`.

5.1.2 And How to Set It Up

\setyagitemize

The macro `\setyagitemize` enables you to define different symbols for the different level of nesting of the `yagitemize` environment. It takes an only mandatory argument which must have the following form:

`symfam1, number1, colour1. symfam2, number2, colour2. ... symfam`n`, number`n`, colour`n

With a period "`.`" you change depth level. For each level you have to provide three values separated by commas. The first one is a *symfam* as defined above, the second is the number of the required symbol and the third is the colour of that symbol. I have not provided means of considering default values. All three must be explicitly set up.

When `yagitemize` encounters the $n+1$th level, where n is the number of the last given triplet, it emits a warning or an error, depending of the value of the global key `onerror`, and if `onerror` has not the value ***`tough`***, it assumes the n-th level setup for the subsequent level.

I have taken measures to ensure that the first `yagitemize` will use the first definition given in `\setyagitemize`, at the price of some new LaTeX counters, whichever level of nested list you begin it.

5.2 Environment yagitemize*

5.2.1 The Environment yagitemize*...

yagitemize*

The `yagitemize*` environment has the following syntax:

`\begin{yagitemize*}[`⟨*ding-name*⟩`]`

its optional argument, if provided, must be a symbol name, as usual for a starred macro. When no argument ⟨*ding-name*⟩ is provided, it uses the default setup defined via `\setyag itemize*`.

5.2.2 And How to Set It Up

\setyagitemize*

The macro `\setyagitemize*` enables you to define different symbols for the different level of nesting of the `yagitemize*` environment. It takes an only mandatory argument which must have the following form:

`dingname1. dingname2. ... . dingname`n

where each dingname*k* must be a valid name defined with \defdingname(+), see page 19, example 8.

The mechanism is analog to that of \setyagitemize and \setyagitemize* has on the environment yagitemize* the same effect as the non-starred macro has on the non-starred environment.

Remark I do not provide a plussed version of the yagitemize environment for it is possible to define symbol names with \defdingname+ and use them in \setyagitemize*.

6 Enumerate

As yagitemize mimics the usual LaTeX itemize, yagenumerate apes LaTeX enumerate but with a yagusylo dressing ;-)

6.1 Special Keys for yagenumerate

The behaviour of the yagenumerate environment is controlled by the following keys: symfam, symcolor, firstitemnum, enumlength , and enumpattern. Both symfam and symcolor, in this context, are different from the *non special local keys* symfam and symcolor. We could say that there are two bunches of keys, one for yagenumerate — refered at as the *enum bunch* — and the other — which I will refer to as the *general bunch* — for all the rest and that, even if they looks alike, two keys attached to different bunches do not open the same doors.

symfam [enum]
symcolor [enum]

Nonetheless, both symfam and symcolor of the enum bunch do control the symfam and the colour of the dings used in the enumeration. At the beginning of the document — after \begin{document} — they have values pifont and blue respectively.

firstitemnum

In a yagenumerate environment, each \item increment a counter which will point to the glyph used for that item. The number, in the symfam as usual, of the first such glyph is set by firstitemnum which defaults to 172.

enumlength

With enumlength we control the number of items which may appear on the same level of yagenumerate. Its default value is 10. After that you will have an error whatever the value of the key onerror.

enumpattern

The special key enumpattern is even more special. You will read more detailled explanations about it in the following section.

6.2 Patterns and how to create them

A pattern for yagenumerate is a means of keeping and calling a whole set of special keys.

There are four pre-defined pattern, viz. piwcr, piwcs, pibcr , and pibcs. The defaut pattern is piwcr. In those names pi means pifont; c is for circle; w for white and b for black which could be a bit misleading as you will soon see; s is for sans-serif and r for roman. They all limit the enumerate length to 10.

Here are the first number for those patterns:

- ☞ piwcr: ①
- ☞ piwcs: ①
- ☞ pibcr: ❶
- ☞ pibcs: ❶

\newenumpattern

You can defined your own pattern with \newenumpattern the syntax of which is:

\newenumpattern{⟨*patname*⟩}{⟨*list of key-value pairs*⟩}

in which ⟨*patname*⟩ is the name of the pattern and ⟨*list of key-value pairs*⟩ contains at least symfam, firstitemnum, and enumlength. If symcolor is not used, the colour of the pattern will be the default one at the time of the definition. I have not provided a mechanism similar to that of \defdingname, feel free to ask if you find it would be useful.

6.3 The environment yagenumerate

yagenumerate

The environment `yagenumerate` begins with

`\begin{yagenumerate}[`⟨*list of key-value pairs/*`*`⟩`]`

and inside you will use `\item` as in a usual LaTeX `enumerate`.

If there is no argument, i. e. you type something like

```
\begin{yagenumerate}
    \item ...
```

then the aspect of the enumeration is provided by the current values of `symfam`, `symcolor`, `firstitemnum`, and `enumlength`.

If the argument is a `*` then the aspect is controlled by the current default pattern.

Lastly the aspect is determined by the list of key-value pairs provided by the user. The keys not explicity provided will have their default values.

Because `yagenumerate` redefines `\item` you can't use a normal `enumerate` nested in a `yagenumerate` without using the environment `notyagenum` as a kind of wrapper of the LaTeX `enumerate`, see page 20, example 9.

The limits of nesting are those of LaTeX.

6.4 To Set the Environment

\setyagenumeratekeys

To set the keys which govern the aspect of `yagenumerate` you can use the macro `\setyag enumeratekeys` which has a syntax analog to that of `\setyagusylokeys`, see page 6.

With `\setyagenumeratekeys{*}` keys `symfam`, `symcolor`, `firstitemnum`, `enumlength`, and `enumpattern` revert to their default values.

7 Miscellaneous

I put here some items I have not yet been able to insert cleverly in the course of the documentation.

7.1 Configuration file

configfile

It is possible to use a configuration file. yagusylo may read the file `yagusylo.cfg` if you have set the boolean key `configfile` to ***true***. Its default value is ***false***. The file `yagusylo.cfg` must be findable by TeX. In case it is not an error will be produced.

7.2 Colours and How to Get Rid of It

color

As stated before the colour management is devoluted to xcolor if the key `color` is set, globally, to ***true***. That provides two means of reverting to black on white only.

The first method is simply to change the value of `color` to ***false***. All the colours of yagusylo will be then turned off.

The second method is to pass the option `monochrome` to xcolor. To do so you will load yagusylo thus:

```
\usepackage[color=true, XcolorOptions=monochrome]{yagusylo}
```

XcolorOptions

I will seize the opportunity of mentioning `XcolorOptions` to add this: if you want to pass more than one options to xcolor you have to list them inside curly braces thus:

```
\usepackage[color=true, XcolorOptions={monochrome, table}]{yagusylo}
```

Part II
Summary of usage

8 The Package Keys

The table 1 lists the symfams known to the day by yagusylo, the symfams are also the possible values of the key `symfam`.

package	symfam	package	symfam
`pifont`	pifont	`marvosym`	marvosym
`ifsym`	ifsym	`fourier`	fourier
	ifsymgeo	`wasysym`	wasysym
	ifsymgeonarrow	`bbding`	bbding
	ifsymgeowide	`dingbat`	dingbat
	ifsymweather		ark
	ifsymclock		

Table 1: The Symfams

The table 2 shows all the option keys, default values and other possible values. Self-understandingly, when I write "any length" it must be understood that that length has to make sense in the context.

key	default value	other possible values
	Global option keys	
`info`	normal	verbose, mute
`onerror`	tough	nice
`color`	false	true
`XcolorOptions`		list of options known by xcolor
`configfile`	false	true
	Local option keys, general bunch	
`symfam`	pifont	see table 1
`symcolor`	red	any colour known by xcolor
`leadtype`	l	c, x
`symplace`	c	l, r, a, n
`sympos`	0	integer between 0 and 1,000 inclusive
`boxwidth`	0.2 in	any non-negative length
`before`	0 pt	any non-negative length
`after`	0 pt	any non-negative length
`head`	36.135 pt	any length
`tail`	36.135 pt	any length
	Local option keys, enum bunch	
`firstitemnum`	172	integer between 0 and 255 inclusive
`enumlength`	10	integer
`symcolor`	blue	any colour known by xcolor
`symfam`	pifont	see table 1

Table 2: The Keys of yagusylo

9 The Commands and Environments

I give here all the possible usages of the yagusylo macros and environments.

I will use the following ⟨*denomination*⟩s to refer to some well defined objects:

- ⟨*char-num*⟩: an integer between 0 and 255 inclusive,

♄ ⟨*number*⟩: an integer for which some additional properties could be provided,

♉ ⟨*symfam*⟩: the symbolic name of a symfam as listed in table 1 ,

☿ ⟨*colour*⟩: the symbolic name of a colour known by xcolor,

♇ ⟨*ding-name*⟩: the name of a ding as defined with `\defdingname(+)` ,

♉ ⟨*defext*⟩: the "extension" of the definition, can be `local` — default — or `global`,

♊ ⟨*G-list*⟩: a list which consists of any number of key-value pairs in which the keys are attached to the general bunch, see page 13,

♋ ⟨*E-list*⟩: a list which consists of any number of key-value pairs in which the keys are attached to the enum bunch, see page 13,

♍ ⟨*length*⟩: any LaTeX length,

♎ ⟨*P-length*⟩: any non-negative LaTeX length.

By the way, the preceding yagenumaration has been obtain with
`\begin{yagenumerate}[symfam=wasysym, firstitemnum=88, enumlength=14, symcolor=purple]`

9.1 One Glyph Macros

`\yagding{`⟨*char-num*⟩`}`
`\yagding{`⟨*char-num*⟩`}[`⟨*colour*⟩`]`
`\yagding[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}`
`\yagding[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}[`⟨*colour*⟩`]`
`\yagding{`⟨*ding-name*⟩`}`

`\yagding+[`⟨*encoding*⟩`]{`⟨*family*⟩`}{`⟨*series*/`*`⟩`}{`⟨*shape*/`*`⟩`}{`⟨*char-num*⟩`}[`⟨*colour*⟩`]`
where ⟨*encoding*⟩ is a font encoding (default `U`), ⟨*family*⟩ a font family, ⟨*series*⟩ a font series — use `*` to provide no series —, ⟨*shape*⟩ a font shape — use `*` to provide no shape.

`\defdingname{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}`
`\defdingname[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}`
`\defdingname[*]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}`
`\defdingname[`⟨*symfam*⟩`] [`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}`
`\defdingname[*] [`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}`
`\defdingname{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour*⟩`]`
`\defdingname[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour*⟩`]`
`\defdingname[*]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour*⟩`]`
`\defdingname[`⟨*symfam*⟩`] [`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour*⟩`]`
`\defdingname[*] [`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour*⟩`]`
`\defdingname{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[*]`
`\defdingname[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[*]`
`\defdingname[*]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[*]`
`\defdingname[`⟨*symfam*⟩`] [`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[*]`
`\defdingname[*] [`⟨*defext*⟩`]{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[*]`

`\defdingname+[`⟨*enc*⟩`] [`⟨*defext*⟩`]{`⟨*family*⟩`}{`⟨*series*⟩`}`
`{`⟨*shape*⟩`}{`⟨*char-num*⟩`}{`⟨*ding-name*⟩`}[`⟨*colour*/`*`⟩`]`

`\setyagusylokeys{`⟨*G-list*⟩`}`
`\setyagusylokeys{*}`

9.2 Filling and Line Drawing

`\yagfill{`⟨*char-num*⟩`}`
`\yagfill[`⟨*G-list*⟩`]{`⟨*char-num*⟩`}`

`\yagfill*{`⟨*ding-name*⟩`}`
`\yagfill*[`⟨*G-list*⟩`]{`⟨*ding-name*⟩`}`

`\yagfill+{`⟨*material*⟩`}`
`\yagfill+[`⟨*G-list*⟩`]{`⟨*material*⟩`}`
where ⟨*material*⟩ is something which can be typeset and has positive width. **Caution**: do not expect it to work with just anything!

`\setyagline{`⟨*length*⟩`}`
`\setyagline{`⟨*length*⟩`}[`⟨*length*⟩`]`

`\yagline{`⟨*char-num*⟩`}`
`\yagline[`⟨*G-list*⟩`]{`⟨*char-num*⟩`}`
`\yagline*{`⟨*ding-name*⟩`}`
`\yagline*[`⟨*G-list*⟩`]{`⟨*ding-name*⟩`}`
`\yagline+{`⟨*material*⟩`}`
`\yagline+[`⟨*G-list*⟩`]{`⟨*material*⟩`}`

9.3 Itemize and Enumerate (Yagusylo Way)

All the environments are list-like, inside you have to use `\item` to achieve some meaningfull typesetting but it is up to you. I only give the syntax of the beginning of environments for I assume you know how to `end` them `;-)`

`\begin{yagitemize}{`⟨*char-num*⟩`}`
`\begin{yagitemize}{*}`
`\begin{yagitemize}[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}`
`\begin{yagitemize}[`⟨*symfam*⟩`]{*}`
`\begin{yagitemize}{`⟨*char-num*⟩`}[`⟨*colour*⟩`]`
`\begin{yagitemize}{*}[`⟨*colour*⟩`]`
`\begin{yagitemize}[`⟨*symfam*⟩`]{`⟨*char-num*⟩`}[`⟨*colour*⟩`]`
`\begin{yagitemize}[`⟨*symfam*⟩`]{*}[`⟨*colour*⟩`]`

`\begin{yagitemize*}`
`\begin{yagitemize*}[`⟨*ding-name*⟩`]`

`\setyagitemize{`⟨*special list*⟩`}`
whith
⟨*special list*⟩ = ⟨*triple*⟩`.` ⋯ ⟨*triple*⟩`.`⟨*triple*⟩
where
⟨*triple*⟩ = ⟨*symfam*⟩`,`⟨*char-num*⟩`,`⟨*colour*⟩
with the usual meaning of ⟨*symfam*⟩, ⟨*char-num*⟩, and ⟨*colour*⟩.

`\setyagitemize* {`⟨*list of ding-names*⟩`}`
whith
⟨*list of ding-names*⟩ = ⟨*ding-name*⟩`.` ⋯ ⟨*ding-name*⟩`.`⟨*ding-name*⟩
with the usual meaning of ⟨*ding-name*⟩.

`\begin{yagenumerate}`
`\begin{yagenumerate}[`⟨*E-list*⟩`]`
`\begin{yagenumerate}[*]`

`\setyagenumeratekeys{`⟨*E-list*⟩`}`
`\setyagenumeratekeys{*}`

`\newenumpattern{`⟨*patname*⟩`}{`⟨*E-list*⟩`}`
where ⟨*patname*⟩ is a name you can later use as value for the key `enumpattern`.

Part III
Examples

10 \yagding, \defdingname, and \yagding*

1 — \defdingname

```
  \setyagusylokeys{symfam=fourier}
  \begin{quote}
inside \texttt{quote} environment\par
  \defdingname{116}{lHand}\yagding*{lHand}
  \quad
  \defdingname{116}{lHandStar}[*]\yagding*{lHandStar}
  \quad
  \defdingname[*][global]{116}{gHandRed}[red]\yagding*{gHandRed}
  \quad
  \defdingname[*][global]{116}{gHandStar}[*]\yagding*{gHandStar}
  \quad \yagding{117}

  \setyagusylokeys{symcolor=blue}
  \yagding*{lHand}\quad\yagding*{lHandStar}\quad
  \yagding*{gHandRed}\quad\yagding*{gHandStar}\quad \yagding{117}
  \end{quote}
outside \texttt{quote} environment\par
  \yagding*{lHand}\quad\yagding*{lHandStar}\quad
  \yagding*{gHandRed}\quad\yagding*{gHandStar}\quad \yagding{117}

  \setyagusylokeys{symcolor=green, symfam=pifont}
  \yagding*{lHand}\quad\yagding*{lHandStar}\quad
  \yagding*{gHandRed}\quad\yagding*{gHandStar}\quad \yagding{117}
```

inside quote environment

outside quote environment

[? lHand ?] [? lHandStar ?]

[? lHand ?] [? lHandStar ?]

2 — \yagding+

```
  \yagding+{logo}{m}{n}{77}[blue]\yagding+{logo}{m}{n}{69}[red]%
  \yagding+{logo}{m}{n}{84}[gray]\yagding+{logo}{m}{n}{65}[black]%
  \yagding+{logo}{m}{n}{80}[orange]\yagding+{logo}{m}{n}{79}[purple]%
  \yagding+{logo}{m}{n}{83}[brown]\yagding+{logo}{m}{n}{84}[green]
```

METAPOST

11 \yafgill and \yagfill*

11.1 The Key leadtype

3 — leadtype

```
thinggummy\yagfill{84}kinda big%
\setyagusylokeys{boxwidth=2cm,symcolor=gray}\par
```

```
thinggummy\yagfill{84}kinda big\par
thing\yagfill{84}kinda very very big\par
thinggummy\yagfill[leadtype=x]{84}kinda big\par
thing\yagfill[leadtype=x]{84}kinda very very big\par
thinggummy\yagfill[leadtype=c]{84}kinda big\par
thing\yagfill[leadtype=c]{84}kinda very very big
```

thinggummy ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ kinda big
thinggummy ✶ ✶ ✶ ✶ kinda big
thing ✶ ✶ ✶ kinda very very big
thinggummy ✶ ✶ ✶ ✶ kinda big
thing ✶ ✶ ✶ ✶ kinda very very big
thinggummy ✶ ✶ ✶ ✶ kinda big
thing ✶ ✶ ✶ ✶ kinda very very big

and the same thing in which I show the boxes used by TEX to type the leaders:

thinggummy ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ ✶ kinda big
thinggummy ✶ ✶ ✶ ✶ kinda big
thing ✶ ✶ ✶ kinda very very big
thinggummy ✶ ✶ ✶ ✶ kinda big
thing ✶ ✶ ✶ ✶ kinda very very big
thinggummy ✶ ✶ ✶ ✶ kinda big
thing ✶ ✶ ✶ ✶ kinda very very big

11.2 The Key symplace

11.2.1 Values other than a

5 — exemple

```
\setyagusylokeys{boxwidth=2cm, symcolor=blue}%
Caversham Heights \yagfill[symplace=c]{87}Bradshaw\par
Caversham Heights \yagfill[symplace=l]{87}Bradshaw\par
Caversham Heights \yagfill[symplace=r]{87}Bradshaw\par
Caversham Heights \yagfill[symplace=n,sympos=250]{87}Bradshaw\par
```

Caversham Heights ✶ ✶ ✶ Bradshaw
Caversham Heights ✶ ✶ ✶ Bradshaw
Caversham Heights ✶ ✶ ✶ Bradshaw
Caversham Heights ✶ ✶ ✶ Bradshaw

11.2.2 Value a

6 — symplace, before, and after

```
The Squire of High Potternews
\yagfill[symplace=a, before=0.1in]{84}Jurisfiction\par
The Squire of High Potternews
\yagfill[symplace=a, before=0.1in, after=0.3in]{84}Jurisfiction\par
The Squire of High Potternews
\yagfill[symplace=a, after=0.3in]{84}Jurisfiction\par
The Squire of High Potternews
\yagfill[leadtype=x, symplace=a,
   before=0.1in, after=0.3in]{84}Jurisfiction \par
```

The Squire of High Potternews ✶ ✶ ✶ ✶ ✶ ✶ Jurisfiction
The Squire of High Potternews ✶ ✶ ✶ ✶ Jurisfiction
The Squire of High Potternews ✶ ✶ ✶ ✶ ✶ Jurisfiction
The Squire of High Potternews ✶ ✶ ✶ ✶ Jurisfiction

7 — \yagfill+

```
\defdingname[fourier]{116}{mainv}[green]
\defdingname[fourier]{116}{mainb}[blue]
\defdingname[fourier]{116}{mainr}[red]
\yagfill+[boxwidth=1.25cm]{\yagding*{mainv}%
  \yagding*{mainb}\yagding*{mainr}}
```

12 Environments yagitemize and yagitemize*

12.1 Environment yagitemize*

8 — \yagitemize* and \setyagitemize*

```
\defdingname[fourier]{116}{mainv}[green]
\defdingname[fourier]{116}{mainb}[blue]
\defdingname[fourier]{116}{mainr}[red]
\defdingname[fourier]{116}{maing}[gray]
\setyagitemize*{mainv. mainb. mainr. maing}
\begin{yagitemize*}\item A\begin{yagitemize*}\item B
\begin{yagitemize*}\item C\begin{yagitemize*}\item D
\begin{yagitemize*}\item E
\begin{yagitemize*}\item F \item G \end{yagitemize*}
\item H\end{yagitemize*}  \item I
\end{yagitemize*}  \item J \end{yagitemize*}  \item K
\end{yagitemize*}  \item L\end{yagitemize*}
```

- A
  - B
    - C
      - D
        - E
          - F
          - G

☞ H

☞ I

☞ J

☞ K

☞ L

With such a setup, for this document has `onerror=nice` at the loading of yagusylo, you'll find the following text in the `.log` file:

```
Package yagusylo Warning: Too deeply nested for your setup.
(yagusylo)                I keep on using the last symbol.
(yagusylo)                You could have a look at your last
(yagusylo)                ‘‘setyagitemize’’
(yagusylo)                First ‘‘yagitemize*’’ too many on input line ***.
```

in which *** would give the number of the line on which is the fifth `\begin{yagitemize*}` for we provided explicit setup for four levels only.

9 — yagenumerate and notyagenum

```
\begin{yagenumerate}
  \item Thursday Next;
    \begin{notyagenum}
      \begin{enumerate}
      \item Light armoured brigade;
      \item SpecOps 27;
      \end{enumerate}
    \end{notyagenum}
  \item Landen Park-Lane;
\end{yagenumerate}
```

① Thursday Next;

(a) Light armoured brigade;

(b) SpecOps 27;

② Landen Park-Lane;

13 The configuration file

Here is the core of the configuration file `yagusylo.cfg` provided, as an example, with this package:

```
\defyagenumpattern{wastrol}{symfam=wasysym,
  firstitemnum=88, enumlength=14, symcolor=purple}
```

Part IV
Showcase

14 From the package pifont

14.1 The Symbols of Symfam `pifont`

33 : ✁	34 : ✂	35 : ✃	36 : ✄	37 : ☎	38 : ✆
39 : ✇	40 : ✈	41 : ✉	42 : ☛	43 : ☞	44 : ✌
45 : ✍	46 : ✎	47 : ✏	48 : ✐	49 : ✑	50 : ✒
51 : ✓	52 : ✔	53 : ✕	54 : ✖	55 : ✗	56 : ✘
57 : ✙	58 : ✚	59 : ✛	60 : ✜	61 : ✝	62 : ✞
63 : ✟	64 : ✠	65 : ✡	66 : ✢	67 : ✣	68 : ✤
69 : ✥	70 : ✦	71 : ✧	72 : ★	73 : ☆	74 : ✪
75 : ✫	76 : ✬	77 : ✭	78 : ✮	79 : ✯	80 : ✰
81 : ✱	82 : ✲	83 : ✳	84 : ✴	85 : ✵	86 : ✶
87 : ✷	88 : ✸	89 : ✹	90 : ✺	91 : ✻	92 : ✼
93 : ✽	94 : ✾	95 : ✿	96 : ❀	97 : ❁	98 : ❂
99 : ❃	100 : ❄	101 : ❅	102 : ❆	103 : ❇	104 : ❈
105 : ❉	106 : ❊	107 : ❋	108 : ●	109 : ❍	110 : ■
111 : ❑	112 : ❒	113 : ❑	114 : ❒	115 : ▲	116 : ▼
117 : ◆	118 : ❖	119 : ◗	120 : ❙	121 : ❚	122 : ❚
123 : ❛	124 : ❜	125 : ❝	126 : ❞		

161 : ❡	162 : ❢	163 : ❣	164 : ❤	165 : ❥	166 : ❦
167 : ❧	168 : ♣	169 : ♦	170 : ♥	171 : ♠	172 : ①
173 : ②	174 : ③	175 : ④	176 : ⑤	177 : ⑥	178 : ⑦
179 : ⑧	180 : ⑨	181 : ⑩	182 : ❶	183 : ❷	184 : ❸
185 : ❹	186 : ❺	187 : ❻	188 : ❼	189 : ❽	190 : ❾
191 : ❿	192 : ①	193 : ②	194 : ③	195 : ④	196 : ⑤
197 : ⑥	198 : ⑦	199 : ⑧	200 : ⑨	201 : ⑩	202 : ❶
203 : ❷	204 : ❸	205 : ❹	206 : ❺	207 : ❻	208 : ❼
209 : ❽	210 : ❾	211 : ❿	212 : ➔	213 : →	214 : ↔
215 : ↕	216 : ↘	217 : ➙	218 : ➚	219 : ➛	220 : ➜
221 : ➝	222 : ➞	223 : ➟	224 : ➠	225 : ➡	226 : ➢
227 : ➣	228 : ➤	229 : ➥	230 : ➦	231 : ➧	232 : ➨
233 : ➩	234 : ➪	235 : ➫	236 : ➬	237 : ➭	238 : ➮
239 : ➯	240 :	241 : ➱	242 : ➲	243 : ➳	244 : ➴
245 : ➵	246 : ➶	247 : ➷	248 : ➸	249 : ➹	250 : ➺
251 : ➻	252 : ➼	253 : ➽	254 : ⇛		

15 From the package ifsym

15.1 The Symbols of Symfam `ifsym`

0 :  1 : 2 : 3 : 4 : 5 :
6 :  7 : 8 : 9 : 10 : 11 :
12 : 

14 : 

16 :  17 : 18 : 19 : 20 : 21 :
22 :  23 : 24 : 25 : 26 : 27 :
28 :  29 : 30 : 31 : 32 : 33 :
34 :  35 : 36 : 37 : 38 : 39 :
40 :  41 :  42 :

45 : 46 :

48 :  49 : 50 : 51 : 52 : 53 :
54 : 55 : 56 : 57 : 58 : 59 :
60 :  61 : 62 : 63 :

68 :

72 :

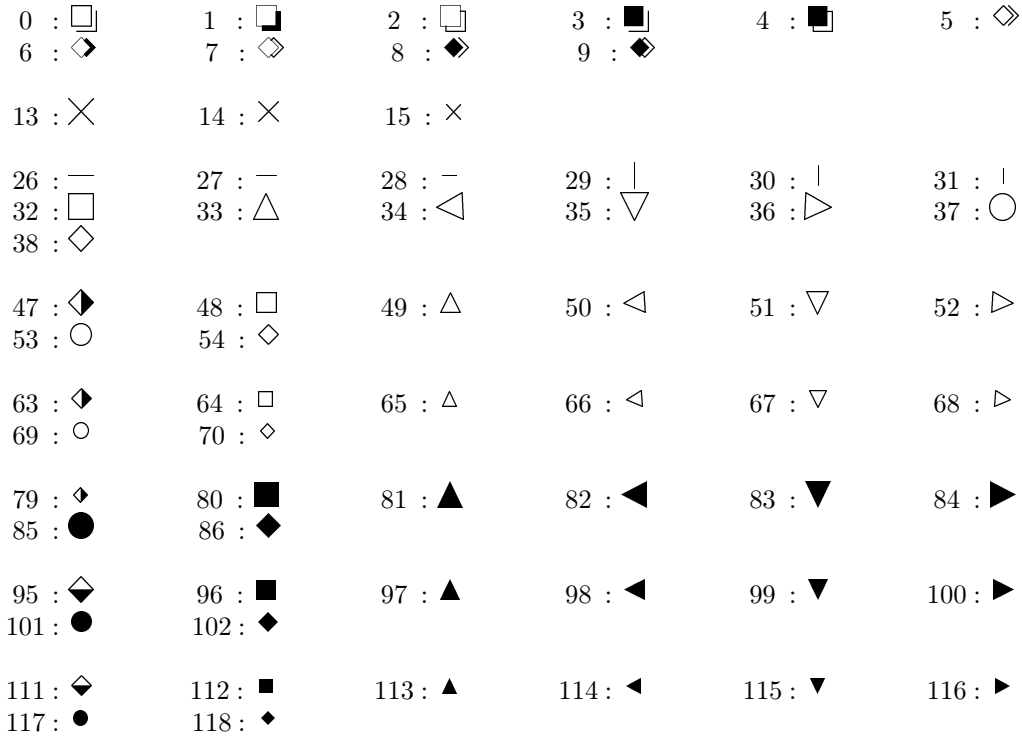
76 : 77 :

100 :

108 : 109 :

124 :

15.2 The Symbols of Symfam `ifsymgeo`



15.3 The Symbols of Symfam `ifsymgeonarrow`

0 :
1 :
2 :
3 :
4 : 
5 : 
6 :
7 :
8 :
9 :

13 :
14 : 
15 :

26 :
27 : 
28 :
29 :
30 : 
31 : 
32 :
33 : 
34 :
35 :
36 :
37 : 
38 :

47 :
48 : 
49 :
50 :
51 :
52 : 
53 :
54 : 

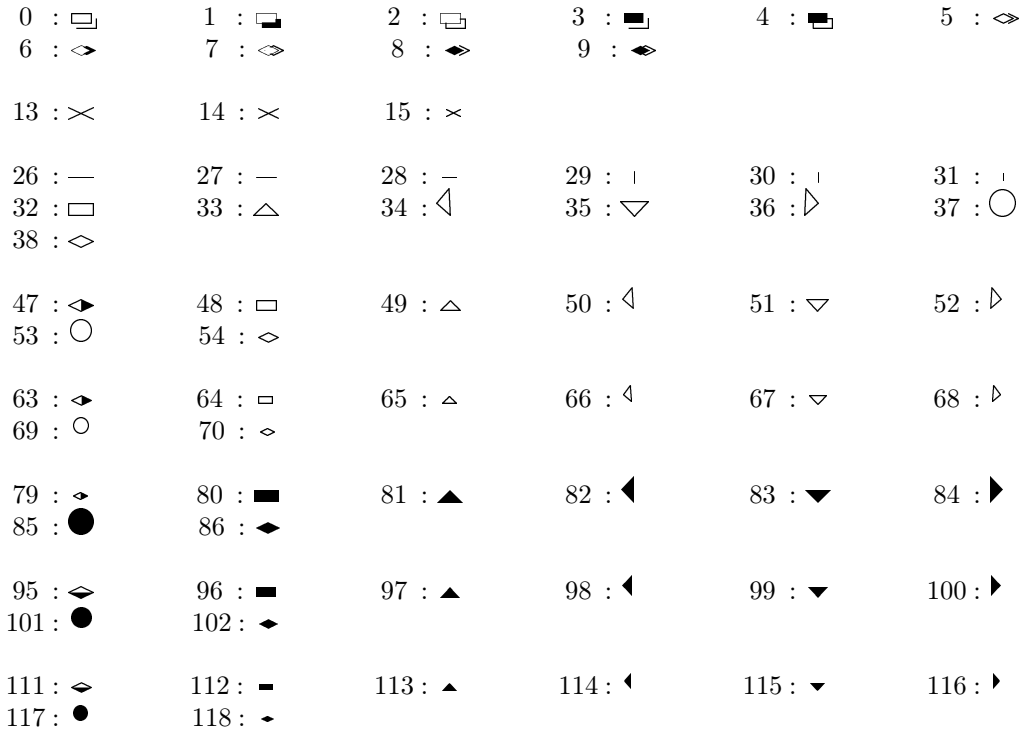
63 :
64 : 
65 :
66 :
67 :
68 : 
69 :
70 :

79 :
80 : 
81 :
82 :
83 :
84 : 
85 :
86 : 

95 :
96 : 
97 :
98 :
99 :
100 : 
101 :
102 :

111 :
112 :
113 :
114 :
115 :
116 : 
117 : 
118 :

15.4 The Symbols of Symfam `ifsymgeowide`



15.5 The Symbols of Symfam `ifsymweather`

0 :
1 :
2 :
3 :
4 :
5 :
6 :
7 :
8 :
9 :
10 :
11 :

16 :
17 :
18 :
19 :
20 :
21 :
22 :
23 :
24 :
25 :
26 :
27 :
28 :
29 :
30 :
31 :
32 :
33 :
34 :
35 :
36 :

48 :
49 :
50 :
51 :
52 :
53 :
54 :
55 :
56 :
57 :
58 :

15.6 The Symbols of Symfam `ifsymclock`

0 :
1 : 
2 :
3 : 
4 :
5 :
6 :
7 : 
8 :
9 : 
10 :
11 :
12 :
13 : 
14 : 
15 : 
16 :
17 :
18 :
19 : 
20 : 
21 :
22 :
23 :
24 :
25 : 
26 : 
27 : 
28 :
29 :
30 :
31 : 
32 : 
33 :
34 :
35 :
36 :
37 : 
38 : 
39 : 
40 :
41 :
42 :
43 :
44 :
45 :
46 :
47 :
48 :
49 :
50 :
51 :
52 :
53 :
54 :
55 :
56 :
57 :
58 :
59 :
60 :
61 :
62 :
63 :
64 :
65 :
66 :
67 :
68 :
69 :
70 :
71 :
72 :
73 :
74 :
75 :
76 :
77 :
78 :
79 :
80 :
81 :
82 :
83 :
84 :
85 :
86 :
87 :
88 :
89 :
90 :
91 :
92 :
93 :
94 :
95 :
96 :
97 :
98 :
99 :
100 :
101 :
102 :
103 :
104 :
105 :
106 :
107 :
108 :
109 :
110 :
111 :
112 :
113 :
114 :
115 :
116 :
117 :
118 :
119 :
120 :
121 :
122 :
123 :
124 :
125 :
126 :
127 :
128 :
129 :
130 :
131 :
132 :
133 :
134 :
135 :
136 :
137 :
138 :
139 :
140 :
141 :
142 :
143 :

148 :
149 :
150 :
151 :
152 :
153 :
154 :
155 :

16 From the package marvosym

16.1 The Symbols of Symfam marvosym

0 : □	1 : □	2 : □	3 : □	4 : □	5 : □
6 : □	7 : □	8 : □	9 : □	10 : □	11 : □
12 : □	13 : □	14 : □	15 : □	16 : □	17 : □
18 : □	19 : □	20 : □	21 : □	22 : □	23 : □
24 : □	25 : □	26 : □	27 : □	28 : □	29 : □
30 : □	31 : □	32 :	33 :	34 :	35 :
36 :	37 :	38 :	39 :	40 : (	41 :)
42 : ×	43 : +	44 : ,	45 : -	46 : .	47 : /
48 : 0	49 : 1	50 : 2	51 : 3	52 : 4	53 : 5
54 : 6	55 : 7	56 : 8	57 : 9	58 : →	59 : ⇒
60 : ≤	61 : ≙	62 : ≥	63 : ⇔	64 : @	65 :
66 :	67 :	68 : €	69 :	70 :	71 :
72 :	73 :	74 :	75 :	76 :	77 :
78 : □	79 : □	80 :	81 : ✂	82 : ---	83 :
84 : ☎	85 :	86 :	87 :	88 :	89 :
90 :	91 :	92 :	93 :	94 :	95 :
96 :	97 :	98 :	99 : €	100 : €	101 : €
102 :	103 :	104 :	105 :	106 :	107 :
108 : ↓	109 :	110 :	111 :	112 :	113 :
114 : ---	115 :	116 :	117 : FAX	118 :	119 :
120 :	121 :	122 :	123 : ○	124 :	125 :
126 : ♀	127 :	128 :	129 :	130 : □	131 :
132 :	133 :	134 : †	135 :	136 : †	137 :
138 :	139 :	140 : ♡	141 : @	142 :	143 :
144 :	145 : ●	146 :	147 : ■	148 : ■	149 : ●
150 :	151 :	152 :	153 :	154 :	155 : ○
156 :	157 :	158 :	159 :	160 :	161 :
162 :	163 :	164 : €	165 :	166 : $	167 :
168 :	169 :	170 :	171 :	172 :	173 :
174 :	175 :	176 :	177 :	178 :	179 : □
180 :	181 :	182 : ◀	183 : ▶	184 :	185 :
186 : ▲	187 : ▼	188 :	189 :	190 :	191 :
192 :	193 :	194 :	195 :	196 :	197 :
198 :	199 :	200 :	201 :	202 :	203 : △
204 :	205 :	206 :	207 :	208 :	209 :
210 :	211 :	212 :	213 :	214 :	215 :
216 :	217 :	218 :	219 :	220 :	221 :
222 : □	223 : □	224 :	225 :	226 :	227 :
228 :	229 :	230 : Ω	231 :	232 :	233 :
234 :	235 :	236 : □	237 : □	238 : □	239 : □
240 : A	241 : p	242 : □	243 : □	244 : □	245 : □
246 : □	247 : ·	248 : □	249 : □	250 : □	251 : □
252 : □	253 :	254 :	255 :		

17 From the package fourier

17.1 The Symbols of Symfam `fourier`

65 : ⛔ 66 : ⚠

69 : €

76 : 💣 77 : ☹ 78 : 🍸

84 : ⚔ 85 : ⚔

88 : ✠ 89 : ๛ 90 : ๛ 91 : ๛ 92 : ๛ 93 : ✧

102 : ❧ 103 : ❧ 104 : ❧

106 : ❧

109 : ❦ 110 : ❦ 111 : ❦

116 : ☞ 117 : ☞

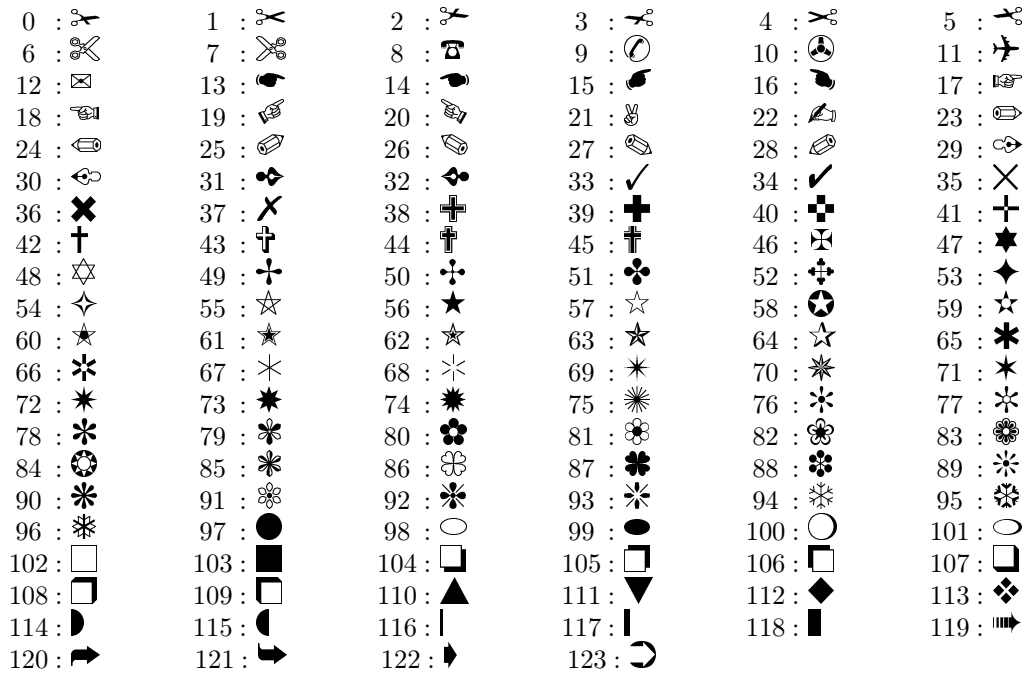
18 From the package wasysym

18.1 The Symbols of Symfam `wasysym`

0 : △	1 : ◁	2 : ⊴	3 : ▷	4 : ⊵	5 : ∴
6 : ⌕	7 : ☎	8 : ✓	9 : ➪	10 : ➧	11 : ♪
12 : ♩	13 : ♪	14 : 𝅗𝅥	15 : ♫	16 : ◀	17 : ▶
18 : ϟ	19 : ♌	20 : ♍	21 : ⍀	22 : ⊛	23 : ♈
24 : ⌐	25 : ♀	26 : ♂	27 : ¤	28 : ⏲	29 : ∝
30 : ⊰	31 : ⌀	32 : ●	33 : ⥁	34 : ⥀	35 : ○
36 : ☾	37 : ☽	38 : ♁	39 : ☿	40 : ‹	41 : ›
42 : ˆ	43 : ˇ	44 : ☺	45 : ☻	46 : ☼	47 : ☹
48 : ℧	49 : ⋈	50 : □	51 : ◇	52 : ⊠	53 : ¤
54 : ✠	55 : ⎔	56 : ◯	57 : ⬡	58 : ∼	59 : ⇝
60 : ⊏	61 : ⊐	62 : ≲	63 : ≳	64 : ≋	65 : ⚹
66 : ✳	67 : ✡	68 : ⬠	69 : ⋆	70 : ▽	71 : ◖
72 : ◗	73 : ◑	74 : ◗	75 : ▲	76 : ▼	
80 : ϑ	81 : ◜	82 : ◝			
85 : ə	86 : ♂	87 : ⚣	88 : ♄	89 : ♄	90 : ♁
91 : ☿	92 : ♇	93 : ☿	94 : ♊	95 : ♋	96 : ♍
97 : ♒	98 : ♏	99 : ♐	100 : ♉	101 : ♒	102 : ♓
103 : ¢	104 : ‰	105 : þ	106 : Þ	107 : ð	108 : ɔ
109 : ⚱	110 : ⍐	111 : ⍗	112 : ⍇	113 : ⍈	114 : ∫
115 : ∬	116 : ∭	117 : ∮	118 : ⨌	119 : ∫	120 : ∬
121 : ∭	122 : ∮	123 : ⨌	124 : ¦	125 : ⍞	126 : □
127 : ⍟					

19 From the package bbding

19.1 The Symbols of Symfam bbding



20 From the package dingbat

20.1 The Symbols of Symfam dingbat

66 :  67 :  68 : ✓

73 : 

78 :  79 : 

83 : 

90 : 

97 :  98 :  99 :  100 : 101 : 102 : 

103 : 104 : 

20.2 Large Symbols of `dingbat`

69: 

70: 

71: 

72: 

74: 

75: 

76: 

77: 

20.3 The Symbols of Symfam ark

67 :
68 :
69 :

76 :

80 :

82 :

85 :

87 :

100 :

117 :

Index