The **namespc** package*

Alexander Dreyer
`adreyer@web.de`

March 18, 2004

Abstract

The **namespc** package adds rudimentary *c++*-like namespace functionality to LaTeX. It may be used to declare local LaTeX commands, which can be made accessible in a later contexts without defining them globally.

1 Introduction

This package can be used to declare local LaTeX commands, which may be reused in a later context without polluting the global namespace, as an equivalent to *c++*-like namespaces.

2 Usage

`\namespace` This command is used to instantiate and reuse a certain namespace. The `\namespace` command has the following syntax:
`\namespace{`*spcname*`}{`*before*`}{`*body*`}`
The first parameter corresponds to the name of the namespace to be started or used. The *before* arguments of the current call of `\namespace` are appended to the preamble of namespace *spcname*, which is processed before the text in *body* gets processed.

spcname With the first call of `\namespace{`*spcname*`}{...}{...}` the corresponding environment *spcname* is defined which processes all *before* arguments within its preamble.

`\namespace*` Essentially the same as the `\namespace` command but without generation of the corresponding environment.

`\usingnamespace` By using `\usingnamespace{`*spcname*`}` one can force LaTeX to process the namespace preamble of *spcname*.

3 Implementation

We first make `@` to a character in order to use and define internal commands:

```
\makeatletter
```

`\@defnamespace@ifundef` First we define an internal macro which will be used to instatiate the (internal) namespace preamble macro `\@namespace@`*spcname*

```
\def\@defnamespace@ifundef#1{
```

---

*This document corresponds to **namespc** namespace, dated 2003/10/03.

```
  \@ifundefined{#1}{
    \long\expandafter\def\csname#1\endcsname{}
  }{}
}
```

`\@n@mesp@cestar` Now we are able to define the internal version of the starred `\namespace` command.

```
\newcommand{\@n@mesp@cestar}[3]{
  \@defnamespace@ifundef{@namespace@#1}
  \expandafter\g@addto@macro\csname @namespace@#1\endcsname{#2}%
  {
    \expandafter\relax\csname @namespace@#1\endcsname
    #3
  }
}
```

`\usingnamespace` The user command `\usingnamespace` just forces initiatiation and processing of the namespace preamble.

```
\newcommand{\usingnamespace}[1]{
  \@defnamespace@ifundef{@namespace@#1}
  \csname @namespace@#1\endcsname
}
```

`\@provide@namespace@env` This is an internal command which provides the corresponding environment.

```
\def\@provide@namespace@env#1{
  \@ifundefined{#1} {
   \newenvironment{#1}{\usingnamespace{#1}}{}
  }{}
}
```

`\::` The double colon notation may be used to access commands defined within a namespace preamble once. Note that the command's arguments have to be surrounded by colons:

```
\def\::#1::#2:#3:{%
  {\csname @namespace@#1\endcsname
   \csname#2\endcsname#3}
}
```

`\@n@mesp@ce` Next we just add the definition of the respective environment to `\namespace*` in order to obtain the internal version of `\namespace`:

```
\newcommand{\@n@mesp@ce}[3]{
  \@provide@namespace@env{#1}
  \@n@mesp@cestar{#1}{#2}{#3}
}
```

`\namespace` Finally we use the `\@ifstar` macro to define the user commands `\namespace` and `\namespace*`:

```
\newcommand*{\namespace}{\@ifstar\@n@mesp@cestar\@n@mesp@ce}
```

`@` is made special character again:

```
\makeatother
```

4 Example

This is example code for the use of the namespc package:

```
\namespace{spcname}
{\newcommand{\spcwidecmd}{\emph{command}\xspace}}{
 Using \spcwidecmd for the first time.
} % end of namespace

 % not defined here: \spcwidecmd

\namespace{spcname}{}{
  \spcwidecmd may be used again later.
} % end of namespace

\begin{spcname}
  We can also use \spcwidecmd within the corresponding environment\ldots
\end{spcname}

  Alternatively we may use the double colon notation: \::spcname::spcwidecmd::

\usingnamespace{spcname}
\ldots{} or globally: \spcwidecmd.
```

And here's the corresponding LaTeX output:

Using *command* for the first time.
command may be used again later.
We can also use *command* within the corresponding environment…
Alternatively we may use the double colon notation: *command*
… or globally: *command*

Index

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