

DoX – Doc, only eXtended*

Didier Verna

<mailto:didier@lrde.epita.fr>

<http://www.lrde.epita.fr/~didier/>

v2.0 (2009/09/21)

Abstract

The `doc` package provides L^AT_EX developers with means to describe the usage and the definition of new commands and environments. However, there is no simple way to extend this functionality to other items (options or counters for instance). DoX is designed to circumvent this limitation, and provides some improvements over the existing functionality as well.

The DoX package is Copyright © 2009 Didier Verna, and distributed under the terms of the LPPL license.

Contents

1	Installation	2
1.1	Extraction	2
1.2	TDS-compliant layout	2
1.3	AUC-T _E X support	2
2	Usage	2
2.1	Initialization	2
2.1.1	Requirements	2
2.1.2	Loading the package	3
2.2	Creating new documentation items	3
2.2.1	Example	3
2.2.2	Details	3
2.2.3	Options to <code>\doxitem</code>	4
2.3	Improvements over <code>doc</code> 's original API	4
2.3.1	Additional (optional) argument	4
2.3.2	Available options	4
2.3.3	Global effect	5
3	Conclusion	5
4	History	5

*DoX homepage: <http://www.lrde.epita.fr/~didier/software/latex.php#dox>

5	Implementation	5
5.1	Preamble	5
5.2	DoX options	5
5.3	DoX environments	6
5.4	DoX descriptions	7
5.5	API construction	7
5.6	Doc overrides	8
5.6.1	Macro facilities	8
5.6.2	Environment facilities	8
5.7	API creation	9
5.8	Finale	9

1 Installation

1.1 Extraction

If you are building DoX from the tarball you need to execute the following steps in order to extract the necessary files:

```
[pdf]latex dox.ins
[pdf]latex dox.dtx
[pdf]latex dox.dtx
```

After that, you need to install the generated documentation and style files to a location where L^AT_EX can find them.

1.2 TDS-compliant layout

For a TDS-compliant layout, the following locations are suggested:

```
[TEXMF]/tex/latex/dox/dox.sty
[TEXMF]/doc/latex/dox/dox.[pdf|dvi]
```

1.3 AUC-T_EX support

AUC-T_EX is a powerful major mode for editing T_EX documents in [X]Emacs. In particular, it provides automatic completion of command names once they are known. DoX supports AUC-T_EX by providing a style file named `dox.el` which contains AUC-T_EX definitions for the relevant commands. This file should be installed in a place where AUC-T_EX can find it. Please refer to the AUC-T_EX documentation for more information on this.

2 Usage

2.1 Initialization

2.1.1 Requirements

In order to work properly, DoX requires the presence of some L^AT_EX packages. You don't have to load them explicitly though. As long as L^AT_EX can locate them, they will be used automatically. DoX currently depends on `kvoptions`.

2.1.2 Loading the package

In order to load DoX, simply say `\usepackage[options]{dox}` in the preamble of your document. The package options will be discussed when appropriate.

2.2 Creating new documentation items

Note: we assume that you know about doc's `\DescribeMacro`, `\DescribeEnv` and all other associated commands and environments.

`\doxitem` [*options*]{*name*}{*envname*}{*idxcat*}

DoX provides a command named `\doxitem` to create new documentation items with functionalities equivalent to what `doc` provides for commands and environments. A whole API is created for every new item.

2.2.1 Example

Perhaps the simplest way to describe what it does is to give an example. Suppose you would like to describe package options. Here is what you need to do:

```
\usepackage{dox}
\doxitem{Option}{option}{options}
```

DoX then creates the following API for you:

- `\DescribeOption`
- the `option` environment
- `\PrintDescribeOption`
- `\PrintOptionName`
- `\SpecialMainOptionIndex`
- `\SpecialOptionIndex`

In order to comply with `doc`'s original behavior, the commands `\PrintDescribeOption` and `\PrintOptionName` will only be defined if they do not already exist.

2.2.2 Details

Here is a more precise description of the arguments to `\doxitem`.

- *name* (`Option` in our example) is used to construct the names of the commands in the new API. It usually starts with an upcase letter.
- *envname* (`option` in our example) is the name of the created environment. Be sure to avoid name clashes here! If you start experimenting odd behavior, you've probably overridden an existing command with your new environment.¹

¹It is a pity that L^AT_EX use the same namespace for commands and environments. The opening command for environment `env` should be named `\beginenv` and not just `\env`

- $\langle idxcat \rangle$ (**options** in our example) is the index category under which your items will appear. For example, all indexed options will be listed in the “options:” index entry.

2.2.3 Options to `\doxitem`

The first (optional) argument to `\doxitem` may contain a comma-separated list of options. Currently, there is only one supported option: **idxtype**.

We saw earlier that individual items appear under the **idxcat** index entry, but items also appear as standalone index entries, alphabetically sorted with their type in parenthesis. For instance, the **final** option would appear like this under the **F** index entry: “**final** (option)”.

By default, the index type is the same as the environment’s name (the $\langle envname \rangle$ argument). However, you can change this by providing a value to the **idxtype** option.

For instance, you may find the word “environment” too long for an index type. In that case, you may redefine the **environment** API like this:

```
\doxitem[idxtype=env.]{Env}{environment}{environments}
```

2.3 Improvements over `doc`’s original api

Please note that the improvements described in this section are also available in `doc`’s original command and environment API’s, because DoX redefines them.

2.3.1 Additional (optional) argument

Compared to `doc`, the API’s created by `\doxitem` are extended with a first optional argument containing a comma-separated list of options, such that, continuing with our initial example, the real prototypes are in fact the following:

```
\DescribeOption[opt,...]{name}

\begin{option}[opt,...]{name}
%% ...
\end{option}
```

2.3.2 Available options

The options currently supported are:

noprint Avoid printing $\langle name \rangle$ in the margin

noindex Avoid indexing $\langle name \rangle$

These options are useful if you don’t want printing or indexing locally for one particular item. Without them, one would need to locally `\let` the relevant commands to `\@gobble` which is very inconvenient.

There is also another advantage in using the **noprint** option: in `doc`’s original implementation, a margin paragraph will still be created just to be empty, hence

wasting float resources. If you're referencing a lot of items close to each other, this may lead to a "Too many unprocessed floats" error. With DoX, the `\marginpar` is avoided altogether.

2.3.3 Global effect

All options described in the previous section are also available to `\usepackage`. Their effect then becomes global. Because these options are boolean, it is still possible to counteract their global effect locally. For instance, one could do:

```
\usepackage[noprint]{dox}
```

and then later:

```
\DescribeOption[noprint=false]{french}
```

3 Conclusion

If you want to see DoX in action, take a look at the documentation of the `FiXme` package (version 4.0 or later). In fact, I wrote DoX for it in the first place.

4 History

- v2.0 Optional argument to `\doxitem` (`idxtype` option to change the item's index type).
Optional argument to `\Describe⟨Item⟩` and the `⟨item⟩` environment (`noprint` to avoid marginal printing and `noindex` to avoid indexing).
Extend `\DescribeMacro`, `\DescribeEnv` and their corresponding environments with the same features.

v1.0 First public version

5 Implementation

5.1 Preamble

```
1 <dox>\NeedsTeXFormat{LaTeX2e}
2 <*header>
3 \ProvidesPackage{dox}[2009/09/21 v2.0 Extensions to the doc package]
4
5 </header>
6 <*dox>
7 \RequirePackage{kvoptions}
8 \SetupKeyvalOptions{family=dox,prefix=dox@}
9
```

5.2 DoX options

These two options are available for use in `\usepackage` or in the generated item API's:

```

10 \DeclareBoolOption{noprint}
11 \DeclareBoolOption{noindex}
This one is for \doxitem only:
12 \DeclareStringOption{idxtyp}
13

```

5.3 DoX environments

`\@@doxenv` $\{\langle item \rangle\}\{\langle name \rangle\}$

In `doc.sty`, the macro and environment environments go through the `\m@cro@` macro which implements specific parts by testing a boolean condition as its first argument. This mechanism is not extensible, so I have to hack away a more generic version that would work for any new dox item, only which looks pretty much like the original one.

```

14 \long\def\@@doxenv#1#2{%
15   \endgroup%
16   \topsep\MacroTopsep%
17   \trivlist%
18   \edef\saved@macroname{\string#2}%
19   \def\makelabel##1{\llap{##1}}%
20   \if@inlabel%
21     \let\@tempa\@empty%
22     \count@\macro@cnt%
23     \loop\ifnum\count@>\z@%
24       \edef\@tempa{\@tempa\hbox{\strut}}\advance\count@\m@ne%
25     \repeat%
26     \edef\makelabel##1{\llap{\vtop to\baselineskip{\@tempa\hbox{##1}\vss}}}%
27     \advance\macro@cnt\@ne%
28   \else%
29     \macro@cnt\@ne%
30   \fi%
31   \ifdox@noprint%
32     \item%
33   \else%
34     \edef\@tempa{%
35       \noexpand\item[%

```

Apart from dependency on options, the first modification to the original macro involves dynamically constructing the name of the print macro:

```

36     \expandafter\noexpand\csname Print#1Name\endcsname{\saved@macroname}}}%
37     \@tempa%
38   \fi%
39   \ifdox@noindex\else%
40     \global\advance\c@CodelineNo\@ne%

```

and the second one involves dynamically constructing the name of the index macro:

```

41     \@nameuse{SpecialMain#1Index}-#2\nobreak%
42     \global\advance\c@CodelineNo\m@ne%
43   \fi%
44   \ignorespaces}

```

`\@doxenv` $\{\langle item \rangle\}[\langle options \rangle]$

Handle optional arguments and call `\@@doxenv`. Because environments can be

nested, we can't rely on grouping for getting options default values. Hence, we need to reset the options at every call.

```

45 \def\@doxenv#1[#2]{%
46   \@nameuse{dox@noprint\dox@noprintdefault}%
47   \@nameuse{dox@noindex\dox@noindexdefault}%
48   \setkeys{dox}{#2}%
49   \begingroup%
50     \catcode'\12%
51     \MakePrivateLetters%
52     \@doxenv{#1}}
53

```

5.4 DoX descriptions

`\@@doxdescribe` $\{\langle item \rangle\}\{\langle name \rangle\}$

The first closed group was the one opened to parse the $\langle name \rangle$ argument. The second one was opened to handle local options.

```

54 \def\@@doxdescribe#1#2{%
55   \endgroup%
56   \ifdox@noprint\else%
57     \marginpar{\raggedleft\@nameuse{PrintDescribe#1}{#2}}%
58   \fi%
59   \ifdox@noindex\else%
60     \@nameuse{Special#1Index}{#2}%
61   \fi%
62   \endgroup%
63   \@esphack\ignorespaces}

```

`\@doxdescribe` $\{\langle item \rangle\}[\langle options \rangle]$

Handle optional arguments and call `\@@doxdescribe`.

```

64 \def\@doxdescribe#1[#2]{%
65   \leavevmode\@bsphack%
66   \begingroup%
67     \setkeys{dox}{#2}%
68     \begingroup%
69       \MakePrivateLetters%
70       \@doxdescribe{#1}}
71

```

5.5 API construction

`\@doxcreatespecialmainindex` $\{\langle item \rangle\}\{\langle idxtype \rangle\}\{\langle idxcat \rangle\}$

```

72 \def\@doxcreatespecialmainindex#1#2#3{%
73   \expandafter\def\csname SpecialMain#1Index\endcsname##1{%
74     \@bsphack%
75     \special@index{%
76       ##1\actualchar{\string\ttfamily\space##1} (#2)\encapchar main}%
77     \special@index{%
78       #3:\levelchar##1\actualchar%
79       {\string\ttfamily\space##1}\encapchar main}%
80     \@esphack}}
81

```

```

\@doxcreatespecialindex  {\langle item \rangle}{\langle idxtype \rangle}{\langle idxcat \rangle}
82 \def\@doxcreatespecialindex#1#2#3{%
83   \expandafter\def\csname Special#1Index\endcsname##1{%
84     \@bsphack%
85     \index{##1\actualchar{\protect\ttfamily##1}
86       (#2)\encapchar usage}%
87     \index{#3:\levelchar##1\actualchar{\protect\ttfamily##1}\encapchar
88       usage}%
89     \@esphack}}
90

\@doxcreatedescribe  {\langle item \rangle}
91 \def\@doxcreatedescribe#1{%
92   \expandafter\def\csname Describe#1\endcsname{%
93     \@ifnextchar[%
94       {\@doxdescribe{#1}}{\@doxdescribe{#1} []}}
95

\@doxcreateenv  {\langle item \rangle}{\langle envname \rangle}
96 \def\@doxcreateenv#1#2{%
97   \expandafter\def\csname #2\endcsname{%
98     \@ifnextchar[%
99       {\@doxenv{#1}}{\@doxenv{#1} []}}
100   \expandafter\let\csname end#2\endcsname\endtrivlist}
101

```

5.6 Doc overrides

5.6.1 Macro facilities

Making `\DescribeMacro` work the DoX way is straightforward. The only precaution we need is to provide an alias to `\SpecialUsageIndex` because it should really be named `\SpecialMacroIndex`.

```

102 \let\SpecialMacroIndex\SpecialUsageIndex
103 \@doxcreatedescribe{Macro}
104

```

Making the macro environment work the DoX way is straightforward. The only precaution we need is to provide a `\SpecialMainMacroIndex` macro that does the job originally done in doc's `\macro@`.

```

105 \def\SpecialMainMacroIndex#1{%
106   \SpecialMainIndex{#1}\nobreak%
107   \DoNotIndex{#1}}
108 \@doxcreateenv{Macro}{macro}
109

```

5.6.2 Environment facilities

Making `\DescribeEnv` and the environment environment work the DoX way is straightforward.

```

110 \@doxcreatedescribe{Env}
111 \@doxcreateenv{Env}{environment}
112

```


5.7 API creation

The whole user interface is created in one macro call.

```

\doxitem  [⟨options⟩]{⟨name⟩}{⟨envname⟩}{⟨idxcat⟩}
113 \newcommand\doxitem[4][]{%
114   \def\dox@idxtype{#3}%
115   \setkeys{dox}{#1}

\Print...Name  {⟨name⟩}
116   \@ifundefined{Print#2Name}{%
117     \expandafter\def\csname Print#2Name\endcsname##1{%
118       \strut\MacroFont##1\ }{}

\SpecialMain...Index  {⟨name⟩}
119   \def\@doxexpr{\@doxcreatespecialmainindex{#2}}%
120   \expandafter\@doxexpr\expandafter{\dox@idxtype}{#4}%

\PrintDescribe...  {⟨name⟩}
121   \@ifundefined{PrintDescribe#2}{%
122     \expandafter\def\csname PrintDescribe#2\endcsname##1{%
123       \strut\MacroFont##1\ }{}

\Special...Index  {⟨name⟩}
124   \def\@doxexpr{\@doxcreatespecialindex{#2}}%
125   \expandafter\@doxexpr\expandafter{\dox@idxtype}{#4}%

\Describe...  [⟨options⟩]{⟨name⟩}
126   \@doxcreatedescribe{#2}

item  [⟨options⟩]{⟨name⟩}
127   \@doxcreateenv{#2}{#3}
128

```

5.8 Finale

We need to save the default value for every option because DoX environments need to reset them at every call.

```

129 \ProcessKeyvalOptions*
130 \ifdox@noprint
131   \def\dox@noprintdefault{true}
132 \else
133   \def\dox@noprintdefault{false}
134 \fi
135 \ifdox@noindex
136   \def\dox@noindexdefault{true}
137 \else
138   \def\dox@noindexdefault{false}
139 \fi
140
141 </dox>

```