The GETOPTK package

Michael Le Barbier Grünewald

6th of June 2011

Abstract

The *getoptk* package eases the definition of macros accepting optional arguments in the same style as `\hrule` or `\hbox`. It is meant to be used with *plain TEX*.

1 Introduction

A flexible way to pass optional arguments to a procedure is to rely on *dictionaries* of optional arguments, that is, a set of bindings between formal names of arguments and their values. Some TEX primitives, like `\hrule` or `\hbox`, use such an interface style. We call this style the *keyword* interface style. There is no facility in TEX to define new macros using the *keyword* interface style. The *getoptk* package provides such a service.

2 Quick guide

In order to define a macro using the *keyword* interface style, we have to setup first a *behaviour dictionary* binding *keywords* and *behaviours*. A keyword introduces an optional argument and a behaviour describes its effect, we will soon show an example of this. In the definition of the macro itself, we first select the behaviour dictionary we want to use and call `\getoptk`, the control sequence responsible of the detection of optional arguments. In this call, we need to provide a *callback* as argument to `\getoptk`, this callback is a macro taking control of the execution after `\getoptk` has completed its task. It will be called with an argument, that is derived from the list of optional arguments.

For explanatory purposes, let us assume that we want to define a macro `\begindisplay` using the *keyword* interface style and accepting the following optional arguments:

ragged Fill, but do not adjust the right margin (only left-justify).

literal Display block with literal font (usually fixed-width). Useful for source code or simple tabbed or spaced text.

file ⟨***file name***⟩ The file whose name, enclosed in curly braces, follows the *file* keyword is read and displayed using the selected display type.

offset ⟨***dimen***⟩ Use *dimen* as indentation for the display.

We first create a fresh new behaviour dictionary:

```
\newgetoptkdictionary{display}
```

and fill it with behaviours:

```
\defgetoptkflag{ragged}{\raggedright}
\defgetoptkflag{literal}{\let\displayfont\literalfont}
\defgetoptktoks{file}{\input #1}
\defgetoptkdimen{dimen}{\displayindent=#1\relax}
```

Besides registering the behaviours in the dictionary *display*, these commands also bind the behaviours to the following control sequences:

```
\getoptk@behaviour@display@ragged
\getoptk@behaviour@display@literal
\getoptk@behaviour@display@file
\getoptk@behaviour@display@dimen
```

The control sequences created with `\getoptkflag` must do not have an argument, while those created by `\getoptktoks` or `\getoptkdimen` do have one. The definition of `\begindisplay` is

```
\def\begindisplay{%
  \setgetoptkdictionary{display}%
  \getoptk\display@M
}
```

The control sequence `\getoptk` is such that the input text

```
\begindisplay file {chapter1} literal offset 20pt
```

is *replaced* by

```
\display@M{%
  \getoptk@behaviour@display@file{chapter1}%
  \getoptk@behaviour@display@literal
  \getoptk@behaviour@display@dimen{20pt}%
}
```

so that `\display@M` can do its job and trigger the behaviours at the appropriate time.

3 Defining a behaviour dictionary

The command `\newgetoptkdictionary`⟨*dictionary*⟩ creates a behaviour dictionary that will be filled by subsequent calls to commands binding keywords and behaviours. The bindings commands are:

```
\defgetoptkflag
\defgetoptkcount
\defgetoptkdimen
\defgetoptkskip
\defgetoptktoks
\defgetoptkbracket
```

They all must be called like in the previous examples, following the pattern:

⟨*binding_command*⟩⟨*keyword*⟩⟨*behaviour*⟩

defgetoptkflag The ⟨*behaviour*⟩ is the replacement text of a macro without arguments. When it finds ⟨*keyword*⟩, the `\getoptk` macro does not look for an argument but starts again scanning for keywords.

defgetoptkcount The ⟨*behaviour*⟩ is the replacement text of a macro having one argument. When it finds ⟨*keyword*⟩, the `\getoptk` macro scans further for an argument that is valid right hand side value for a *count* register. This argument will be supplied to the ⟨*behaviour*⟩ when behaviours are triggered.

defgetoptkdimen The ⟨*behaviour*⟩ is the replacement text of a macro having one argument. When it finds ⟨*keyword*⟩, the `\getoptk` macro scans further for an argument that is valid right hand side value for a *dimen* register. This argument will be supplied to the ⟨*behaviour*⟩ when behaviours are triggered.

defgetoptkskip The ⟨*behaviour*⟩ is the replacement text of a macro having one argument. When it finds ⟨*keyword*⟩, the `\getoptk` macro scans further for an argument that is valid right hand side value for a *skip* register. This argument will be supplied to the ⟨*behaviour*⟩ when behaviours are triggered.

defgetoptktoks The ⟨*behaviour*⟩ is the replacement text of a macro having one argument. When it finds ⟨*keyword*⟩, the `\getoptk` macro scans further for an argument that is valid right hand side value for a *toks* register. This argument will be supplied to the ⟨*behaviour*⟩ when behaviours are triggered.

defgetoptkbracket The ⟨*behaviour*⟩ is the replacement text of a macro having one argument. When it finds ⟨*keyword*⟩, the `\getoptk` macro scans further for an optional argument enclosed by square brackets. If such an argument is found, it is supplied to ⟨*behaviour*⟩ when behaviours are triggered and the predicate `\ifgetoptkbracket` is bound to `\iftrue`. If no such an argument is found, then the empty argument is supplied to ⟨*behaviour*⟩ when behaviours are triggered and the predicate `\ifgetoptkbracket` is bound to `\iffalse`.

4 Licence