L'extension pour TEX/LATEX

simplekv

v 0.2

27 avril 2020

Christian Tellechea
unbonpetit@netc.fr

Cette petite extension est une implémentation d'un système dit à « ⟨*clés*⟩/⟨*valeurs*⟩ » pour TEX ou LATEX. Elle comporte juste l'essentiel, aucune fioriture inutile n'a été codée et aucune extension tierce n'est nécessaire à son fonctionnement.

1 Clés, valeurs

Lorsqu'une macro doit recevoir des paramètres dont le nombre n'est pas fixe ou connu, il est commode de procéder par ⟨*clés*⟩ et ⟨*valeurs*⟩. Voici brièvement les définitions et les limitations des structures mises à disposition :

- une ⟨*clé*⟩ est un mot désignant un paramètre ; il est formé de préférence avec des caractères de code de catégorie 11 (lettres), 12 (autres caractères sauf la virgule et le signe =) et 10 (l'espace). On peut cependant y mettre des caractères ayant d'autres codes de catégorie, dans la limitation de ce qui est admis dans la primitive `\detokenize` ; une ⟨*clé*⟩, même si cela revêt peu de signification, peut être vide ;
- la syntaxe pour assigner une ⟨*valeur*⟩ à une ⟨*clé*⟩ est : ⟨*clé*⟩=⟨*valeur*⟩ ;
- les espaces qui précèdent et qui suivent la ⟨*clé*⟩ et la ⟨*valeur*⟩ sont ignorés, mais *pas ceux* qui se trouvent à l'intérieur de la ⟨*clé*⟩ ou de la ⟨*valeur*⟩ ;
- une ⟨*valeur*⟩ est un ⟨*code*⟩ arbitraire ;
- si une ⟨*valeur*⟩ est entourée d'accolades, ces dernières seront retirées : ⟨*clé*⟩=⟨*valeur*⟩ est donc équivalent à ⟨*clé*⟩={⟨*valeur*⟩} ;
- lorsqu'une valeur est entourée de *plusieurs* imbrications d'accolades, seul le niveau externe est retiré et donc ⟨*clé*⟩={{⟨*valeur*⟩}} est compris comme ⟨*clé*⟩={⟨*valeur*⟩} ;
- lorsque plusieurs couples de ⟨*clés*⟩/⟨*valeurs*⟩ doivent être spécifiés, ils sont séparés les uns des autres par des virgules ;
- une virgule ne peut figurer dans une ⟨*valeur*⟩ que si la virgule est dans un niveau d'accolades ; par exemple, `foo=1,5` n'est pas valide car la ⟨*valeur*⟩ s'étend jusqu'au 1. Il faudrait écrire `foo={1,5}` pour spécifier une valeur de `1,5` ;
- les ⟨*valeurs*⟩ sont stockées *telles qu'elles sont lues* ; en particulier, aucun développement n'est effectué ;
- les définitions sont *locales* : par conséquent, toute ⟨*clé*⟩ définie ou modifiée dans un groupe est restaurée à son état antérieur à la sortie du groupe ;
- des ⟨*clé*⟩/⟨*valeurs*⟩ destinées à une même macro ou à un même usage doivent être regroupées dans un ensemble dont on choisit le nom. Un tel ensemble est appelé ⟨*trousseau*⟩.

2 Commandes mises à disposition

Les macro `\setKV` et `\setKVdefault` Ces commandes définissent des ⟨*clés*⟩ et leur assignent des ⟨*valeurs*⟩ dans un ⟨*trousseau*⟩. La seule différence entre les deux macros est que `\setKVdefault`, en plus d'assigner les ⟨*valeurs*⟩ aux ⟨*clés*⟩, les sauvegarde en vue d'une restauration ultérieure avec `\restoreKV`.
On écrit

```
\setKV[⟨trousseau⟩]{⟨clé 1⟩=⟨valeur 1⟩,⟨clé 2⟩=⟨valeur 2⟩,...,⟨clé n⟩=⟨valeur n⟩}
```

Il faut noter que

- l'argument entre accolades contenant les ⟨*clés*⟩ et les ⟨*valeurs*⟩ ne devrait pas être vide, sauf à vouloir définir une ⟨*clé*⟩ booléenne vide égale à `true` ;
- lors de la lecture des ⟨*clés*⟩/⟨*valeurs*⟩, la virgule et le signe égal ont leurs catcodes rendus égaux à 12 ;
- le nom du ⟨*trousseau*⟩, bien qu'entre crochet, est *obligatoire*, mais il peut être vide bien que cela ne soit pas conseillé ;
- si une même ⟨*clé*⟩ figure plusieurs fois, la ⟨*valeur*⟩ retenue sera celle de la dernière assignation ;
- les ⟨*valeurs*⟩ peuvent être booléennes auquel cas, elles *doivent* être « `true` » ou « `false` » en caractères de catcode 11 ;
- si une ⟨*valeur*⟩ est omise, elle est comprise comme étant « `true` ». Ainsi, écrire

  ```
  \setKV[foo]{mon bool}
  ```

  est équivalent à

  ```
  \setKV[foo]{mon bool = true}
  ```

La macro \useKV Cette macro purement développable renvoie la ⟨*valeur*⟩ préalablement associée à une ⟨*clé*⟩ dans un ⟨*trousseau*⟩ :

```
\useKV[⟨trousseau⟩]{⟨clé⟩}
```

Il faut noter que

- si la ⟨*clé*⟩ n'a pas été définie, une erreur sera émise ;
- si la ⟨*clé*⟩ est booléenne, le texte « `true` » ou « `false` » sera renvoyé ;
- il faut 2 développements à `\useKV[⟨trousseau⟩]{⟨clé⟩}` pour donner la ⟨*valeur*⟩ associée à la ⟨*clé*⟩.

```
\setKV[foo]{nombre = 5 , lettres= AB \textit{CD} , mon bool}
a) \useKV[foo]{nombre}.\qquad   b) \useKV[foo]{lettres}.\qquad   c) \useKV[foo]{mon bool}.

\setKV[foo]{lettres = X Y Z \textbf{123} }
a) \useKV[foo]{nombre}.\qquad   b) \useKV[foo]{lettres}.\qquad   c) \useKV[foo]{mon bool}.
```

a) 5. b) AB *CD*. c) true.
a) 5. b) X Y Z **123**. c) true.

La macro \restoreKV La macro `\restoreKV[⟨trousseau⟩]` réinitialise toutes les ⟨*clés*⟩ du ⟨*trousseau*⟩ aux ⟨*valeurs*⟩ qui ont été définies lors de l'exécution `\setKVdefault`. La macro `\useKVdefault[⟨trousseau⟩]` lui est équivalente.

La macro \ifboolKV Cette macro permet, selon la valeur d'une ⟨*clé booléenne*⟩, d'exécuter un des deux ⟨*codes*⟩ donnés. La syntaxe est

```
\ifboolKV[⟨trousseau⟩]{⟨clé⟩}{⟨code si "true"⟩}{⟨code si "false⟩}
```

La macro est purement développable, elle nécessite 2 développements pour donner l'un des deux codes, et exige que la ⟨*clé*⟩ soit booléenne sans quoi un message d'erreur est émis.

La macro \showKV Cette commande écrit dans le fichier `log` la ⟨*valeur*⟩ assignée à une ⟨*clé*⟩ d'un ⟨*trousseau*⟩ :

```
\showKV[⟨trousseau⟩]{⟨clé⟩}
```

Si la ⟨*clé*⟩ n'est pas définie, « `not defined` » est affiché dans le fichier log.

3 Code

En plus d'une ⟨*valeur*⟩, un ⟨*code*⟩ arbitraire peut être assigné à n'importe quelle ⟨*clé*⟩. Pour ce faire, on écrit

```
\defKV[⟨trousseau⟩]{⟨clé 1⟩=⟨code 1⟩,⟨clé 2⟩=⟨code 2⟩,...,⟨clé n⟩=⟨code n⟩}
```

Chaque ⟨*code*⟩ peut contenir `#1` qui représente la ⟨*valeur*⟩ de la ⟨*clé*⟩. Ce ⟨*code*⟩ est exécuté lorsque une ⟨*valeur*⟩ est assignée à la ⟨*clé*⟩ avec `\setKV`, `\setKVdefault` ou `\restoreKV`.

Ainsi déclarer

```
\defKV[x]{ mykey = \def\foo{\textbf{#1}}
```

va définir une macro `\foo` dès que la ⟨*clé*⟩ « `mykey` » va être définie (ou redéfinie) et donc, si l'on écrit

```
\setKV[x]{ mykey = bonjour }
```

le code qui est exécuté en coulisses est

```
\long\def\foo{\textbf{bonjour}}
```

```
\defKV[x]{ mykey = \def\foo{\textbf{#1}} }
\setKV[x]{ mykey = bonjour }% définition
1) \meaning\foo\par
2) \useKV[x]{ mykey }

\setKV[x]{ mykey = hello }% redéfinition
3) \meaning\foo\par
4) \useKV[x]{ mykey }
```

1) macro:->\textbf {bonjour}
2) bonjour
3) macro:->\textbf {hello}
4) hello

La macro `\testboolKV` permet de tester, par exemple dans un ⟨*code*⟩, si son argument est « true » ou « false »

```
\testboolKV{⟨argument⟩}{⟨code si true⟩}{⟨code si false⟩}
```

La macro est purement développable, elle nécessite 2 développements pour donner l'un des deux codes, et exige que l'⟨*argument*⟩ soit booléen sans quoi un message d'erreur est émis.

```
\defKV[x]{ x = \def\test{\testboolKV{#1}{test positif}{test négatif}}}
\setKV[x]{ x = true}
1) \test

\setKV[x]{ x= false}
2) \test
```

1) test positif
2) test négatif

Toute autre valeur que « `true` » ou « `false` » génèrera un message d'erreur.

4 Un exemple d'utilisation

Voici comment on pourrait programmer une macro qui affiche un cadre sur une ligne, grâce à la macro `\fbox` et l'environnement `center` de LaTeX. Pour cela les ⟨*clés*⟩ suivantes seront utilisées :

- le booléen `inline` qui affichera le cadre dans le texte s'il est vrai et sur une ligne dédié s'il est faux ;
- `sep` qui est une dimension mesurant la distance entre le texte et le cadre (par défaut `3pt`) ;
- `width` qui est la largeur des traits du cadre (par défaut `0.5pt`) ;
- `style` qui contient le code exécuté avant le texte.

Une première façon de faire, sans recours à `\defKV` ;

```
\setKVdefault[frame]{
  sep        = 3pt,
  line width = 0.5pt,
  style      = \bfseries,
  inline
  }
\newcommand\frametxt[2][]{%
  \restoreKV[frame]% revenir au valeurs par défaut
  \setKV[frame]{#1}% lit les arguments optionnels
  \fboxsep = \useKV[frame]{sep}
  \fboxrule= \useKV[frame]{line width}
  \ifboolKV[frame]{inline}
    {}
    {\begin{center}}%
  \fbox{\useKV[frame]{style}#2}%
  \ifboolKV[frame]{inline}
    {}
    {\end{center}}%
}
Un essai en ligne par défaut \frametxt{essai} puis un autre \frametxt[sep=5pt,line width=2pt]{essai}
et un dernier \frametxt[sep=1pt,style=\itshape]{essai}.

Un essai hors ligne : \frametxt[inline = false, style=\bfseries\color{red}]{essai centré}
```

Un essai en ligne par défaut **essai** puis un autre **essai** et un dernier *essai*.
Un essai hors ligne :

essai centré

Dans l'exemple repris ci-dessous et grâce à `\defKV`, on stocke tous les paramètres lors de leur assignation. Il y a bien moins de verbosité dans le code de `frametxt` ce qui le rend plus léger et plus lisible.

```
\defKV[frame]{%
  sep        = {\fboxsep = #1 },
```

```
  line width = {\fboxrule= #1 },
  inline     = \testboolKV{#1}
                    {\def\hookpre{}\def\hookpost{}}
                    {\def\hookpre{\begin{center}}\def\hookpost{\end{center}}},
  style      = \def\fstyle{#1}
}
\setKVdefault[frame]{
  sep        = 3pt,
  line width = 0.5pt,
  style      = \bfseries,
  inline
  }
\newcommand\frametxt[2][]{%
  \restoreKV[frame]% revenir au valeurs par défaut
  \setKV[frame]{#1}% lit les arguments optionnels
  \hookpre
  \fbox{\fstyle #2}%
  \hookpost
}
Un essai en ligne par défaut \frametxt{essai} puis un autre \frametxt[sep=5pt,line width=2pt]{essai}
et un dernier \frametxt[sep=1pt,style=\itshape]{essai}.

Un essai hors ligne : \frametxt[inline = false, style=\bfseries\color{red}]{essai centré}
```

Un essai en ligne par défaut **essai** puis un autre **essai** et un dernier *essai*.
Un essai hors ligne :

essai centré

5 Le code

Le code ci-dessous est l'exact verbatim du fichier `simplekv.tex` :

```
% !TeX encoding = ISO-8859-1
% Ce fichier contient le code commenté de l'extension "simplekv"
%
% IMPORTANT : pour que les commentaires s'affichent correctement,
%             ouvrir ce fichier avec l'encodage ISO-8859-1
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
\def\skvname                    {simplekv}                                      %
\def\skvver                       {0.2}                                         %
%                                                                               %
\def\skvdate                    {2020/04/27}                                    %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% ------------------------------------------------------------------------
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3
% of this license or (at your option) any later version.
% The latest version of this license is in
%
% %     http://www.latex-project.org/lppl.txt
%
% and version 1.3 or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
% ------------------------------------------------------------------------
% This work has the LPPL maintenance status ‘maintained’.
%
% The Current Maintainer of this work is Christian Tellechea
% email: unbonpetit@netc.fr
%        Commentaires, suggestions et signalement de bugs bienvenus !
```

```
%        Comments, bug reports and suggestions are welcome.
% Copyright: Christian Tellechea 2017-2020
% ----------------------------------------------------------------------
% L'extension simplekv est composée des 5 fichiers suivants :
%   - code               : simplekv           (.tex et .sty)
%   - manuel en français : simplekv-fr        (.tex et .pdf)
%   - fichier lisezmoi   : README
% ----------------------------------------------------------------------

%########################################
%################ Préalable ###############
%########################################
\csname skvloadonce\endcsname
\let\skvloadonce\endinput
\ifdefined\skvfromSTY\else
  \immediate\write -1 {%
    Package: \skvname\space\skvdate\space v\skvver\space Simple keyval package (CT)%
  }%
\fi
%########################################
%############ Gestion catcodes ############
%########################################
\begingroup
  \def\X#1{\catcode\number`#1=\number\catcode`#1\relax}
  \expandafter\xdef\csname skv_restorecatcode\endcsname{\X\,\X\=\X\_}
\endgroup
\catcode`\_11
\chardef\skv_other12
\catcode`\,\skv_other\catcode`\=\skv_other
%########################################
%############ Macros auxilaires ###########
%########################################
\chardef\skv_stop 0
\long\def\skv_first#1#2{#1}
\long\def\skv_second#1#2{#2}
\long\def\skv_gob#1{}
\long\def\skv_exe#1{#1}
\expandafter\def\expandafter\skv_gobspace\space{}% pour garder la compatibilité
\long\def\skv_earg#1#2{\expandafter\skv_earg_i\expandafter{#2}{#1}}\let\skv_exparg\skv_earg
\long\def\skv_eearg#1#2{\expandafter\expandafter\expandafter\skv_earg_i\expandafter\expandafter\expandafter
    {#2}{#1}}
\long\def\skv_earg_i#1#2{#2{#1}}
\long\def\skv_expafter#1#2{\expandafter\skv_expafter_i\expandafter{#2}{#1}}% {<a>}{<b>} devient <a><*b>
\long\def\skv_expafter_i#1#2{#2#1}
\def\skv_ifcsname#1{\ifcsname#1\endcsname\expandafter\skv_first\else\expandafter\skv_second\fi}
\long\def\skv_ifx#1{\ifx#1\expandafter\skv_first\else\expandafter\skv_second\fi}
\long\def\skv_ifempty#1{\skv_ifempty_i#1\_nil\_nil\skv_second\skv_first\__nil}%
\long\def\skv_ifempty_i#1#2\_nil#3#4#5\__nil{#4}
\def\skv_stripsp#1{%
\long\def\skv_stripsp##1##2{\expanded{\skv_stripsp_i\_marksp##2\__nil\_marksp#1\_marksp\_nil{##1}}}%
\long\def\skv_stripsp_i##1\_marksp#1##2\_marksp##3\_nil{\skv_stripsp_ii##3##1##2\__nil#1\__nil\_nil}%
\long\def\skv_stripsp_ii##1#1\__nil##2\_nil{\skv_stripsp_iii##1##2\_nil}%
\long\def\skv_stripsp_iii##1##2\__nil##3\_nil##4{\unexpanded{##4{##2}}}%
}\skv_stripsp{ }
%########################################
%########## Macros de définition ##########
%########################################
\def\setKVdefault{\let\skv_find_kv_i\skv_find_kv_nocode\skv_readKV\skv_exe}
\def\setKV       {\let\skv_find_kv_i\skv_find_kv_nocode\skv_readKV\skv_gob}
\def\defKV       {\let\skv_find_kv_i\skv_find_kv_code  \skv_readKV\skv_gob}
\def\skv_readKV{%
  \edef\skv_restorecatcode{\catcode44=\the\catcode44 \relax\catcode61=\the\catcode61 \relax}%
  \catcode44\skv_other\catcode61\skv_other
  \skv_readKV_i
}
\long\def\skv_readKV_i#1[#2]#3{%
```

```
  #1{\expandafter\def\csname skv_[#2]\endcsname{#3}}% exécute (si \defKV) ou pas
  \def\skv_setname{#2}%
  \skv_readKV_ii#3,\__,%
  \skv_restorecatcode
}
\long\def\skv_readKV_ii#1,{\skv_readKV_iii\skv_find_kv#1=true=\_nil\skv_find_kv\__\__nil}% si #1=\__ ne rien faire sinon \skv_find_kv#1=true=\_nil
\long\def\skv_readKV_iii#1\skv_find_kv\__#2\__nil{#1}

\long\def\skv_find_kv#1=#2=#3\_nil{%
  \edef\__key{_[\skv_setname]_\skv_stripsp\detokenize{#1}}%
  \skv_stripsp\skv_find_kv_i{#2}%
  \skv_readKV_ii
}
\long\def\skv_find_kv_nocode#1{%
  \expandafter\def\csname skv\__key\endcsname{#1}%\__val% stocker la clé
  \ifcsname skvcode\__key\endcsname% si le code correspondant existe
    \csname skvcode\__key\endcsname{#1}% exécute le code
  \fi
}
\long\def\skv_find_kv_code#1{%
  \expandafter\def\csname skvcode\__key\endcsname##1{#1}%
}

\def\restoreKV[#1]{%
  \skv_ifcsname{skv_[#1]}
    {\skv_eearg{\setKV[#1]}{\csname skv_[#1]\endcsname}}
    {\errmessage{Undefined or not saved set of keys "#1"}}%
}
\let\useKVdefault\restoreKV
%########################################
%############## Macro \useKV ##############
%########################################
\def\useKV[#1]#2{\expanded{\skv_stripsp{\useKV_i[#1]}{#2}}}
\def\useKV_i[#1]#2{\expandafter\useKV_ii\csname skv_[#1]_#2\endcsname{#2}}
\def\useKV_ii#1#2{%
  \ifdefined#1\unexpanded\expandafter{#1}%
  \else     \errmessage{Key "#2" not defined}%
  \fi
}
%########################################
%############## Macros de test #############
%########################################
\def\ifboolKV[#1]#2{\romannumeral\skv_stripsp{\ifboolKV_i[#1]}{#2}}
\def\ifboolKV_i[#1]#2{%
  \skv_ifempty{#2}
    {\skv_stop\errmessage{Empty argument is not a valid boolean}\skv_second
    }
    {\skv_ifcsname{skv_[#1]_#2}
      {\skv_eearg\ifboolKV_ii{\csname skv_[#1]_#2\endcsname}}
      {\skv_stop\errmessage{Key "#2" not defined}\skv_second}%
    }%
}
\def\ifboolKV_ii#1{%% Cette macro teste si #1, qui est une <valeur>, vaut "true" ou "false"
  \skv_ifargtrue{#1}
    {\expandafter\skv_stop\skv_first
    }
    {\skv_ifargfalse{#1}
      {\expandafter\skv_stop\skv_second}
      {\skv_stop\errmessage{Value "#1" is not a valid boolean}\skv_second}%
    }%
}

\def\testboolKV#1{\romannumeral\skv_stripsp{\testboolKV_i}{#1}}% macro publique qui teste si #1 est <true> ou <false>, erreur sinon
\def\testboolKV_i#1{%
```

```
  \skv_ifempty{#1}
    {\skv_stop\errmessage{Empty argument is not a valid boolean}\skv_second}
    {\skv_stripsp{\ifboolKV_ii}{#1}}%
}

\def\skv_ifargtrue#1{\skv_ifargtrue_i#1true\_nil}
\def\skv_ifargtrue_i#1true#2\_nil{\skv_ifempty{#1}{\skv_ifargtrue_ii#2\_nil}\skv_second}
\def\skv_ifargtrue_ii#1true#2\_nil{\skv_ifempty{#1#2}}
\def\skv_ifargfalse#1{\skv_ifargfalse_i#1false\_nil}
\def\skv_ifargfalse_i#1false#2\_nil{\skv_ifempty{#1}{\skv_ifargfalse_ii#2\_nil}\skv_second}
\def\skv_ifargfalse_ii#1false#2\_nil{\skv_ifempty{#1#2}}
%########################################
%############## Macro \showKV ##############
%########################################
\def\showKV[#1]#2{\expanded{\skv_stripsp{\showKV_i[#1]}{#2}}}
\def\showKV_i[#1]#2{%
  \immediate\write-1 {%
    ^^JKey\space\space[#1]#2=%
      \skv_ifcsname{skv_[#1]_#2}
        {\expandafter\expandafter\expandafter\skv_show\expandafter
        \meaning\csname skv_[#1]_#2\endcsname
        \skv_ifcsname{skvcode_[#1]_#2}
          {^^JCode [#1]#2=\expandafter\expandafter\expandafter\skv_show\expandafter
            \meaning\csname skvcode_[#1]_#2\endcsname
          }
          {}%
        }
        {not defined%
        }%
  ^^J\relax}%
}
\def\skv_show#1->{}
\skv_restorecatcode
\endinput

Versions :
 ____________________________________________________________________________
| Version |    Date    | Changements                                         |
|---------|------------|-----------------------------------------------------|
|   0.1   | 08/08/2017 | Première version                                    |
|---------|------------|-----------------------------------------------------|
|   0.2   | 27/04/2020 | - Un <code> peut être assigné à une <clé>           |
|         |            | - Correction de bugs                                |
|         |            | - Optimisations                                     |
|---------|------------|-----------------------------------------------------|
```