L'extension

listofitems

v1.1

01 septembre 2016

Christian Tellechea [*]
Steven B. Segletes [†]

Cette petite extension pour est destinée à lire une liste d'éléments dont le séparateur peut être choisi par l'utilisateur. Une fois la liste lue, ses éléments sont stockés dans une structure qui se comporte comme un tableau unidimensionnel et ainsi, il devient très facile d'accéder à un élément de la liste par son numéro. Par exemple, si la liste est stockée dans la macro `\foo`, l'élément n° 3 est désigné par `\foo[3]`.
Un élément peut à son tour être une liste disposant d'un autre séparateur que celui de la liste de niveau supérieur, ce qui ouvre la voie à des imbrications et donne une syntaxe rappelant celle des tableaux à plusieurs dimensions du type `\foo[3,2]` pour accéder à l'élément n° 2 de la liste contenue dans l'élément n° 3 de la liste de plus haut niveau.

*. unbonpetit@openmailbox.org
†. steven.b.segletes.civ@mail.mil

1 Avant-propos

Cette extension ne requiert aucun package, doit être utilisée avec un moteur ε-TEX, et doit être appelée sous (pdf)(Xe)(lua)LATEX par

```
\usepackage{listofitems}
```

et sous (pdf)(Xe)(lua)TEX par

```
\input listofitems.tex
```

2 Lire une liste simple

Définir le séparateur Le ⟨*séparateur*⟩ par défaut est la virgule et si l'on souhaite en changer, il faut, avant de lire une liste d'éléments, le définir avec `\setsepchar{`⟨*séparateur*⟩`}`. Un ⟨*séparateur*⟩ est un ensemble de tokens dont les catcodes sont différents de 1 et 2 (les accolades ouvrantes et fermantes), 14 (habituellement %) et 15. Le token de catcode 6 (habituellement #) n'est accepté que s'il est suivi d'un entier auquel cas l'ensemble désigne l'argument d'une macro ; en aucun cas ce token ne doit se trouver seul dans le ⟨*séparateur*⟩. Des commandes peuvent se trouver dans cet ensemble de tokens, y compris la primitive `\par`.

Le ⟨*séparateur*⟩ « `/` » est réservé par défaut pour les listes imbriquées (voir page 3). Il ne faut donc pas écrire « `\setsepchar{/}` » car listofitems comprendrait que l'on souhaite lire une liste imbriquée. Pour définir « `/` » comme ⟨*séparateur*⟩ d'une liste simple, il faut, à l'aide de l'argument optionnel, choisir un autre ⟨*séparateur*⟩ de listes imbriquées, par exemple « `.` » et écrire « `\setsepchar[.]{/}` ».

Il n'est pas possible de choisir « `|` » comme ⟨*séparateur*⟩ car il entrerait en conflit avec l'opérateur logique **OU** noté « `||` » (voir plus bas). On peut cependant contourner cette limitation, à ses risques et périls, en écrivant « `\setsepchar{{|}}` ».

Lire la liste Pour lire la liste d'éléments, la commande `\readlist`⟨*macroliste*⟩`{`⟨*liste*⟩`}` doit être appelée. Ce faisant, la ⟨*liste*⟩ est lue et les éléments sont stockés dans une macro, notée ⟨*macroliste*⟩ qui dès lors, se comporte comme un tableau composé des éléments de la ⟨*liste*⟩. Un élément est un ensemble de tokens dont les accolades *doivent* être équilibrées. Les tokens de catcode 6 seuls, 14 et 15 ne sont pas autorisés dans les listes.
Par exemple, pour définir la ⟨*macroliste*⟩ nommée `\foo`, on peut écrire

```
\setsepchar{,}
\readlist\foo{12,abc,x y ,{\bfseries z},,\TeX,,!}
```

Si la ⟨*liste*⟩ est contenue dans une macro, alors cette macro est développée par listofitems. On peut donc écrire `\readlist`⟨*macroliste*⟩⟨*macro*⟩ ce qui donnerait

```
\setsepchar{,}
\def\liste{12,abc,x y ,{\bfseries z},,\TeX,,!}
\readlist\foo\liste
```

Accéder à un élément La macro `\foo` attend un argument numérique *obligatoire* entre crochets, que nous notons i et qui désigne le rang de l'élément auquel on souhaite accéder. Ainsi, `\foo[1]` est [3] « `12` ». De la même façon, `\foo[4]` est « `{\bfseries z}` ».
Le nombre i peut également être négatif auquel cas le comptage se fait à partir de la fin de la liste : -1 représente le dernier rang, -2 l'avant-dernier, etc. Si le nombre d'éléments est n, alors l'argument $-n$ est le premier élément.

D'une façon générale, si une ⟨*liste*⟩ a une longueur n, alors l'index i peut se trouver dans l'intervalle $[1\,;n]$ ou $[-n\,;-1]$ et dans le cas contraire, une erreur de compilation survient.
Si l'index est vide, alors `\foo[]` se développe en la ⟨*liste*⟩ entière.
La macro `\foosep` est créé. Elle s'utilise avec la syntaxe `\foosep[`⟨*index*⟩`]` et permet d'accéder au séparateur qui suit l'élément de rang ⟨*index*⟩. Le dernier séparateur (celui qui suit le dernier élément) est défini par listofitems. Si l'⟨*index*⟩ est vide, `\foosep[]` a un développement vide.

3. Il faut 2 développements à `\foo[`i`]` pour obtenir l'élément n° i.

Choisir plusieurs séparateurs possibles Pour spécifier plusieurs séparateurs possibles, il faut utiliser l'opérateur **OU** noté « `||` ». On peut par exemple utiliser cette fonctionnalité pour isoler les termes dans une somme algébrique :

```
\setsepchar{+||-}
\readlist\terme{17-8+4-11}
1) \terme[1] (séparateur = \termesep[1])\par
2) \terme[2] (séparateur = \termesep[2])\par
3) \terme[3] (séparateur = \termesep[3])\par
4) \terme[4] (séparateur = \termesep[4])
```

1) 17 (séparateur = -)
2) 8 (séparateur = +)
3) 4 (séparateur = -)
4) 11 (séparateur = +)

Nombre d'éléments Si l'on écrit `\readlist`⟨*macroliste*⟩`{`⟨*liste*⟩`}` alors la macro ⟨*macroliste*⟩`len` contient [4] le nombre d'éléments de la ⟨*liste*⟩. Dans l'exemple avec `\foo`, la macro `\foolen` contient 8.

Afficher tous les éléments À des fins de débogage, la macro `\showitems`⟨*macroliste*⟩ compose tous les éléments d'une liste tandis que que sa version étoilée affiche ces éléments « détokénisés [5] ».

```
\showitems\foo\par
\showitems*\foo
```

12 abc x y z [] TEX ‖ !
12 abc x y {\bfseries z} [] \TeX ‖ !

La présentation de chaque élément est confiée à la macro `\showitemsmacro` dont le code est

```
\newcommand\showitemsmacro[1]{%
    \begingroup\fboxsep=0.25pt \fboxrule=0.5pt \fbox{\strut#1}\endgroup
    \hskip0.25em\relax}
```

Il est donc possible — et souhaitable — de la redéfinir si l'on cherche un autre effet.

La macro `\fbox` et ses dimensions afférentes `\fboxsep` et `\fboxrule` sont définies par listofitems lorsqu'on ne compile pas sous LaTeX de façon à obtenir le même résultat qu'avec LaTeX.

Suppression des espaces extrêmes Par défaut, listofitems lit et prend en compte le (ou les) espaces se trouvant au début et à la fin d'un élément. Pour que ces espaces soient ignorés lors de la lecture de la ⟨*liste*⟩, il faut exécuter la version étoilée `\readlist*`⟨*macro*⟩`{`⟨*liste*⟩`}` :

```
\setsepchar{,}
\readlist*\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!}
\showitems\foo
```

12 abc x y z ‖ TEX ‖ !

Gestion des éléments vides Par défaut, listofitems prend en compte les éléments vides. Ainsi, dans l'exemple précédent, le 2-développement de `\foo[7]` est vide. Pour que des éléments vides (ceux délimités par deux séparateurs consécutifs dans la liste) soient ignorés, il faut, avant de lancer la macro `\readlist`, exécuter la macro `\ignoreemptyitems`. La macro `\reademptyitems` revient au comportement par défaut. Cette option peut être utilisée seule ou combinée avec `\readlist*` auquel cas la suppression des espaces extrêmes intervient *avant* que listofitems n'ignore les éléments vides :

```
\setsepchar{,}
\ignoreemptyitems
\readlist\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!}
a) nombre d'éléments = \foolen\par
   \showitems\foo

\readlist*\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!}
b) nombre d'éléments = \foolen\par
   \showitems\foo
```

a) nombre d'éléments = 7
12 abc x y z [] TEX !
b) nombre d'éléments = 6
12 abc x y z TEX !

4. C'est-à-dire qu'elle est purement développable et se développe en un nombre
5. La primitive `\detokenize` qui procède à cette dénaturation insère un espace après chaque séquence de contrôle.

Itérer sur la liste Une fois une liste lue par `\readlist` et stockée dans une ⟨*macroliste*⟩, la commande `\foreachitem` ⟨*variable*⟩ `\in` ⟨*macroliste*⟩`{`⟨*code*⟩`}` itère sur la liste : la ⟨*variable*⟩ est une macro choisire par l'utilisateur qui prendra tour à tour la valeur de chaque élément.
La macro ⟨*variable*⟩`cnt` représente le numéro de l'élément contenu dans ⟨*variable*⟩.

```
\setsepchar{ }% séparateur = espace
\readlist\phrase{Une phrase de test.}
\foreachitem\mot\in\phrase{Le mot numéro \motcnt{} : \mot\par}
```

Le mot numéro 1 : Une
Le mot numéro 2 : phrase
Le mot numéro 3 : de
Le mot numéro 4 : test.

Assigner un élément à une macro La commande `\itemtomacro`⟨*macroliste*⟩`[index]`⟨*macro*⟩ assigne à la ⟨*macro*⟩ l'élément désigné par ⟨*macroliste*⟩`[index]`. La ⟨*macro*⟩ ainsi définie est purement développable, sous réserve que l'élément qu'elle contient le soit.

```
\setsepchar{ }% séparateur = espace
\readlist\phrase{Une phrase de test.}
\itemtomacro\phrase[2]\unmot
\meaning\unmot\par
\itemtomacro\phrase[-1]\motdelafin
\meaning\motdelafin
```

macro:->phrase
macro:->test.

3 Listes imbriquées

On parle de liste « imbriquée » lorsque l'on demande à listofitems de lire une liste où les éléments sont à leur tour compris comme une liste (dont le séparateur est différent de la liste de niveau supérieur). Le nombre d'imbrication n'est pas limité, mais dans la pratique, un niveau d'imbrication de 2, voire 3, semble un maximum.

Définir les séparateurs Pour indiquer que les éléments de la liste doivent eux-mêmes être compris comme des listes et que la recherche des éléments sera récursive, il faut spécifier plusieurs ⟨*séparateurs*⟩, chacun correspondant à un niveau d'imbrication. Pour déclarer une ⟨*liste de séparateurs*⟩ il faut définir le ⟨*séparateur*⟩ de cette ⟨*liste de séparateurs*⟩ à l'aide de l'argument optionnel de la macro `\setsepchar` et écrire `\setsepchar[`⟨*separateur*⟩`]{`⟨*liste des séparateurs*⟩`}`.
Par défaut, le ⟨*séparateur*⟩ est « / ». Ainsi, si l'on donne l'ordre

```
\setsepchar{\\/,/ }
```

on indique une profondeur récursive de 3 et on choisit comme séparateur de la ⟨*liste des séparateurs*⟩ le caractère par défaut « / » :

- les éléments de niveau 1 sont trouvés entre les séparateurs « \\ » ;
- les éléments de niveau 2 sont trouvés dans les éléments de niveau 1 entre les séparateurs « , » ;
- enfin, les éléments de niveau 3 sont trouvés dans ceux de niveau 2 entre les séparateurs « ␣ ».

La ⟨*profondeur*⟩ de recherche est contenue dans la macro purement développable `\nestdepth`.

Lire et accéder aux éléments Pour les listes imbriquées, les index obéissent à la règle suivante :

- `[]` désigne la liste principale, c'est-à-dire l'argument de `\readlist` ;
- `[`⟨*i*⟩`]` désigne l'élément n° ⟨*i*⟩ de la liste principale ;
- `[`⟨*i*⟩`,`⟨*j*⟩`]` désigne l'élément n° ⟨*j*⟩ de la liste constituée par l'élément évoqué au point précédent ;
- `[`⟨*i*⟩`,`⟨*j*⟩`,`⟨*k*⟩`]` désigne l'élément n° ⟨*k*⟩ de la liste constituée par l'élément évoqué au point précédent ;
- etc.

Comme pour les liste non imbriquées, les index peuvent être négatifs.

Pour lire les éléments, la syntaxe de \readlist est exactement la même qu'avec les listes simples :

```
\setsepchar{\\/,/ }
\readlist\baz{1,2 a b,3 c\\4 d e f,5,6\\7,,8, ,9 xy z}
a) \string\baz[1] est \baz[1]\par
b) \string\baz[1,1] est \baz[1,1]\par
c) \string\baz[1,1,1] est \baz[1,1,1]\par
b) \string\bar[1,2] est \baz[1,2]\par
e) \string\baz[1,2,3] est \baz[1,2,3]\par
f) \string\baz[-2,1,-1] est \baz[-2,1,-1]
```

a) \baz[1] est 1,2 a b,3 c
b) \baz[1,1] est 1
c) \baz[1,1,1] est 1
b) \bar[1,2] est 2 a b
e) \baz[1,2,3] est b
f) \baz[-2,1,-1] est f

L'opérateur « || » Cet opérateur peut se trouver dans n'importe quel niveau d'imbrication.

```
\setsepchar[,]{+||-,*||/}
\readlist\nombres{1+2*3-4/5*6}
Terme 1 : \nombres[1]\par
Terme 2 : \nombres[2] (facteurs : \nombres[2,1] et
          \nombres[2,2])\par
Terme 3 : \nombres[3] (facteurs : \nombres[3,1],
          \nombres[3,2] et \nombres[3,3])
```

Terme 1 : 1
Terme 2 : 2*3 (facteurs : 2 et 3)
Terme 3 : 4/5*6 (facteurs : 4, 5 et 6)

Nombre d'éléments La macro \listlen⟨*macrolist*⟩[⟨*index*⟩] nécessite 2 développements pour donner le nombre d'éléments de la liste spécifiée par l'⟨*index*⟩.

La ⟨*profondeur*⟩ de l'⟨*index*⟩ doit être strictement inférieure à celle de la ⟨*liste*⟩.

Dans le cas d'un ⟨*index*⟩ vide, \listlen⟨*macrolist*⟩[] donne en 2 développements le même résultat que ⟨*macrolist*⟩len qui le donne en 1.

```
\setsepchar{\\/,/ }
\readlist\baz{1,2 a b,3 c\\4 d e f,5,6\\7,,8, ,9 xy z}
a) \bazlen\ ou \listlen\baz[]\par
b) \listlen\baz[1]\par
c) \listlen\baz[2]\par
d) \listlen\baz[3]\par
e) \listlen\baz[3,1]\par
f) \listlen\baz[3,4]\par% 2 éléments vides
g) \listlen\baz[3,5]
```

a) 3 ou 3
b) 3
c) 3
d) 5
e) 1
f) 2
g) 3

Afficher les éléments La macro \showitems⟨*macrolist*⟩[⟨*index*⟩] affiche les éléments de la liste spécifiée par ⟨*index*⟩, selon le même principe que \listlen.

La ⟨*profondeur*⟩ de l'⟨*index*⟩ doit être strictement inférieure à celle de la ⟨*liste*⟩.

```
\setsepchar{\\/,/ }
\readlist\baz{1,2 a b,3 c\\4 d e f,5,6\\7,,8, ,9 xy z}
a) \showitems\baz[]\par
b) \showitems\baz[1]\par
c) \showitems\baz[2]\par
d) \showitems\baz[3]\par
e) \showitems\baz[3,1]\par
f) \showitems\baz[3,4]\par% 2 éléments vides
g) \showitems\baz[3,5]
```

a) [1,2 a b,3 c] [4 d e f,5,6] [7„8, ,9 xy z]
b) [1] [2 a b] [3 c]
c) [4 d e f] [5] [6]
d) [7] [] [8] [] [9 xy z]
e) [7]
f) [] []
g) [9] [xy] [z]

Éléments vides et espaces extrêmes La suppression des éléments vides et/ou des espaces extrêmes intervient dans *tous* les éléments, quel que soit le degré d'imbrication. Il est clair que choisir un espace comme séparateur est inutile si l'on veut utiliser \readlist*. C'est pourquoi dans cet exemple, « * » est choisi comme séparateur.

Dans cet exemple, on ne supprime que les espaces extrêmes en gardant les éléments vides.

```
\setsepchar{\\/,/*}
\readlist*\baz{1, 2*a*b ,3*c\\4*d*e*f,5,6\\7,,8, ,9* xy *z}
a) \showitems\baz[]\par
b) \showitems\baz[1]\par
c) \showitems\baz[2]\par
d) \showitems\baz[3]\par
e) \showitems\baz[3,1]\par
f) \showitems\baz[3,4]\par
g) \showitems\baz[3,5]% "xy" sans espaces extrêmes
```

a) [1, 2*a*b ,3*c] [4*d*e*f,5,6] [7„8, ,9* xy *z]
b) [1] [2*a*b] [3*c]
c) [4*d*e*f] [5] [6]
d) [7] ‖ [8] ‖ [9* xy *z]
e) [7]
f) ‖
g) [9] [xy] [z]

Itérer sur une liste La syntaxe `\foreachitem` ⟨*variable*⟩ `\in` ⟨*macro*⟩`[`⟨*index*⟩`]{`⟨*code*⟩`}` reste valable où désormais, l'⟨*index*⟩ spécifie sur quel élément (compris comme une liste) on veut itérer.

La ⟨*profondeur*⟩ de l'⟨*index*⟩ doit être strictement inférieure à celle de la ⟨*liste*⟩.

Assigner un élément à une macro La syntaxe `\itemtomacro`⟨*macroliste*⟩`[`⟨*index*⟩`]`⟨*macro*⟩ reste valable pour assigner à ⟨*macro*⟩ l'élément spécifié par ⟨*macroliste*⟩`[`⟨*index*⟩`]`.

```
\setsepchar[,]{\\, }
\readlist\poeme{J'arrive tout couvert encore de rosée\\%
Que le vent du matin vient glacer à mon front.\\%
Souffrez que ma fatigue à vos pieds reposée\\%
Rêve des chers instants qui la délasseront.}% 2e strophe de
« Green », Paul Verlaine
\itemtomacro\poeme[2]\vers
2e vers = \vers

\itemtomacro\poeme[2,-4]\mot
Un mot = \mot
```

2e vers = Que le vent du matin vient glacer à mon front.
Un mot = glacer