L'extension

listofitems

v1.64

10 février 2024

Christian Tellechea [*]

Traducteur : 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@netc.fr
†. steven.b.segletes.civ@mail.mil

1 Avant-propos

Important : pour toute modification du code source de cette extension (remontée de bug, demande d'ajout ou modification de fonctionnalité), j'invite les utilisateurs de listofitems à envoyer un email à unbonpetit@netc.fr. Il est notamment inutile d'envoyer un email à une autre adresse ou de poster sur un site internet spécialisé.

Cette extension ne requiert aucun package, doit être utilisée avec un moteur ε-TeX fournissant la primitive \expanded, et doit être appelée sous (pdf)(Xe)(lua)LaTeX par

```
\usepackage{listofitems}
```

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

```
\input listofitems.tex
```

Je remercie Steven B. Segletes pour la traduction de ce manuel mais aussi pour l'énergie, la motivation et la persévérance qu'il déploie pour apporter de l'aide aux utilisateurs de listofitems en répondant à leurs questions sur un site spécialisé. J'avoue que, vu l'ampleur et la répétitivité de la tâche, je serais lassé depuis belle lurette.

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
```

La macro \greadlist agit comme \readlist, mais effectue des assignations *globales* et donc, la ⟨*macroliste*⟩ est utilisable hors du groupe où a été exécutée \greadlist.

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.

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

Accéder à un séparateur Lorsque \readlist\foo{⟨*liste*⟩} est exécuté, 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 vide. Si l'⟨*index*⟩ est vide, \foosep[] a un développement vide.

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 choisie 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 deux développements le même résultat que ⟨*macrolist*⟩len qui le donne en un seul.

```
\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

La macro \gitemtomacro fait une assignation globale.

4 Tokens appariés

Pour le découpage des items, il est possible à partir de la version 1.6, de tenir compte de la présence de caractères *appariés*. Ainsi, si une liste de caractères appariés est définie, chaque item s'étend jusqu'au prochain ⟨*séparateur*⟩ qui équilibre les tokens appariés.
Pour définir une liste de tokens appariés, on utilise

\defpair{⟨*tok1*⟩⟨*tok2*⟩⟨*tok3*⟩⟨*tok4*⟩...}

où les tokens sont lus deux par deux pour former les paires d'appariement. Un ⟨*token*⟩ d'appariement doit être constitué d'un seul caractère ; les macros, primitives, espaces, accolades, token «#», ensembles de plusieurs tokens entre accolades sont interdits. Deux tokens formant une paire *doivent* être différents.

```
\setsepchar{+|-}
\defpair{()[]}
\readlist\termes{1+2*[3+4*(5+6-7)+8]-9+10}
\showitems\termes
```

1 2*[3+4*(5+6-7)+8] 9 10

Pour revenir au comportement par défaut, c'est-à-dire sans tokens appariés, il faut exécuter

\defpair{}

Dans une expression, pour stocker dans une macro ce qui se trouve entre deux tokens appariés, on peut faire appel à

\insidepair⟨*tok1*⟩⟨*tok2*⟩{⟨*expression*⟩}\macro

qui mettra dans la \macro ce qui de trouve entre la paire ⟨*tok1*⟩⟨*tok2*⟩ dans l'⟨*expression*⟩.

```
\setsepchar{+|-}
\defpair{()}
\readlist\termes{1+2*(3+4*(5+6-7)+8)-9+10}
\showitems\termes

\itemtomacro\termes[2]\parenterm
Dans la parenthèse extérieure :
\insidepair()\parenterm\inbigparen
"\inbigparen"

Dans la parenthèse intérieure :
\insidepair()\inbigparen\insmallparen
"\insmallparen"
```

1 2*(3+4*(5+6-7)+8) 9 10
Dans la parenthèse extérieure : "3+4*(5+6-7)+8"
Dans la parenthèse intérieure : "5+6-7"

5 Le code

Le code ci-dessous est l'exact verbatim du fichier listofitems.tex. J'espère que les quelques commentaires qui le parsèment de-ci de-là seront suffisants pour que l'utilisateur ou le curieux comprenne la machinerie interne de ce package :

```
% Ce fichier contient le code de l'extension "listofitems"
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
\def\loiname                {listofitems}                                      %
\def\loiver                     {1.64}                                         %
%                                                                              %
\def\loidate                 {2024/02/10}                                      %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% Author     : Christian Tellechea                                             %
% Status     : Maintained                                                      %
% Maintainer : Christian Tellechea                                             %
% Email      : unbonpetit@netc.fr                                              %
% Package URL: https://www.ctan.org/pkg/listofitems                            %
% Copyright  : Christian Tellechea 2016-2024                                   %
% Licence    : Released under the LaTeX Project Public License v1.3c           %
%              or later, see http://www.latex-project.org/lppl.txt             %
% Files      : 1) listofitems.tex                                              %
%              2) listofitems.sty                                              %
%              3) listofitems-fr.tex                                           %
%              4) listofitems-fr.pdf                                           %
%              5) listofitems-en.tex                                           %
%              6) listofitems-en.pdf                                           %
%              7) README                                                       %
%              8) listofitems-test-tex.tex                                     %
%              9) listofitems-test-tex.pdf                                     %
%             10) listofitems-test-latex.tex                                   %
%             11) listofitems-test-latex.pdf                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\csname loiloadonce\endcsname
\let\loiloadonce\endinput
\expandafter\edef\csname loi_restorecatcode\endcsname{%
  \catcode\number`\_=\number\catcode`\_\relax
}
\catcode`\_11

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% gestion des erreurs + annonce package %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ifdefined\loi_fromsty
  \def\loi_error#1{\PackageError\loiname{#1}{Read the \loiname\space manual}}% pour LaTeX
\else
  \def\loi_error#1{\errmessage{Package \loiname\space Error: #1^^J}}% pour TeX
  \immediate\write -1 {Package: \loidate\space v\loiver\space Grab items in lists using user-specified sep char (CT)}%
\fi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% vérification des prérequis %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\loi_checkprimitive#1#2{% Vérifie que #1 est une primitive et sinon, émet le message #2 et exécute \endinput
  \begingroup
    \edef\__tempa{\meaning#1}%
    \edef\__tempb{\string #1}%
    \expandafter
  \endgroup
  \ifx\__tempa\__tempb\else
    \loi_error{#2}%
    \loi_restorecatcode
    \expandafter\endinput
  \fi
}
\loi_checkprimitive\eTeXversion
  {%
  You are not using an eTeX engine, listofitems cannot work.%
  }%
\loi_checkprimitive\expanded
  {%
  The \string\expanded\space primitive is not provided by your TeX engine, , listofitems cannot work.%
```

```
  }%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macros auxiliaires %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\loi_quark{\loi_quark}
\long\def\loi_identity#1{#1}
\long\def\loi_gobarg#1{}
\long\def\loi_first#1#2{#1}
\long\def\loi_second#1#2{#2}
\long\def\loi_firsttonil#1#2\_nil{#1}
\long\def\loi_antefi#1#2\fi{#2\fi#1}
\long\def\loi_exparg#1#2{%
  \expandafter\loi_exparg_a\expandafter{#2}{#1}% \loi_exparg{<a>}{<b>} devient <a>{<*b>}
}
\long\def\loi_exparg_a#1#2{#2{#1}}
\long\def\loi_expafter#1#2{%
  \expandafter\loi_expafter_a\expandafter{#2}{#1}% \loi_expafter{<a>}{<b>} devient <a><*b>
}
\long\def\loi_expafter_a#1#2{#2#1}
\def\loi_macroname{%
  \loi_ifinrange\escapechar[[0:255]]%
    {%
    \expandafter\loi_gobarg
    }
    {%
    }%
    \string}
\def\loi_argcsname#1#{\loi_argcsname_a{#1}}
\def\loi_argcsname_a#1#2{\loi_expafter{#1}{\csname#2\endcsname}}
\long\def\loi_addtomacro#1#2{\loi_exparg{\def#1}{#1#2}}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macros de test %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\long\def\loi_ifnum#1{%
  \ifnum#1%
    \expandafter\loi_first
  \else
    \expandafter\loi_second
  \fi
}
\long\def\loi_ifx#1{%
  \ifx#1%
    \expandafter\loi_first
  \else
    \expandafter\loi_second
  \fi
}
\long\def\loi_ifempty#1{%
  \loi_exparg\loi_ifx{\expandafter\relax\detokenize{#1}\relax}%
}
\def\loi_ifstar#1#2{%
  \def\loi_ifstar_a{\loi_ifx{*\loi_nxttok}{\loi_first{#1}}{#2}}%
  \futurelet\loi_nxttok\loi_ifstar_a
}
\edef\loi_escapechar{\expandafter\loi_gobarg\string\\}
\long\def\loi_ifcsexpandable#1{% #1 est-il constitué d'une seule sc _développable_ ?
  \loi_ifempty{#1}
    {%
    \loi_second
    }
    {\loi_ifspacefirst{#1}
      {%
      \loi_second% si espace en 1er, faux
      }
      {%
      \csname loi_\if\loi_escapechar\expandafter\loi_firsttonil\detokenize{#1}\_nil first\else second\fi\endcsname
        {%
        \loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul arg commençant par "\" ?
```

```
            {%
            \def\loi_tempa{#1}\loi_exparg{\def\loi_tempb}{#1}% est-il développable ?
            \expandafter\unless\loi_ifx{\loi_tempa\loi_tempb}%
            }
            {%
            \loi_second
            }%
          }
          {%
          \loi_second
          }%
        }%
      }%
}
\def\loi_ifinrange#1[[#2:#3]]{%
  \expandafter\unless\loi_ifnum{\numexpr(#1-#2)*(#1-#3)>0 }%
}
\def\loi_ifstring#1#2{% si la chaine #1 est contenue dans #2
  \def\loi_ifstring_a##1#1##2\_nil{%
    \loi_ifempty{##2}
      \loi_second
      \loi_first
  }%
  \loi_ifstring_a#2#1\_nil% appel de la macro auxiliaire
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% macro \loi_foreach %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\loi_cnt_foreach_nest
\loi_cnt_foreach_nest=0
\def\end_foreach{\end_foreach}% quark
\def\loi_def_foreachsep#1{%
  \long\def\loi_foreach##1\in##2##3{%
    \global\advance\loi_cnt_foreach_nest1
    \loi_argcsname\def{loop_code_\number\loi_cnt_foreach_nest}{##3}%
    \loi_foreach_a##1##2#1\end_foreach#1%
    \loi_argcsname\let{loop_code_\number\loi_cnt_foreach_nest}\empty
    \global\advance\loi_cnt_foreach_nest-1
  }%
  \long\def\loi_foreach_a##1##2#1{%
    \def##1{##2}%
    \loi_ifx{\end_foreach##1}
      {}
      {%
      \csname loop_code_\number\loi_cnt_foreach_nest\endcsname% exécute le code
      \loi_foreach_a##1%
      }%
  }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% macros gérant l'appariement %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\long\def\defpair#1{%
  \let\loi_listofpair\empty
  \loi_ifempty{#1}
    {%
    }
    {%
    \defpair_a{}#1\loi_quark\loi_quark
    }%
}
\long\def\defpair_a#1#2#3{%
  \loi_ifx{\loi_quark#2}
    {%
    \def\loi_sanitizelist##1,\_nil{\def\loi_listofpair{##1}}%
    \loi_sanitizelist#1\_nil
    }
    {%
```

```
    \loi_if_validpair#2#3%
      {%
      \long\def\loi_paired_a{#2}\long\def\loi_paired_b{#3}%
      \loi_ifx{\loi_paired_a\loi_paired_b}
        {%
        \loi_error{Paired tokens must not be equal, the pair \detokenize{#2#3} is ignored}%
        \defpair_a{#1}%
        }
        {%
        \defpair_a{#1#2#3,}%
        }%
      }
      {%
      \loi_error{Invalid paired tokens, the pair ”\detokenize{#2}” and ”\detokenize{#3}” is ignored}%
      \defpair_a{#1}%
      }%
    }%
}
\long\def\loi_if_validpair#1#2{%
  \def\loi_validpair{1}%
  \loi_if_invalid_pairtoken{#1}
    {%
    \def\loi_validpair{0}%
    }%
  \loi_if_invalid_pairtoken{#2}
    {%
    \def\loi_validpair{0}%
    }%
  \loi_ifnum{\loi_validpair=1 }
}
\long\def\loi_if_invalid_pairtoken#1{%
  \loi_ifempty{#1}
    {%
    \loi_identity
    }
    {%
    \loi_ifspacefirst{#1}
      {%
      \loi_identity
      }
      {%
      \loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul token ?
        {%
        \ifcat\relax\noexpand#1%
          \expandafter\loi_identity
        \else
          \expandafter\loi_gobarg
        \fi
        }
        {%
        \loi_identity% si plusieurs tokens, faux
        }%
      }%
    }%
}
\long\def\loi_count_occur#1\in#2:#3{% compte le nombre d'occurrences de #1 dans #2 et met le résultat dans la macro #3
  \long\def\loi_count_occur_a##1##2#1##3\_nil{%
    \loi_ifempty{##3}
      {%
      \def#3{##1}%
      }
      {%
      \expandafter\loi_count_occur_a\number\numexpr##1+1\relax##3\_nil
      }%
  }%
  \loi_count_occur_a0#2#1\_nil
}
\long\def\loi_check_pair#1#2\in#3{% teste l'appariement de #1 et #2 dans #3
  \loi_ifempty{#3}
    {%
```

```
    \loi_second
    }
    {%
    \loi_count_occur#1\in#3:\loi_tempa
    \loi_count_occur#2\in#3:\loi_tempb
    \loi_ifnum{\loi_tempa=\loi_tempb\relax}%
    }%
}
\long\def\loi_grabpaired_expr#1#2#3#4#5{% #1=liste de paires  #2=expression  #3=séparateur #4=résultat    #5=reste
  \let#4\empty
  \def\loi_remain{#2#3}%
  \loi_foreach\loi_pair\in{#1}{%
    \expandafter\loi_grabpaired_expr_a\loi_pair{#3}#4%
    }%
  \def\loi_remove_lastsep##1#3\_nil{\def#4{##1}}%
  \expandafter\loi_remove_lastsep#4\_nil
  \loi_expafter{\long\def\loi_grab_remain}#4##1\_nil{%
    \loi_ifempty{##1}
      {%
      \let#5\empty
      }
      {%
      \loi_exparg{\def#5}{\loi_gobarg##1}%
      }%
  }%
  \loi_grab_remain#2\_nil
}
\long\def\loi_grabpaired_expr_a#1#2#3#4{% #1#2=paire en cours  #3=séparateur   #4=résultat
  \loi_exparg{\loi_check_pair#1#2\in}#4% si les paires sont appariées dans le résultat
    {%
    }% passer à la paire suivante
    {%
    \long\def\loi_grabpaired_expr_b##1#3##2\_nil{%
      \loi_addtomacro#4{##1#3}% ajouter au résultat ce qui est jusqu'au prochain séparateur
      \def\loi_remain{##2}%
      \loi_exparg{\loi_check_pair#1#2\in}{#4}
        {%
        }
        {%
        \loi_ifempty{##2}
          {%
          \loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired}%
          }
          {%
          \loi_grabpaired_expr_b##2\_nil
          }%
        }%
      }%
    \expandafter\loi_grabpaired_expr_b\loi_remain\_nil
    }%
}
\def\insidepair#1#2#3#4{% #1#2=paire  #3=expr  #4=macro reçevant le resultat
  \loi_if_validpair#1#2%
    {%
    \loi_ifcsexpandable{#3}
      {%
      \loi_exparg{\insidepair#1#2}{#3}#4%
      }
      {%
      \loi_check_pair#1#2\in{#3}% si les paires sont appariées dans le résultat
        {%
        \def\insidepair_a##1#1##2\_nil{\insidepair_b##2\_nil{#1}}%
        \def\insidepair_b##1#2##2\_nil##3{%
          \loi_check_pair#1#2\in{##3##1#2}
            {%
            \loi_exparg{\def#4}{\loi_gobarg##3##1}%
            \def\loi_remainafterparen{##2}%
            }%
            {%
            \insidepair_b##2\_nil{##3##1#2}%
```

```
            }%
          }%
        \insidepair_a#3\_nil
        }
        {%
        \loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired in "#3"}%
        }%
      }%
    }
    {%
    \loi_error{Invalid paired tokens "\detokenize{#1}" and "\detokenize{#2}", empty \string#4 returned}% et bim
    \let#4\empty% voilà, bien fait pour vos gueules
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% macro \loi_fornum %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\loi_fornum#1=#2to#3\do{%
  \edef#1{\number\numexpr#2}%
  \expandafter\loi_fornum_a
    \csname loi_fornum_\string#1\expandafter\endcsname\expandafter
    {\number\numexpr#3\expandafter}%
    \expanded{\ifnum#1<\numexpr#3\relax>+\else<-\fi}%
    #1%
}
\long\def\loi_fornum_a#1#2#3#4#5#6{%
  \def#1{%
    \unless\ifnum#5#3#2\relax
      \loi_antefi{#6\edef#5{\number\numexpr#5#41\relax}#1}%
    \fi%
  }%
  #1%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%% macro retirant les espaces extrêmes %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\long\def\loi_ifspacefirst#1{%
  \expandafter\loi_ifspacefirst_a\detokenize{#10} \_nil
}
\long\def\loi_ifspacefirst_a#1 #2\_nil{%
  \loi_ifempty{#1}%
}
\loi_expafter{\def\loi_gobspace}\space{}
\long\def\loi_removefirstspaces#1{%##BUGFIX v1.63
  \loi_ifspacefirst{#1}
    {\loi_exparg\loi_removefirstspaces{\loi_gobspace#1}}
    {\unexpanded{#1}}%
}%
\begingroup
  \catcode0 12
  \long\gdef\loi_removelastspaces#1{\loi_removelastspaces_a#1^^00 ^^00\_nil}
  \long\gdef\loi_removelastspaces_a#1 ^^00{\loi_removelastspaces_b#1^^00}
  \long\gdef\loi_removelastspaces_b#1^^00#2\_nil{\loi_ifspacefirst{#2}{\loi_removelastspaces_a#1^^00 ^^00\_nil}{\↙
      unexpanded{#1}}}
\endgroup
\long\def\loi_removeextremespaces#1{\expanded{\loi_exparg\loi_removelastspaces{\expanded{\loi_removefirstspaces↙
    {#1}}}}}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% macro publique \setsepchar %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\setsepchar{\futurelet\loi_nxttok\setsepchar_a}
\def\setsepchar_a{%
  \loi_ifx{[\loi_nxttok}
    {%
    \setsepchar_b
    }
    {%
```

```
    \setsepchar_b[/]%
    }%
}
\long\def\setsepchar_b[#1]#2{% #1=sepcar de <liste des sepcar>  #2=<liste des sepcar>
  \loi_ifempty{#1}
    {%
    \loi_error{Empty separator not allowed, separator "/" used}%
    \setsepchar_b[/]{#2}%
    }
    {%
    \def\loi_currentsep{#1}%
    \_loi_removeextremespacesfalse
    \loi_nestcnt1 % réinitaliser niveau initial à 1
    \def\nestdepth{1}%
    \loi_argcsname\let{loi_previndex[\number\loi_nestcnt]}\empty
    \def\loi_listname{loi_listofsep}%
    \let\loi_def\def
    \let\loi_edef\edef
    \let\loi_let\let
    \let\loi_listofpair_saved\loi_list_ofpair
    \let\loi_list_ofpair\empty
    \loi_ifempty{#2}
      {%
      \loi_error{Empty list of separators not allowed, "," used}%
      \readlist_g1{,}%
      }
      {%
      \readlist_g1{#2}%
      }%
    \loi_argcsname\let\nestdepth{loi_listofseplen[0]}%
    \loi_argcsname\let\loi_currentsep{loi_listofsep[1]}% 1er car de séparation
    \let\loi_listofpair\loi_listofpair_saved
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% macro normalisant l'index %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\loi_normalizeindex#1#2#3{% #1=correction de profondeur #2=macroname  #3=liste d'index  --> renvoie {err}{indx ↙
      norm}
  \loi_ifempty{#3}
    {%
    {}{}%
    }
    {%
    \loi_exparg{\loi_normalizeindex_a1{}}{\number\numexpr\csname#2nest\endcsname-#1\relax}{#2}#3,\loi_quark,%
    }%
}%
\def\loi_normalizeindex_a#1#2#3#4#5,{% #1=compteur de profondeur #2=index précédents  #3=profondeur max #4=macroname  ↙
      #5=index courant
  \loi_ifx{\loi_quark#5}
    {%
    \loi_normalizeindex_c#2\loi_quark% supprimer la dernière virgule
    }
    {%
    \loi_ifnum{#1>#3 }
      {%
      \loi_invalidindex{Too deeply nested index, index [.] retained}{#2}% si profondeur trop grande
      }
      {%
      \loi_ifinrange\ifnum\numexpr#5<0 -1*\fi(#5)[[1:\csname #4len[#20]\endcsname]]% si abs(#5) hors de [1,len]
        {%
        \loi_exparg\loi_normalizeindex_b
          {\number\numexpr#5\ifnum\numexpr#5<0 +\csname #4len[#20]\endcsname+1\fi}%
          {#1}%
          {#2}%
          {#3}%
          {#4}%
        }
        {%
```

```
          \loi_invalidindex{#5 is an invalid index, index [.] retained}{#2}%
          }%
        }%
    }%
}
\def\loi_normalizeindex_b#1#2#3{%
  \loi_exparg\loi_normalizeindex_a{\number\numexpr#2+1}{#3#1,}% #1=index à rajouter  #2=compteur de profondeur #3=↙
      index précédents
}
\def\loi_normalizeindex_c#1,\loi_quark{{}{#1}}
\def\loi_invalidindex#1#2{%
  \loi_ifempty{#2}
    {%
    \loi_invalidindex_a{#1},%
    }
    {%
    \loi_invalidindex_a{#1}{#2}% BUGFIX 1.64
    }%
}
\def\loi_invalidindex_a#1#2{%
  \loi_invalidindex_b#1\loi_quark#2\loi_quark
}
\def\loi_invalidindex_b#1[.]#2\loi_quark#3,\loi_quark#4\loi_quark,{% #4= index ignorés
  {#1[#3]#2}{#3}%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% macro publique \readlist %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\loi_nestcnt
\def\greadlist{%
  \let\loi_def\gdef
  \let\loi_edef\xdef
  \def\loi_let{\global\let}%
  \readlist_a
}%
\def\readlist{%
  \let\loi_def\def
  \let\loi_edef\edef
  \let\loi_let\let
  \readlist_a%
}
\def\readlist_a{%
  \loi_nestcnt1 % niveau initial = 1
  \loi_argcsname\let{loi_previndex[\number\loi_nestcnt]}\empty
  \loi_ifstar
    {%
    \_loi_removeextremespacestrue
    \readlist_b
    }
    {%
    \_loi_removeextremespacesfalse
    \readlist_b
    }%
}
\long\def\readlist_b#1#2{% #1=macro stockant les éléments  #2=liste des éléments
  \loi_ifcsexpandable{#2}
    {%
    \loi_exparg{\readlist_b#1}{#2}%
    }
    {%
    \loi_edef\loi_listname{\loi_macroname#1}%
    \loi_exparg{\readlist_c#1{#2}}{\loi_listname}%%
    }%
}
\long\def\readlist_c#1#2#3{% #1=macro stockant les éléments  #2=liste des éléments #3=macroname
  \loi_argcsname\loi_let{#3nest}\nestdepth
  \loi_argcsname\loi_def{#3[]}{#2}% la liste entière
  \loi_argcsname\loi_def{#3sep[]}{}% séparateur vide
  \loi_ifempty{#2}
```

```
    {%
    \loi_def#1[##1]{}%
    \loi_argcsname\loi_def{#3len}{0}\loi_argcsname\loi_def{#3len[0]}{0}%
    \loi_error{Empty list ignored, nothing to do}%
    }
    {%
    \loi_def#1[##1]{\expanded{\expandafter\readlist_d\expanded{\loi_normalizeindex0{#3}{##1}}{#3}}}%
    \loi_argcsname\loi_def{#3sep}[##1]{\expanded{\expandafter\readlist_d\expanded{\loi_normalizeindex0{#3}{##1}}{#3sep}}}%
    \readlist_e{#2}%
    \loi_argcsname\loi_argcsname\loi_let{#3len}{#3len[0]}% longueur du niveau 0
    }%
}
\def\readlist_d#1#2#3{%
  \unexpanded\expandafter\expandafter\expandafter{\csname#3[#2]\expandafter\endcsname\expandafter}%
  \expanded{\loi_ifempty{#1}{}{\unexpanded{\unexpanded{\loi_error{#1}}}}}%
}
\def\readlist_e{%
  \loi_argcsname\loi_let\loi_currentsep{loi_listofsep[\number\loi_nestcnt]}%
  \expandafter\readlist_f\loi_currentsep||\_nil
}
\long\def\readlist_f#1||#2\_nil#3{\readlist_g1{#3#1}}% #1=<sep courant simple>  #3=liste -> rajoute un élément vide pour le test \ifempty ci dessous
\long\def\readlist_g#1#2{% #1=compteur d'index  #2=liste d'éléments à examiner terminée par <sep courant simple> >>RIEN laissé après
  \loi_ifempty{#2}
    {%
    \loi_argcsname\loi_edef{\loi_listname len[\csname loi_previndex[\number\loi_nestcnt]\endcsname0]}{\number\numexpr#1-1\relax}%
    \loi_argcsname\loi_let{\loi_listname sep[\csname loi_previndex[\number\loi_nestcnt]\endcsname\number\numexpr#1-1\relax]}\empty% le dernier <sep> est <vide> ##NEW v1.52
    \advance\loi_nestcnt-1
    \loi_argcsname\loi_let\loi_currentsep{loi_listofsep[\number\loi_nestcnt]}%
    }
    {%
    \loi_expafter{\readlist_h{#2}{}}\loi_currentsep||\loi_quark||#2\_nil{#1}% aller isoler le 1er item
    }%
}
\long\def\readlist_h#1#2#3||{% #1=liste restante   #2=<dernier sep utilisé>  #3=<sep courant>
  \loi_ifx{\loi_quark#3}% on a épuisé tous les <séparateurs> ? RESTE à lire <expr+sep1>\_nil{<compteur>}
    {%
    \loi_ifempty{#2}% si #2 vide, aucun <sep utilisé> n'a été trouvé, il reste à lire "<liste complète>\_nil"
      {%
      \long\def\readlist_i##1\_nil##2{\loi_exparg{\readlist_j{##2}{}}{\loi_gobarg##1}{#2}}% ##2=compteur d'index
      }
      {%
      \loi_ifx{\loi_listofpair\empty}% paires définies ?
        {%
        \long\def\readlist_i##1#2##2\_nil##3{\loi_exparg{\readlist_j{##3}{##2}}{\loi_gobarg##1}{#2}}%
        }
        {%
        \long\def\readlist_i##1\_nil##2{%
          \loi_exparg{\loi_exparg\loi_grabpaired_expr\loi_listofpair}{\loi_gobarg##1}{#2}\loi_grabpaired_result\loi_grabpaired_remain
          \loi_exparg{\loi_exparg{\readlist_j{##2}}{\loi_grabpaired_remain}}{\loi_grabpaired_result}{#2}}%{#}
        }%
      }%
    \readlist_i\relax% le \relax meuble l'argument délimité
    }
    {%
    \long\def\readlist_i##1#3##2\_nil{%
      \loi_ifempty{##2}% si <liste restante> ne contient pas le <sep courant>
        {%
        \readlist_h{#1}{#2}% recommencer avec le même <sep utile>
        }%
        {%
        \loi_ifx{\loi_listofpair\empty}% si pas de paires définies
          {%
          % ##BUGFIX v1.53 (manger le \relax)
          % ##BUGFIX v 1.64 (##2 n'étant pas vide, ne pas ajouter #3 après ##1
```

```
          %                  puisque #3 subsiste toujours avant le \_nil)
          \loi_exparg\readlist_h{\loi_gobarg##1}{#3}% raccourcir <liste restante> et <sep courant>:=<sep utile>
          }%
          {%
          \loi_exparg\loi_grabpaired_expr\loi_listofpair{#1#3}{#3}\loi_grabpaired_result\loi_grabpaired_remain
          \loi_ifx{\loi_grabpaired_remain\empty}% si liste non raccourcie #BUGFIX 1.63
            {\loi_exparg\readlist_h{\loi_grabpaired_result}{#2}}% garder le précédent <sep>
            {\loi_exparg\readlist_h{\loi_grabpaired_result}{#3}}%
          }%
        }%
      }%
    \readlist_i\relax#1#3\_nil% ##BUGFIX v1.53 (ajouter \relax)
    }%
}
\long\def\readlist_j#1#2#3{% #1=compteur d'index  #2=liste restante  #3=élément courant
  \loi_ifnum{0\loi_exparg\loi_ifspacefirst{\loi_currentsep}{}1\if_loi_removeextremespaces1\fi=11 }% s'il faur retirer les espaces extrêmes
    {%
    \loi_exparg{\loi_exparg{\readlist_k{#1}{#2}}}{\loi_removeextremespaces{#3}}% redéfinir l'élément courant
    }%
    {%
    \readlist_k{#1}{#2}{#3}%
    }%
}
\long\def\readlist_k#1#2#3#4{% #1=compteur d'index  #2=liste restante  #3=élément courant   #4=sep utilisé
  \loi_ifnum{0\if_loi_ignoreemptyitems1\fi\loi_ifempty{#3}1{}=11 }
    {%
    \readlist_g{#1}{#2}% si l'on n'ignore pas les éléments vides
    }%
    {%
    \loi_argcsname\loi_def{\loi_listname[\csname loi_previndex[\number\loi_nestcnt]\endcsname#1]}{#3}% assignation
    \loi_argcsname\loi_def{\loi_listname sep[\csname loi_previndex[\number\loi_nestcnt]\endcsname#1]}{#4}% assignation du <sep> actuel à la macro \<macrolist>sep
    \loi_ifnum{\loi_nestcnt<\nestdepth\relax}% si imbrication max non atteinte
      {%
      \advance\loi_nestcnt1
      \loi_argcsname\edef{loi_previndex[\number\loi_nestcnt]}{\csname loi_previndex[\number\numexpr\loi_nestcnt-1]\endcsname#1,}%
      \readlist_e{#3}% recommencer avec l'élément courant
      }
      {%
      }%
    \loi_exparg\readlist_g{\number\numexpr#1+1}{#2}% puis chercher l'élément suivant dans la liste restante
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% macro \listlen %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\listlen#1[#2]{%
  \expanded{%
    \loi_ifempty{#2}
      {%
      \csname\loi_macroname#1len[0]\endcsname
      }
      {%
      \loi_exparg\listlen_a{\expanded{\loi_macroname#1}}{#2}%
      }%
  }%
}
\def\listlen_a#1#2{% #1=macro name   #2=index non normalisé    prendre <profondeur max-1>
  \expandafter\listlen_b\expanded{\loi_normalizeindex1{#1}{#2}}{#1}%
}
\def\listlen_b#1#2#3{% #1=err   #2=index normalisé  #3=macroname
  \csname#3len[#2,0]\expandafter\endcsname
  \expanded{%
    \loi_ifempty{#1}
      {%
      }
      {%
```

```
      \unexpanded{\unexpanded{\loi_error{#1}}}%
      }%
  }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macro \foreachitem %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\foreachitem#1\in#2{%
  \edef\foreachitem_a{\noexpand\foreachitem_c\noexpand#1{\expandafter\noexpand\csname\loi_macroname#1cnt\endcsname}{\loi_macroname#2}}%
  \futurelet\loi_nxttok\foreachitem_b
}
\def\foreachitem_b{\loi_ifx{\loi_nxttok[}\foreachitem_a{\foreachitem_a[]}}
\def\foreachitem_c#1#2#3[#4]{% prendre <profondeur max-1>
  \expandafter\foreachitem_d\expanded{\loi_normalizeindex1{#3}{#4}}#1{#2}{#3}%
}
\def\foreachitem_d#1#2{%
  \loi_ifempty{#2}
    {%
    \foreachitem_e{#1}{}%
    }
    {%
    \foreachitem_e{#1}{#2,}% #1=err  #2=index norm
    }%
}%
\long\def\foreachitem_e#1#2#3#4#5#6{% #1=err  #2=index norm  #3=macroiter  #4=compteur associé  #5=nom de macrolist  #6=code
  \loi_ifnum{\csname#5len[#20]\endcsname>0 }
    {%
    \loi_ifempty{#1}
      {%
      }
      {%
      \loi_error{#1}%
      }%
    \loi_fornum#4=1to\csname#5len[#20]\endcsname\do{\loi_argcsname\let#3{#5[#2#4]}#6}%
    }
    {%
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macro \showitem %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\showitems{%
  \loi_ifstar
    {%
    \let\showitems_cmd\detokenize
    \showitems_a
    }
    {%
    \let\showitems_cmd\loi_identity
    \showitems_a
    }%
}
\def\showitems_a#1{%
  \def\showitems_b{\showitems_d#1}%
  \futurelet\loi_nxttok\showitems_c
}
\def\showitems_c{%
  \loi_ifx{\loi_nxttok[}
    {%
    \showitems_b
    }
    {%
    \showitems_b[]}%
}
\def\showitems_d#1[#2]{%
  \foreachitem\showitems_iter\in#1[#2]
```

```
    {%
    \showitemsmacro{\expandafter\showitems_cmd\expandafter{\showitems_iter}}%
    }%
}
\unless\ifdefined\fbox
  \newdimen\fboxrule
  \newdimen\fboxsep
  \fboxrule=.4pt
  \fboxsep=3pt % réglages identiques à LaTeX
  \def\fbox#1{% imitation de la macro \fbox de LaTeX, voir pages 271 à 274 de "Apprendre à programmer en TeX"
    \hbox{%
      \vrule width\fboxrule
      \vtop{%
        \vbox{\hrule height\fboxrule \kern\fboxsep \hbox{\kern\fboxsep#1\kern\fboxsep}}%
        \kern\fboxsep \hrule height\fboxrule
      }%
      \vrule width\fboxrule
    }%
  }
\fi
\def\showitemsmacro#1{% encadrement par défaut
  \begingroup
    \fboxsep=0.25pt
    \fboxrule=0.5pt
    \fbox{\strut#1}%
  \endgroup
  \hskip0.25em\relax
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% macro \itemtomacro %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\itemtomacro#1[#2]{% #1[#2]=item  non encore lu: #3=macro
  \edef\loi_listname{\loi_macroname#1}%
  \expandafter\itemtomacro_a\expanded{\loi_normalizeindex0{\loi_listname}{#2}}\let
}
\def\gitemtomacro#1[#2]{% #1[#2]=item
  \xdef\loi_listname{\loi_macroname#1}%
  \expandafter\itemtomacro_a\expanded{\loi_normalizeindex0{\loi_listname}{#2}}{\global\let}%
}
\def\itemtomacro_a#1#2#3#4{%
  \loi_ifempty{#1}{}{\loi_error{#1}}%
  \loi_argcsname#3#4{\loi_listname[#2]}%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% réglages par défaut %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newif\if_loi_removeextremespaces
\newif\if_loi_ignoreemptyitems
\let\ignoreemptyitems\_loi_ignoreemptyitemstrue
\let\reademptyitems\_loi_ignoreemptyitemsfalse
\loi_def_foreachsep{,}
\loi_restorecatcode
\reademptyitems
\setsepchar{,}
\defpair{}
\endinput

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%  historique %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v1.0    19/8/2016
  - Première version publique
------------------------------------------------------------------------
v1.1    01/09/2016
  - Stockage des séparateurs dans <macrolist>sep
  - bug corrigé dans \loi_restorecatcode
------------------------------------------------------------------------
v1.2    22/10/2016
```

```
  - macros \greadlist et \gitemtomacro pour la globalité
-------------------------------------------------------------------------
v1.3    18/11/2016
  - bugs corrigés dans la gestion de la globalité
-------------------------------------------------------------------------
v1.4    05/10/2017
  - test \loi_ifprimitive ajouté au test \loi_ifcs
  - suppression de \loi_expafternil, création de \loi_expafter,
    modification de \loi_argcsname
  - correction d'un bug : \setsepchar{\par} ne provoque plus d'erreur.
    \loi_ifnum devient \long
-------------------------------------------------------------------------
v1.5    06/10/2017
  - correction d'un bug dans \loi_ifcs
-------------------------------------------------------------------------
v1.51   24/10/2017
  - correction d'un bug dans \loi_ifcs
-------------------------------------------------------------------------
v1.52   13/01/2018
  - le dernier séparateur est <vide>
-------------------------------------------------------------------------
v1.53   13/03/2018
  - correction d'un bug dans \readlist_i
-------------------------------------------------------------------------
v1.6    01/11/2018
  - possibilité d'appariement de tokens dans les items
-------------------------------------------------------------------------
v1.61   03/03/2019
  - la macro \loi_ifcs contient une erreur de conception. Il faut
    tester si le token est un sc && s'il est développable pour
    renvoyer vrai car il existe des sc non développables && qui ne
    sont _pas_ des primitives.
    Macro rebaptisée \loi_ifcsexpandable
-------------------------------------------------------------------------
v1.62   18/05/2019
  - utilisation de la nouvelle primitive \expanded au lieu du
    désormais obsolète \romannumeral
  - bug corrigé dans \loi_ifcsexpandable
-------------------------------------------------------------------------
v1.63   21/08/2019
  - bug corrigé dans \readlist_h avec les tokens appariés
  - bug corrigé \loi_removefirstspaces est désormais \long
  ------------------------------------------------------------------------
v1.64   10/02/2024
  - bug corrigé dans \readlist_h. Bug découvert plus d'un an après (sic)
    avoir été signalé dans la question :
                tex.stackexchange.com/questions/670379
    et donc désormais, le code suivant ne plante plus
       \setsepchar{10||00}%
       \readlist\mylist{A10B}%
  - bug corrigé dans \loi_normalizeindex  : la profondeur maxi de
    l'index peut être diminuée de 1 (pour notamment \foreachitem et
    \listlen)
  - la primitive \expanded est désormais obligatoire
  - code source UFT8, manuel compilé avec luaLaTeX
  - code plus lisiblement formaté
```