L'extension pour TEX/LATEX

hlist

v 0.1

1 mai 2017

Christian Tellechea [1]

Cette petite extension sans prétention permet de créer des listes horizontale d'items ; s'ils s'étendent sur plus d'une ligne, ils seront alignés les uns au-dessous des autres, en colonnes de largeurs égales, un peu à la manière d'un tableau.

1. unbonpetit@openmailbox.org

1 Comment ça marche

Au tout début, j'avais écrit ce code pour présenter des questions courtes *en ligne*, comme on peut le voir parfois dans les manuels scolaires, pour une économie de place verticale dans mes feuilles d'exercices ou de contrôles [2] :

```
Qu'est ce que \TeX ?
\begin{hlist}3
  \hitem Une marque \emph{distributeur} de vêtements que l'on trouve dans l'enseigne Carrefour;
  \hitem Une unité de mesure de la finesse de filaments;
  \hitem Un logiciel de composition créé par Donald \bsc{Knuth};
  \hitem Le nom \emph{de scène} de l'humoriste Jean-Christophe \bsc{Le Texier};
  \hitem Une membrane coupe-vent.
\end{hlist}
```

Qu'est ce que TEX ?

1. Une marque *distributeur* de vêtements que l'on trouve dans l'enseigne Carrefour ;
2. Une unité de mesure de la finesse de filaments ;
3. Un logiciel de composition créé par Donald Knuth ;
4. Le nom *de scène* de l'humoriste Jean-Christophe Le Texier ;
5. Une membrane coupe-vent.

Le code initial dont je me suis servi pendant longtemps était bien plus basique que celui de cette extension ; depuis, il s'est complexifié, mais les méthodes qu'il mettait en œuvre sont restées les mêmes. Or, j'avais programmé les listes horizontales en utilisant des *boites* de TEX et non pas des *alignements*. Évidemment, il semble plus naturel de faire appel aux alignements de TEX pour réaliser une telle tâche, mais leur mise en œuvre est plus délicate que celle des boites et j'ai été partisan du moindre effort (et je le suis toujours).

Récemment [3], alors que la finalisation du code avait déjà commencée, j'ai découvert que l'extension `Tasks` offrait *presque* les mêmes fonctionnalités que `hlist`. Comme *presque* n'est pas synonyme d'*exactement*, je me suis finalement décidé à publier cette extension. Il n'est bien entendu pas dans mon intention de dénigrer le travail de C. Niederberger et encore moins de considérer `hlist` comme meilleure que `Tasks`, mais au niveau des principales différences entre `Tasks` et `hlist`, on peut citer :

— `hlist` peut être utilisé avec TEX (ε-TEXen réalité) alors que `Tasks` requiert LATEX ;

— les listes de `hlist` peuvent être imbriquées tandis que celles de `Tasks` ne le peuvent pas ;

— les changements de catcode (et donc le « verbatim ») sont possibles dans les listes de `hlist` et pas dans celles de `Tasks` ;

— le code source de `hlist` est plus concis que celui de `Tasks` (écrit en LATEX3) et, contrairement à ce dernier, `hlist` ne fait appel à aucune extension tierce [4] ;

— la documentation de `Tasks` est en anglais (son auteur est allemand) ; celle de `hlist` est en français. Cesser de docilement se plier à cette prétendue nécessité de fournir une documentation en anglais, fusse pour des extensions de TEX/LATEX, compte *beaucoup* à mes yeux.

En revanche, puisque les deux extensions s'appuient sur des constructions de boites, `hlist` partage avec `Tasks` le principal défaut : une coupure de page ne peut pas avoir lieu au sein d'un item qui s'étendrait sur plusieurs lignes.

2 Syntaxe

L'environnement `hlist` peut être appelé de ces deux manières :

— façon LATEX : `\begin{hlist}[`⟨*paramètres*⟩`]`⟨*nombre*⟩
```
    contenu de l'environnement
\end{hlist}
```

— façon TEX : `\hlist[`⟨*paramètres*⟩`]`⟨*nombre*⟩
```
   contenu de l'environnement
\endhlist
```

2. On ne trouve pas ce genre de questions dans mes contrôles !
3. http://forum.mathematex.net/latex-f6/macros-a-parametres-indefini-t16733.html#p155215
4. Vu les facilités prétendument offertes par LATEX3 et les extensions chargées par `Tasks`, je suis resté médusé face à la longueur du code de `Tasks`. Je ne m'explique pas un telle différence entre la taille des deux codes alors que `hlist` embarque son propre système de clés/valeurs, ses macros de développement, etc. Avant de me lancer dans une critique sans ménagement de LATEX3 et comme j'ai un fort soupçon que quelque chose m'échappe là-dedans, je serais très reconnaissant si quelqu'un, capable de comprendre ce qui se passe, pouvait me fournir une explication.

Les options, décrites plus loin, seront celles qui s'appliquent à cette liste et le ⟨*nombre*⟩ est le nombre de colonnes demandées.

L'environnement `hlist` doit commencer par un item, c'est-à-dire par la macro `\hitem`, éventuellement suivie des arguments optionnels suivants :

1. "`*`" signifiant que cet item doit commencer une nouvelle ligne ;
2. "`>`" signifiant que cet item doit s'étendre sur toutes les colonnes restant disponibles sur la ligne en cours ;
3. l'argument entre crochets "`[`⟨*label*⟩`]`" spécifie ce que doit être la numérotation de cet item (voir la description de cette fonctionnalité plus loin) ;
4. l'argument entre parenthèses "`(`⟨*n*⟩`)`" où ⟨*n*⟩ est un entier, indique sur combien de colonnes doit d'étendre cet item.

Il faut noter que, s'ils sont présents tous les deux, l'argument optionnel "`*`" *doit* précéder "`>`". En revanche pour les deux derniers arguments optionnels, l'ordre n'a pas d'importance et donc "`[`⟨*label*⟩`](`⟨*n*⟩`)`" et accepté tout comme "`(`⟨*n*⟩`)[`⟨*label*⟩`]`".

```
\def\txtA{Après le pain, l'éducation est le premier besoin du peuple.}
\def\txtB{Comment se fait-il que les enfants étant si intelligents,
          la plupart des hommes soient bêtes ?\par Cela doit tenir à l'éducation.}
\hlist3
  \hitem \txtA      \hitem \txtB      \hitem \txtA
  \hitem \txtA      \hitem> \txtB
  \hitem*> \txtB
  \hitem(2) \txtB      \hitem \txtA
\endhlist
```

---

1. Après le pain, l'éducation est le premier besoin du peuple.

2. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
Cela doit tenir à l'éducation.

3. Après le pain, l'éducation est le premier besoin du peuple.

4. Après le pain, l'éducation est le premier besoin du peuple.

5. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
Cela doit tenir à l'éducation.

6. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
Cela doit tenir à l'éducation.

7. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
Cela doit tenir à l'éducation.

8. Après le pain, l'éducation est le premier besoin du peuple.

On peut voir à la page 6 cette même liste horizontale avec des cadres tracés autour des boites, ce qui peut rendre les choses plus claires pour cet exemple, volontairement complexe et peu réaliste.

3 Choix des paramètres

Nous avons déjà vu qu'écrire `\hlist[`⟨*paramètres*⟩`]` permet de régler des ⟨*paramètres*⟩ spécifiques à la liste qui suit.

On peut également régler des ⟨*paramètres*⟩ pour toutes les `hlist` à venir avec

`\sethlist{`⟨*paramètres*⟩`}`

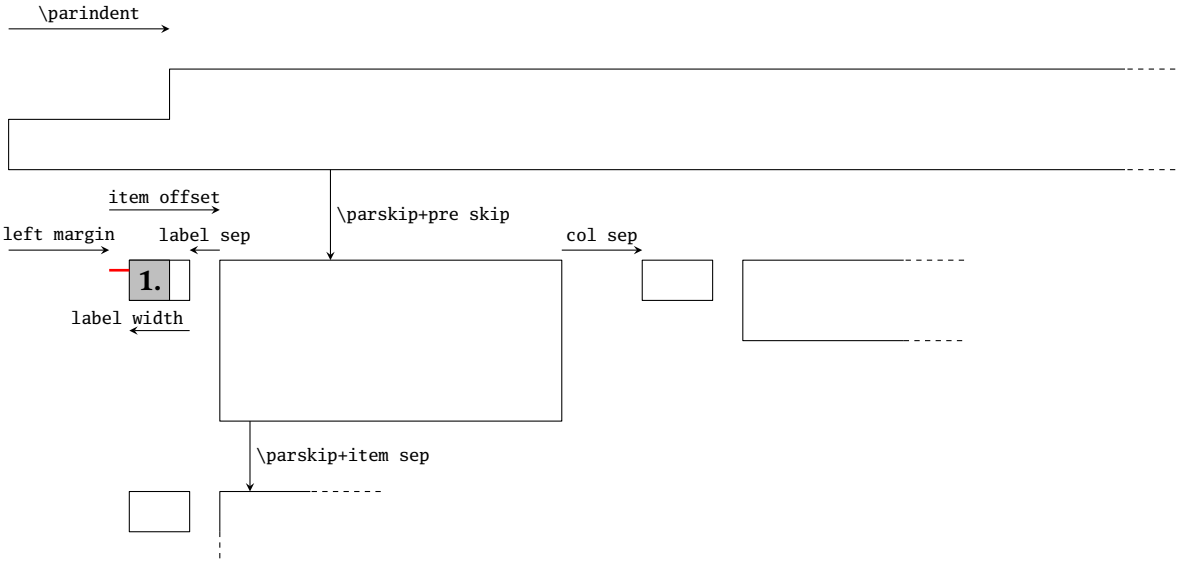
mais aussi régler des ⟨*paramètres*⟩ *par défaut* pour les `hlist` à venir avec

`\setdefaulthlist{`⟨*paramètres*⟩`}`

Chaque paramètre est de la forme ⟨*clé*⟩=⟨*valeur*⟩ où la ⟨*clé*⟩ est le plus souvent constituée de deux mots (l'espace entre ces deux mots est *nécessaire*). Les espaces avant et après la ⟨*clé*⟩ et la ⟨*valeur*⟩ sont ignorés.

Lorsqu'une ⟨*clé*⟩ est seule, elle est comprise comme un *booléen* qui vaut `true`. Pour indiquer qu'un booléen est faux, il faut assigner la valeur `false`.

Les paramètres de *placement* sont résumés dans le schéma ci-dessous :



4 Placement de la liste de plus haut niveau

pre skip=⟨ressort⟩ (*Défaut* : \medskipamount)
Valeur du ⟨*ressort*⟩ vertical inséré avant la liste *de plus haut niveau d'imbrication.*

post skip=⟨ressort⟩ (*Défaut* : \medskipamount)
Valeur du ⟨*ressort*⟩ vertical inséré après la liste *de plus haut niveau d'imbrication.*

left margin=⟨dimension⟩ (*Défaut* : 0pt)
Valeur du décalage vers la droite de la liste *de plus haut niveau d'imbrication.*

```
\parindent=2em
Ceci est un paragraphe s'étendant sur deux ligne et présentant
une liste de questions mises en forme à l'aide de l'extension \texttt{hlist}
et avec les paramètres par défaut :
\hlist3% paramètres par défaut
  \hitem Question 1          \hitem Question 2
  \hitem Une question 3
  \hitem Question 4          \hitem Question 5
\endhlist
Ceci est un paragraphe s'étendant sur deux ligne et présentant
une liste de questions mises en forme à l'aide de l'extension \texttt{hlist}
et avec des paramètres spécifiés par argument optionnel :
\hlist[pre skip = 0pt , post skip = 4pt , left margin = \parindent]3
  \hitem Question 1          \hitem Question 2
  \hitem Une question 3
  \hitem Question 4          \hitem Question 5
\endhlist
Suite du texte
```

Ceci est un paragraphe s'étendant sur deux ligne et présentant une liste de questions mises en forme à l'aide de l'extension `hlist` et avec les paramètres par défaut :

1. Question 1 **2.** Question 2 **3.** Une question 3
4. Question 4 **5.** Question 5

Ceci est un paragraphe s'étendant sur deux ligne et présentant une liste de questions mises en forme à l'aide de l'extension `hlist` et avec des paramètres spécifiés par argument optionnel :

1. Question 1 **2.** Question 2 **3.** Une question 3
4. Question 4 **5.** Question 5

Suite du texte

5 Label et texte des items

item offset=⟨dimension⟩ (*Défaut* : 1.75em)

Cette ⟨*dimension*⟩ est la distance entre la frontière gauche de la boite contenant le label et la frontière gauche de la boite contenant le texte de l'item.

label sep=⟨dimension⟩ (*Défaut* : 0.25em)

Espacement entre la boite contenant le label et celle contenant le texte de l'item.

label width=⟨dimension⟩ (*Défaut* : *)

C'est la dimension de la boite accueillant le label. Le label lui-même (en gris sur la figure) peut être plus court que cette dimension auquel cas, il faut jouer avec la clé « label align » (voir plus bas) pour indiquer quel alignement on souhaite.

Si en revanche le label est *plus large* que la largeur « label width », alors un dépassement du type « Overful hbox » aura lieu, sauf si on met la clé autoindent à la valeur true (voir plus bas).

Les trois clés « item offset », « label sep » et « label width » sont en interdépendance pour le placement du label et plusieurs cas peuvent se présenter :

— si l'une des 3 trois (et seulement *une seule* sur les trois) vaut « * », alors, les 2 autres sont calculées de telles sorte que l'égalité suivante soit vérifiée

$$\texttt{item offset} = \texttt{label sep} + \texttt{label width}$$

— si item offset > label sep + label width comme c'est le cas sur la figure, alors la différence entre le membre de gauche et celui de droite (trait rouge sur la figure) est meublé avec un \hfill ;

— si item offset < label sep + label width, ce qui ne devrait pas se produire, un avertissement sera émis.

label align=⟨lettre⟩ (*Défaut* : left)

Cette clé représente la consigne d'alignement du label à l'intérieur de sa boite. Elle n'a de sens que lorsque le label occupe moins de place horizontale que « label width ». On peut demander une consigne l, r ou c selon que l'on souhaite un alignement à gauche, à droite ou centré.

Seule le première lettre *minuscule* de la valeur est prise en compte. La consigne left est donc équivalente à l.

label=⟨code⟩ (*Défaut* : \arabic {hlisti}.)

Le ⟨*code*⟩ sera exécuté pour afficher le label. Pour afficher des labels en rapport avec le rang de l'item dans la liste, hlist met à disposition les macros suivantes :

— \arabic{⟨*nom*⟩} pour afficher un entier ;

— \roman{⟨*nom*⟩} et \Roman{⟨*nom*⟩} pour afficher des nombres romains, en minuscule ou majuscule ;

— \alpha{⟨*nom*⟩} et \Alpha{⟨*nom*⟩} pour afficher des lettres, minuscules ou majuscules.

L'argument de ces macros est le *nom* d'un compteur au format LaTeX, c'est-à-dire que le registre de compteur qui se trouve derrière est \c@⟨*nom*⟩ [5].

hlist fournit les compteurs nommés « hlisti » pour les listes de plus haut niveau, « hlistii » pour les listes imbriquées de niveau 1, « hlistiii » pour celle de niveau 2, etc.

pre label=⟨code⟩ (*Défaut* : \bfseries)

Contient le ⟨*code*⟩ qui sera exécutée avant d'afficher le label.

post label=⟨code⟩ (*Défaut* : {})

Contient le ⟨*code*⟩ qui sera exécutée après avoir affiché le label.

```
Exemple 1 :\hlist[label=\alpha{hlisti})]3
  \hitem question 1 \hitem question 2 \hitem question 3 \hitem question 4
  \endhlist
Exemple 2:\hlist[label=---]3
  \hitem question 1 \hitem question 2 \hitem question 3 \hitem question 4
  \endhlist
Exemple 3:\hlist[pre label=\begingroup\color{red}\itshape,label=\roman{hlisti}-,post label=\endgroup]3
  \hitem question 1 \hitem question 2 \hitem question 3 \hitem question 4
  \endhlist
```

5. Les compteurs de LaTeX pour l'environnement enumerate, à savoir enumi, enumii, etc sont bien évidemment admis dans les arguments des macros ci-dessus. En règle générale, tout compteur de LaTeX l'est également (section, equation, page, footnote ou autre).

Exemple 1 :

a) question 1 **b)** question 2 **c)** question 3
d) question 4

Exemple 2 :

— question 1 — question 2 — question 3
— question 4

Exemple 3 :

i- question 1 *ii-* question 2 *iii-* question 3
iv- question 4

Lorsque l'argument optionnel entre crochet de `\hitem` est vide, aucun label ne sera affiché et la valeur « `item offset` » sera nulle pour cet item. Cela peut s'avérer utile dans le cas de listes imbriquées :

```
Exemple avec 2\up e label numéroté :
\hlist3
  \hitem question 1
  \hitem% label numéroté pour cet item !
    \hlist[label=\alpha{hlistii}.,col sep=1em]3
      \hitem XX \hitem YY \hitem ZZ
    \endhlist
  \hitem question 3 \hitem question 4 \hitem question 5
\endhlist

Même exemple avec 2\up e label forcé à vide :
\hlist3
  \hitem question 1
  \hitem[]% pas de label pour cet item !
    \hlist[label=\arabic{hlisti}\alpha{hlistii}.]3
      \hitem XX \hitem YY \hitem ZZ
    \endhlist
  \hitem question 3 \hitem question 4 \hitem question 5
\endhlist
```

Exemple avec 2e label numéroté :

1. question 1 **2. a.** XX **b.** YY **c.** ZZ **3.** question 3
4. question 4 **5.** question 5

Même exemple avec 2e label forcé à vide :

1. question 1 **2a.** XX **2b.** YY **2c.** ZZ **3.** question 3
4. question 4 **5.** question 5

pre label=⟨code⟩ (*Défaut* : `\bfseries`)
Il s'agit du ⟨*code*⟩ qui est exécuté juste avant que le label soit affiché.

post label=⟨code⟩ (*Défaut* : `{}`)
Cette valeur contient le ⟨*code*⟩ qui est exécuté juste après l'affichage du label.

resume=⟨booléen⟩ (*Défaut* : `false`)
Lorsque cette clé est assignée à `true`, la numérotation de la liste n'est pas réinitialisée lorsque la macro `\hlist` est exécutée.

autoindent=⟨booléen⟩ (*Défaut* : `false`)
Lorsque la largeur du label dépasse la dimension « `label width`, nous avons vu qu'un avertissement du type `Overful hbox` est produit. Lorsque la clé « `autoindent` » est mise à `true`, ce dépassement se fait en *indentant la première ligne du texte du label* de telle sorte que subsiste entre le label et le texte l'espace de largeur « `label sep` ». Il n'y a alors plus d'avertissement émis par TEX.

Ceci fonctionne évidemment pour un schéma de label *global* passé par la clé « `label={`⟨*code*⟩`}` », mais trouve surtout une utilité pour des labels *exceptionnels* spécifiés par `\hitem[`⟨*label*⟩`]`.

```
\def\txt{et voici le texte de la question}
\hlist[item sep=5pt]3
  \hitem Question 1, \txt           \hitem Question 2, \txt           \hitem[Bonus] Question 3, \txt
  \hitem Question 4, \txt           \hitem Question 5, \txt
\endhlist
Texte intermédiaire...
\hlist[item sep=5pt,autoindent=true]3
  \hitem Question 1, \txt           \hitem Question 2, \txt           \hitem[Bonus] Question 3, \txt
  \hitem Question 4, \txt           \hitem Question 5, \txt
\endhlist
```

---

1. Question 1, et voici le texte de la question
2. Question 2, et voici le texte de la question
Bonus Question 3, et voici le texte de la question
4. Question 4, et voici le texte de la question
5. Question 5, et voici le texte de la question

Texte intermédiaire...

1. Question 1, et voici le texte de la question
2. Question 2, et voici le texte de la question
Bonus Question 3, et voici le texte de la question
4. Question 4, et voici le texte de la question
5. Question 5, et voici le texte de la question

pre item=⟨code⟩ (*Défaut* : `{}`)
Le code exécuté juste avant le texte de l'item.

post item=⟨code⟩ (*Défaut* : `{}`)
Le code exécuté juste après le texte de l'item.

show label=⟨booléen⟩ (*Défaut* : `true`)
Cette clé, lorsqu'elle est assignée à `false`, interdit l'affichage du label et donc, tout se passe comme si les dimensions `item offset`, `label width` et `label sep` valaient `0pt`.

```
Exemple avec \texttt{show label} :
\hlist3
  \hitem Question 1          \hitem Question 2
  \hitem Une question 3
  \hitem Question 4          \hitem Question 5
\endhlist
Exemple sans \texttt{show label} :
\hlist[show label=false]3
  \hitem Question 1          \hitem Question 2
  \hitem Une question 3
  \hitem Question 4          \hitem Question 5
\endhlist
```

Exemple avec `show label` :

1. Question 1
2. Question 2
3. Une question 3
4. Question 4
5. Question 5

Exemple sans `show label` :

Question 1 Question 2 Une question 3
Question 4 Question 5

list parindent=⟨dimension⟩ (*Défaut* : `0pt`)
Cette clé spécifie l'indentation des paragraphes de l'item, sauf pour le premier paragraphe de chaque item qui reste non indenté.

show frame=⟨booléen⟩ (*Défaut* : `false`)
Cette clé, lorsqu'elle est assignée à `true`, encadre les boites contenant le label et le texte de l'item.

```
\def\txtA{Après le pain, l'éducation est le premier besoin du peuple.}
\def\txtB{Comment se fait-il que les enfants étant si intelligents,
          la plupart des hommes soient bêtes ?\par Cela doit tenir à l'éducation.}
\hlist[show frame=true]3
  \hitem \txtA      \hitem \txtB      \hitem \txtA
  \hitem \txtA      \hitem> \txtB
  \hitem*> \txtB
  \hitem(2) \txtB      \hitem \txtA
\endhlist
```

1. Après le pain, l'éducation est le premier besoin du peuple.
2. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
   Cela doit tenir à l'éducation.
3. Après le pain, l'éducation est le premier besoin du peuple.
4. Après le pain, l'éducation est le premier besoin du peuple.
5. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
   Cela doit tenir à l'éducation.
6. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
   Cela doit tenir à l'éducation.
7. Comment se fait-il que les enfants étant si intelligents, la plupart des hommes soient bêtes ?
   Cela doit tenir à l'éducation.
8. Après le pain, l'éducation est le premier besoin du peuple.

6 Le code

Contrairement à mes habitudes, le code est *abondamment* commenté, trop sans doute. J'espère que cela constituera une source d'information exploitable pour les utilisateurs de cette extension qui souhaitent y comprendre quelque chose ou modifier par eux-même une fonctionnalité.

Toute suggestion, remontée de bug, remarque, demande, ajout ou modification de fonctionnalité est bienvenue ; dans ce cas, j'invite les utilisateurs de `hlist` à m'envoyer un email.

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

```
% !TeX encoding = ISO-8859-1
% Ce fichier contient le code commenté de l'extension "hlist"
%
% IMPORTANT : pour que les commentaires s'affichent correctement, lire
%             ce fichier avec l'encodage ISO-8859-1
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
\def\hlstname                   {hlist}                                     %
\def\hlstver                     {0.1}                                      %
%                                                                           %
\def\hlstdate                 {2017/05/01}                                  %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% ----------------------------------------------------------------------
% 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@openmailbox.org
%        Commentaires, suggestions et signalement de bugs bienvenus !
%        Comments, bug reports and suggestions are welcome.
% ----------------------------------------------------------------------
% L'extension hlist est composée des 9 fichiers suivants :
%   - code               : hlist           (.tex et .sty)
%   - manuel en français : hlist-fr        (.tex et .pdf)
%   - fichiers de test   : hlist-latex     (.tex et .pdf)
%                          hlist-tex       (.tex et .pdf)
%   - fichier lisezmoi   : README
% ----------------------------------------------------------------------

\expandafter\edef\csname hlst_restorecatcode\endcsname{\catcode`\noexpand\_=\the\catcode`\_\relax}
\catcode`\_11

%##############################################
%############ macros de développement ############
%##############################################
% Cette macro est équivalente à 0 et sert notamment à stopper le développement
% de \romannumeral
\chardef\hlst_stop 0

% Définition du quark, notamment inséré à la fin d'une liste pour en reconnaitre
% la fin et stopper la récursivité
\def\hlst_quark{\hlst_quark}

% Voici les macros habituelles de sélection d'arguments
\def\hlst_gobarg#1{}
\long\def\hlst_first#1#2{#1}\long\def\hlst_second#1#2{#2}
\long\def\hlst_firsttonil#1#2\_nil{#1}

% Voici la macro pour 1-développer ou 2-développer le 2e argument (le 1er étant
% dépouillé des accolades)
%     \hlst_exparg{<a>}{<b>} devient <a>{<b>}
%     \hlst_eearg{<a>}{<b>} devient <a>{**<b>}
\long\def\hlst_exparg#1#2{\expandafter\hlst_exparg_i\expandafter{#2}{#1}}%
\long\def\hlst_eearg#1#2{\expandafter\expandafter\expandafter\hlst_exparg_i\expandafter\expandafter\↙
    expandafter{#2}{#1}}%
\long\def\hlst_exparg_i#1#2{#2{#1}}

% Et la macro pour 1-développer le 2e argument (le 1er et le 2e argument sont
% dépouillés des accolades)%
%     \hlst_expafter{<a>}{<b>} devient <a><*b>
\long\def\hlst_expafter#1#2{\expandafter\hlst_expafter_i\expandafter{#2}{#1}}
\long\def\hlst_expafter_i#1#2{#2#1}

% Enfin, la macro pour former le nom du 2e argument (le 1er est dépouillé des
% accolades)
%     \hlst_argcsname{<a>}{<b>} devient <a>\<b>
\def\hlst_argcsname#1#{\hlst_argcsname_i{#1}}
\def\hlst_argcsname_i#1#2{\hlst_expafter{#1}{\csname#2\endcsname}}

%##############################################
%################ macros de test ################
%##############################################
% Voici quelques macros à sélection d'arguments pour les tests
\def\hlst_ifnum#1{\ifnum#1\expandafter\hlst_first\else\expandafter\hlst_second\fi}
\def\hlst_ifcsname#1{\ifcsname#1\endcsname\expandafter\hlst_first\else\expandafter\hlst_second\fi}
\long\def\hlst_ifx#1{\ifx#1\expandafter\hlst_first\else\expandafter\hlst_second\fi}
\long\def\hlst_ifempty#1{\hlst_exparg\hlst_ifx{\expandafter\relax\detokenize{#1}\relax}}
\def\hlst_ifstar#1#2{\def\hlst_ifstar_i{\hlst_ifx{*\hlst_nxttok}{\hlst_first{#1}}{#2}}\futurelet\hlst_nxttok↙
    \hlst_ifstar_i}
\def\hlst_eaddtomacro#1#2{\hlst_exparg{\hlst_exparg{\def#1}}{\expandafter#1#2}}

% Ici, une macro pour définir le token <espace>
\def\hlst_deftok#1#2{\let#1= #2\relax}\hlst_deftok\hlst_sptok{ }
```

```
94
95  % Et une macro pour tester le prochain token
96  \def\hlst_ifnxttok#1#2#3{\hlst_deftok\hlst_toktotest{#1}\def\__truecode{#2}\def\__falsecode{#3}\↙
        hlst_ifnxttok_a}
97  \def\hlst_ifnxttok_a{\futurelet\_futurtok\hlst_ifnxttok_b}
98  \def\hlst_ifnxttok_b{%
99    \hlst_ifx{\_futurtok\hlst_sp}
100     {\afterassignment\hlst_ifnxttok_a\let\_futurtok= }
101     {\hlst_ifx{\_futurtok\hlst_toktotest}\__truecode\__falsecode}%
102 }
103
104 % Ces macros sont utiles pour \hlst_removeextremespaces, qui retire les
105 % espaces extrêmes de son argument
106 %     Voir codes 320-324 (http://progtex.fr/wp-content/uploads/2014/09/code.txt)
107 %     et pages 339-343 de "Apprendre à programmer en TeX"
108 \long\def\hlst_ifspacefirst#1{\expandafter\hlst_ifspacefirst_i\detokenize{#10} \_nil}
109 \long\def\hlst_ifspacefirst_i#1 #2\_nil{\hlst_ifempty{#1}}
110 \expandafter\def\expandafter\hlst_gobspace\space{}
111 \def\hlst_removefirstspaces{\romannumeral\hlst_removefirstspaces_i}
112 \long\def\hlst_removefirstspaces_i#1{\hlst_ifspacefirst{#1}{\expandafter\hlst_removefirstspaces_i\↙
        expandafter{\hlst_gobspace#1}}{\hlst_stop#1}}
113 \begingroup
114   \catcode0 12
115   \long\gdef\hlst_removelastspaces#1{\romannumeral\hlst_removelastspaces_i#1^^00 ^^00\_nil}
116   \long\gdef\hlst_removelastspaces_i#1 ^^00{\hlst_removelastspaces_ii#1^^00}
117   \long\gdef\hlst_removelastspaces_ii#1^^00#2\_nil{\hlst_ifspacefirst{#2}{\hlst_removelastspaces_i#1^^00 ↙
        ^^00\_nil}{\hlst_stop#1}}
118 \endgroup
119 \long\def\hlst_removeextremespaces#1{%
120   \romannumeral\expandafter\expandafter\expandafter\hlst_removelastspaces\expandafter\expandafter\↙
        expandafter
121   {\expandafter\expandafter\expandafter\hlst_stop\hlst_removefirstspaces{#1}}%
122 }
123
124 %################################################
125 %############### système clé/valeur ##############
126 %################################################
127 % Ceci est le booléen indiquant si la lecture de <clés>=<valeurs> définit les
128 % <clés> _par défaut_ ou qu'il s'agit d'une _redéfinition_ des <clés> déjà
129 % existantes
130 \newif\ifhlst_default
131
132 % macros chapeau appelant la même macro avec le booléen préalablement défini
133 \def\setKVdefault{\hlst_defaulttrue\hlst_readKV}
134 \def\setKV{\hlst_defaultfalse\hlst_readKV}
135
136 % L'argument obligatoire #1 est le nom du <trousseau> et #2 est l'ensemble
137 % des <clés>=<valeurs>
138 \def\hlst_readKV[#1]#2{%
139   \hlst_ifempty{#2}
140 % Si aucune <clés>=<valeurs> alors qu'on définit les <valeurs> par défaut,
141 % message d'erreur et on s'arrête là
142     {\ifhlst_default\errmessage{No key/val found, no default key/val defined}\fi}
143 % Sinon, initialiser à <vide> la macro \hlst_[<trousseau>] uniquement si on
144 % créé les <valeurs> par défaut
145     {\ifhlst_default\hlst_argcsname\let{hlst_[#1]}\empty\fi
146 % Puis on passe aux choses sérieuses, on va lire un par un tous les éléments
147 % <clé>=<valeur> (contenus dans #2) en mettant le quark comme dernier couple
148 % pour montrer la fin de la liste
149     \hlst_readKV_i[#1]#2,\hlst_quark,%
150     }%
151 }
152
153 \def\hlst_readKV_i[#1]#2,{%
154 % #2 est le premier couple "<clé>=<valeur>" de la liste qui reste à traiter :
155 % tout d'abord, on se débarrasse des espaces extrêmes
```

```
  \hlst_eearg{\def\__temp}{\hlst_removeextremespaces{#2}}%
% Si ce qui en résulte est égal au <quark>,
  \hlst_ifx{\__temp\hlst_quark}
% alors, on a fini et on ne fait rien (fin du processus)
    {}
% Sinon, si ce qui en résulte est vide (le couple "<clé>=<valeur>" était donc
% vide ou composé d'un espace)
    {\hlst_ifx{\__temp\empty}
% On a fini et on ne fait rien (fin du processus)
      {}
% dans le cas contraire, on va isoler la <clé> et la <valeur> du couple lu en
% prenant soin de mettre à la fin "=<quark>" pour se prémunir du cas où
% "<clé>=<valeur>" ne contient que la "<clé>" et pas de signe "=", ce qui
% ferait planter la macro à arguments délimités
      {\expandafter\hlst_find_kv\__temp=\hlst_quark\_nil[#1]%
      }%
% Lorsque la <clé> et la <valeur> est trouvée et stockée, recommencer et aller
% lire le prochain couple "<clé>=<valeur>"
    \hlst_readKV_i[#1]%
    }%
}

% Voici la macro à arguments délimités à qui on a transmis
%     <clé>=<valeur>=<quark> (si <clé>=<valeur> est l'élément lu)
% ou
%     <clé>=<quark> (si <clé> est seule)
% et qui va isoler la <clé> de la <valeur>.
\def\hlst_find_kv#1=#2\_nil[#3]{%
% #1 est ce qui se trouve avant le _premier_ signe "=" et
% #2 est ce qui se trouve entre le premier signe "=" et le \_nil
% Pour la <clé>, pas de problème, c'est _obligatoirement_ ce qui est avant le
% signe "=", que ce signe soit présent dans le couple lu ou pas puisqu'on y a
% rajouté "=<quark>".
% On élimine les espaces extrêmes pour obtenir la <clé> définitive stockée dans
% \__key (il faut 2-développer \hlst_removeextremespaces pour qu'elle donne son
% argument sans espace extrême)
  \edef\__key{\detokenize\expandafter\expandafter\expandafter{\hlst_removeextremespaces{#1}}}%
% Pour la <valeur>, on lui ôte d'abord les espaces extrêmes
  \hlst_eearg{\def\__val}{\hlst_removeextremespaces{#2}}%
  \hlst_ifx{\__val\hlst_quark}
% Si elle est égale au <quark>, alors la <valeur> vaut "true"
    {\def\__val{true}}%
% Sinon, ôter "=<quark>" de la fin de l'argument #2, éliminer les espaces
% extrêmes de ce qui en résulte et stocker le tout dans \__val (tout ceci est
% effectué par la macro \hlst_find_val <valeur>=<quark>)
    {\hlst_find_val#2}%
% Si on lit les <clés>=<valeurs> par défaut,
  \ifhlst_default
% assigner à la macro "\hlst_[<trousseau>]_<clé>" la <valeur> trouvée
    \hlst_argcsname\let{hlst_[#3]_\detokenize\expandafter{\__key}}\__val
% Puis ajouter à la macro "\hlst_[<trousseau>]", qui a été préalablement
% initialisée à <vide> :
%     \def\hlst_[<trousseau>]_<clé>{<valeur>}
    \hlst_argcsname\hlst_eaddtomacro{hlst_[#3]}%
      {\expandafter\def\csname hlst_[#3]_\detokenize\expandafter{\__key}\expandafter\endcsname\expandafter{\
          __val}}%
% C'est selon ce hashage que sont enregistrés les couples <clé>/<valeur> : les
% <clés> sont contenues dans les noms des macros tandis que les <valeurs> sont
% les textes de remplacement de ces macros.
% C'est rapide et simple :
%    a)  pour trouver une <valeur> d'après sa <clé>, il suffit de développer la
%        macro \hlst_[<trousseau>]_<clé>
%    b) pour redéfinir une <clé>, il suffit de redéfinir cette macro avec la
%       nouvelle <valeur>
%    c) il est facile de vérifier qu'une <clé> existe en vérifiant que la macro
%       associée est définie, la primitive \ifcsname le fait très bien
```

```
%    d) en revanche, on ne peut pas faire de recherche _inverse_ de façon
%       pratique : il est en effet plus difficile de trouver la (ou les) <clé>
%       contenant une <valeur> donnée, mais cette limitation n'a pas grande
%       importance ici (je ne sais pas si les autres systèmes de <clé>/<valeur>
%       sont programmés de telle sorte que cela soit simple...)
% Bref, la macro "\hlst_[<trousseau>]" contient donc _toutes_ les définitions
% des macros définissant les <clés>/<valeurs> _par défaut_ et exécuter
% "\hlst_[<trousseau>]" remet donc toutes les <clés> à leur <valeur> par défaut.
  \else
% Dans le cas où on _lit_ des nouvelles <valeurs> pour des <clés>
    \hlst_ifcsname{hlst_[#3]_\__key}
% Si la <clé> existe (ssi la macro "\hlst_[<trousseau>]_<clé>" est définie),
% alors assigner la <valeur> à cette macro
      {\hlst_argcsname\let{hlst_[#3]_\__key}\__val}%
% Sinon, émettre un message d'erreur et ne rien faire de plus
      {\errmessage{Key "\__key" is not defined: nothing is modified}}%
  \fi
}

% Cette macro à qui on a transmis "<valeur>=<quark>" ne garde que <valeur>, en
% ôte les espaces extrêmes et stocke le résultat dans \__val
\def\hlst_find_val#1=\hlst_quark{\hlst_eearg{\def\__val}{\hlst_removeextremespaces{#1}}}

% Cette macro remet toutes les <clés> à leur <valeurs> par défaut en exécutant
% la macro "\hlst_[<trousseau>]"
\def\useKVdefault[#1]{%
  \hlst_ifcsname{hlst_[#1]}
    {\csname hlst_[#1]\endcsname}
% Si la macro "\hlst_[<trousseau>]" n'existe pas, message d'erreur
    {\errmessage{Undefined set of keys "#1"}}%
}

% Cette macro donne la <valeur> correspondant à la <clé> contenue dans #2
\def\useKV[#1]#2{%
% Avec \romannumeral, la <valeur> sera obtenue après _2_ développements de
% \useKV[<trousseau>]{<clé>}
  \romannumeral\hlst_ifempty{#2}
% Si la <clé> est vide, message d'erreur (il ne peut y avoir de <valeur>
% associée à une <clé> vide)
    {\hlst_stop\errmessage{Key name missing}}
    {\hlst_ifcsname{hlst_[#1]_\hlst_removeextremespaces{#2}}
% Si la macro "\hlst_[<trousseau>]_<clé>" existe, 2-développer le \csname pour
% avoir la <valeur>
      {\expandafter\expandafter\expandafter\hlst_stop\csname hlst_[#1]_\hlst_removeextremespaces{#2}\↙
          endcsname}
% Sinon, message d'erreur
      {\hlst_stop\errmessage{Key "\hlst_removeextremespaces{#2}" not defined}}%
    }%
}

% Voici une macro purement développable qui teste si #2 (la <clé> du <trousseau>
% #1) est égale à "true" ou à "false" et selon l'issue, exécute le 1er ou 2e
% argument qui suit (arguments appelés <vrai> et <faux>)
\def\ifboolKV[#1]#2{%
% Cette macro donnera un des 2 arguments <vrai> ou <faux> en _2_ développements
% grâce au \romannumeral
  \romannumeral\hlst_ifempty{#2}
% Si la <clé> est vide, message d'erreur
    {\hlst_stop\errmessage{Key name missing}\hlst_second}
    {\hlst_ifcsname{hlst_[#1]_\hlst_removeextremespaces{#2}}
% Si la <clé> débarrassée de ses espaces extrêmes existe, tester son contenu
      {\hlst_eearg\ifboolKV_i{\csname hlst_[#1]_\hlst_removeextremespaces{#2}\endcsname}}
% Sinon, message d'erreur
      {\hlst_stop\errmessage{Key "\hlst_removeextremespaces{#2}" not defined}\hlst_second}%
    }%
}
```

```
% Cette macro teste si #1, qui est une <valeur>, vaut "true" ou "false"
\def\ifboolKV_i#1{%
% Tester d'abord si elle vaut "true"
  \hlst_ifargtrue{#1}
    {\expandafter\hlst_stop\hlst_first}
    {\hlst_ifargfalse{#1}
% Puis si elle vaut "false"
      {\expandafter\hlst_stop\hlst_second}
% Si ni l'un ni l'autre, la <valeur> n'est pas un booléen acceptable
      {\hlst_stop\errmessage{Value "#1" is not a valid boolean}\hlst_second}%
    }%
}

% La macro \hlst_ifargtrue{<argument>} teste de façon purement développable si
% <argument> vaut "true" ou "false".
% Pour cela, on transmet à \hlst_ifargtrue_i l'argument "<argument>true" qui est
% délimité par \_nil
\def\hlst_ifargtrue#1{\hlst_ifargtrue_i#1true\_nil}
% Dans la macro \hlst_ifargtrue_i, l'argument #1 est ce qui se trouve avant
% "true" dans "<argument>true" :
%     - s'il n'est pas vide, sélectionner l'argument <faux>
%     - s'il est vide, cela signifie que <argument> commence par "true" ; il est
%       donc de la forme "true<autre>"
%       L'argument #2 est ce qui se trouve après "true" dans "true<autre>true",
%       c'est donc "<autre>true".
%       Pour être sûr que <autre> est <vide>, on transmet "<autre>true" à
%       \hlst_ifargtrue_ii qui teste si la réunion de ce qui est avant le
%       premier "true" et ce qui est après est <vide>
\def\hlst_ifargtrue_i#1true#2\_nil{\hlst_ifempty{#1}{\hlst_ifargtrue_ii#2\_nil}\hlst_second}
\def\hlst_ifargtrue_ii#1true#2\_nil{\hlst_ifempty{#1#2}}

% On procède de même pour tester "false"
\def\hlst_ifargfalse#1{\hlst_ifargfalse_i#1false\_nil}
\def\hlst_ifargfalse_i#1false#2\_nil{\hlst_ifempty{#1}{\hlst_ifargfalse_ii#2\_nil}\hlst_second}
\def\hlst_ifargfalse_ii#1false#2\_nil{\hlst_ifempty{#1#2}}

%################################################
%############### macros de  hlist ###############
%################################################
% Voici les registres utilisés :
% 1) le compteur d'imbrication, incrémenté de 1 à chaque nouvelle liste
%    imbriquée :
\newcount\hlst_nest

% 2) Le compteur des \items sur chaque ligne :
\newcount\hlst_colcnt

% 3) La largeur des \hbox enfermant les <items> (un <item> est le <label> _et_
%    le <texte> de l'item) :
\newdimen\hlst_itemboxwidth

% 4) L'espace entre chaque <item> :
\newdimen\hlst_colsep

% 5) La largeur disponible pour loger une ligne d'<items> :
\newdimen\hlst_textwidth

% 6) La profondeur maximale des \vtop rencontrées jusqu'alors dans une ligne :
\newdimen\hlst_maxdepth

% 7) La profondeur du dernier élément de la \vtop la plus profonde de la ligne :
\newdimen\hlst_maxprevdepth

% 8) La boite chapeau contenant l'<item> (<label> + <texte>) :
\newbox\hlst_currenthbox
```

```
352
353 % 9) La boite contenant le <label> :
354 \newbox\hlst_labelbox
355
356 %10) La boite contenant le <texte>
357 \newbox\hlst_textbox
358
359 %11) Un booléen à fins de débogage et une dimension de débogage
360 %    Il sera mis à faux après la phase de tests et supprimé dès
361 %    la version 0.2
362 \newif\ifhlstdebugmode\hlstdebugmodefalse
363 \newdimen\hlst_debugdim
364
365 % La macro qui écrit les avertissement dans le fichier log de façon équivalente
366 % à \wlog :
367 \def\hlst_warning#1{\immediate\write-1 {Package hlist Warning: #1}}
368
369 % Cette macro renvoie le 1er caractère détokénizé de la <valeur> obtenue par
370 % 2-développement de #1, de la forme \useKV[hlist]{<clé>}
371 \def\hlst_firstcarof#1{\expandafter\hlst_firsttonil\detokenize\expandafter\expandafter\expandafter{#1}\_nil}
372
373 % Cette macro évalue la dimension contenue dans la <valeur> d'une <clé>
374 % passée dans l'argument #1 et l'assigne à la <valeur> de cette <clé>
375 \def\hlst_evaldim#1{%
376 % On n'évalue que si la <valeur> ne commence pas par "*" qui est réservé
377   \if\string*\hlst_firstcarof{\useKV[\hlstname]{#1}*}\else
378     \expandafter\edef\csname hlst_[\hlstname]_\detokenize{#1}\endcsname{\the\dimexpr\useKV[\hlstname]{#1}\↙
          relax}%
379   \fi
380 }
381
382 % Même principe pour les ressorts
383 \def\hlst_evalskip#1{\expandafter\edef\csname hlst_[\hlstname]_\detokenize{#1}\endcsname{\the\glueexpr\useKV↙
      [\hlstname]{#1}\relax}}
384
385 % Voici la macro par qui tout commence; elle peut être appelée par \hlist façon TeX
386 % ou par \begin{hlist} façon LaTeX.
387 % On teste d'abord si elle dispose d'un argument optionnel qui serait les
388 % <clé>=<valeur> applicables à cette liste
389 \def\hlist{\begingroup\hlst_ifnxttok[{\hlist_i}{\hlist_i[]}}
390 \def\hlist_i[#1]{%
391 % Si les <clé>=<valeur> ne sont pas vide, les définir dans le groupe semi simple
392 % qui vient d'être ouvert
393   \hlst_ifempty{#1}{}{\sethlist{#1}}%
394 % Fixer les dimensions et les ressorts
395   \hlst_evaldim{list parindent}\hlst_evaldim{item sep}\hlst_evaldim{left margin}\hlst_evaldim{col sep}%
396   \hlst_evaldim{item offset}\hlst_evaldim{label sep}\hlst_evalskip{pre skip}\hlst_evalskip{post skip}%
397 % Ensuite, lire un /entier/ et stocker dans \hlst_colcnt le nombre de colonnes voulu
398   \afterassignment\hlist_ii\hlst_colcnt=
399 }
400
401 \def\hlist_ii{%
402 % Incrémenter le compteur d'imbrications
403   \global\advance\hlst_nest 1
404 % Si le compteur de label correspondant à cette imbrication existe,
405   \hlst_ifcsname{c@hlist\romannumeral\hlst_nest}
406 % l'initialiser sauf si la <clé> "resume" vaut "true"
407     {\ifboolKV[\hlstname]{resume}{}{\csname c@hlist\romannumeral\hlst_nest\endcsname\hlst_stop}}
408 % S'il n'existe pas, le créer (attention, \newcount est \outer)!
409     {\csname newcount\expandafter\endcsname\csname c@hlist\romannumeral\hlst_nest\endcsname}%
410   \hlst_ifnum{\hlst_colcnt<1 }
411 % Envisager le cas où le nombre de colonnes demandé est < 1 et message d'erreur
412     {\errmessage{Invalid column number "\the\hlst_colcnt", 1 column used instead}\def\hlst_nbcol{1}}
413 % La macro \hlst_nbcol contient finalement le nombre de colonnes
414     {\edef\hlst_nbcol{\the\hlst_colcnt}}%
415 % Si on doit afficher les labels
```

```
  \ifboolKV[\hlstname]{show label}
% Calculer combien de dimensions explicites sont demandées par l'utilisateur
    {\edef\hlst_implicitdim{\the\numexpr
      \if\string*\hlst_firstcarof{\useKV[\hlstname]{item offset}*}1\else0\fi+
      \if\string*\hlst_firstcarof{\useKV[\hlstname]{label width}*}1\else0\fi+
      \if\string*\hlst_firstcarof{\useKV[\hlstname]{label sep}*}1\else0\fi\relax
      }%
% Si ce nombre est au moins 1
    \ifnum\hlst_implicitdim>0
      \hlst_ifnum{\hlst_implicitdim>1 }
% S'il est plus grand que 1, message d'erreur car parmi les 3 dimensions, _une_
% seule peut être implicite
        {\errmessage{Too many implicit dimensions for labels, default settings used instead}%
        \sethlist{item offset=1.5em,label sep=0.25em,label width=1.25em}%
        }
% Si ce nombre est égal à 1, calculer la dimension implicite en fonction des 2
% autres dimensions explicites...
        {\if\string*\hlst_firstcarof{\useKV[\hlstname]{item offset}*}%
          \edef\hlst_implicitdim{[\hlstname]{item offset=\the\dimexpr\useKV[\hlstname]{label sep}+\useKV[\hlstname]{label width}\relax}}%
        \else\if\string*\hlst_firstcarof{\useKV[\hlstname]{label sep}*}%
          \edef\hlst_implicitdim{[\hlstname]{label sep=\the\dimexpr\useKV[\hlstname]{item offset}-\useKV[\hlstname]{label width}\relax}}%
        \else\if\string*\hlst_firstcarof{\useKV[\hlstname]{label width}*}%
          \edef\hlst_implicitdim{[\hlstname]{label width=\the\dimexpr\useKV[\hlstname]{item offset}-\useKV[\hlstname]{label sep}\relax}}%
        \fi\fi\fi
% ...et la sauvegarder dans le <trousseau> "hlist" via la <clé> correspondante
        \expandafter\setKV\hlst_implicitdim
        }%
    \fi
% Vérifier que "item offset" >= label sep + label width et dans le cas
% contraire, avertir
    \ifdim\useKV[\hlstname]{item offset}<\dimexpr\useKV[\hlstname]{label sep}+\useKV[\hlstname]{label width}\relax
      \hlst_warning{incompatibles dimensions: item offset (\the\dimexpr\useKV[\hlstname]{item offset}\relax) < label sep (\the\dimexpr\useKV[\hlstname]{label sep}\relax) + label width (\the\dimexpr\useKV[\hlstname]{label width}\relax)}%
    \fi
    }
    {}%
% Si l'imbrication est 1, on est donc dans la hlist de plus haut niveau
  \ifnum\hlst_nest=1 % si liste mère
% Régler la parindent à la <valeur> de la <clé> "left margin"
    \parindent\dimexpr\useKV[\hlstname]{left margin}\relax
% et insérer le ressort "pre skip"
    \ifhmode\par\fi\vskip\useKV[\hlstname]{pre skip}%
  \else
% Sinon, pas d'indentation pour les listes imbriquées
    \parindent0pt
  \fi
% Sauvegarder la largeur actuelle horizontale
  \hlst_textwidth=\dimexpr\ifdefined\linewidth\linewidth\else\hsize\fi-\parindent\relax
% Informations de débogage sur les dimensions des composants
  \ifhlstdebugmode
    \message{*** hlist mode debug : largeur = \the\hlst_textwidth}%
    \hlst_debugdim=0pt \def\hlst_debuglineno{0}%
    \edef\hlst_debuglineno{\number\numexpr\hlst_debuglineno+1}%
    \def\debuginfoB{\space\space Ligne d'items no \hlst_debuglineno\space:^^J}%
  \fi
  \leavevmode
% Initialiser le compteur de colonnes \hlst_colcnt à 0
  \hlst_colcnt0
% Sauvegarder dans \hlst_colsep la <valeur> de la <clé> "col sep"
  \hlst_colsep\useKV[\hlstname]{col sep}\relax
% Au début de la liste : pas de multicolonne
```

```
  \def\hlst_askedmulticol{1}%
% Calculer la largeur de chaque <item> (=<label>+<texte>) s'il est non multicol
  \edef\hlst_colwidth{\the\dimexpr(\hlst_textwidth-\useKV[\hlstname]{col sep}*(\hlst_nbcol-1))/(\hlst_nbcol)↙
      }%
% Collecter dans une boite tout ce qui se trouve jusqu'au premier \hitem
% (neutraliser l'espace, le tab et le retour charriot)
  \setbox0\hbox\bgroup\catcode`\^^I 9 \catcode`\^^M 9 \catcode32 9
}

% Voici maintenant la macro qui ferme l'environnement. Elle aussi peut être
% à la TeX par \endhlist ou par \end{hlist} comme on le fait en LaTeX.
\def\endhlist{%
% En premier lieu, il faut exécuter la routine qui achève un \hitem (fermeture
% des boites, calcul de la \vtop la plus profonde et de son \prevdepth, etc)
  \hlst_enditem
% Débogage
  \ifhlstdebugmode
    \message{\debuginfoB\space\space\space\space TOTAL = \the\hlst_debugdim/\the\hlst_textwidth^^J
      \space\space\space\space Depassement = \number\numexpr\hlst_debugdim-\hlst_textwidth\relax sp^^J
      *** Fin de la liste^^J^^J}%
    \ifnum\numexpr\hlst_debugdim-\hlst_textwidth\relax>4 \errmessage{Depassement !}\fi
  \fi
% Si la liste est non imbriquée, insérer le ressort "post skip"
  \ifnum\hlst_nest=1 \vskip\useKV[\hlstname]{post skip}\fi
% Décrémenter le compteur d'imbrications
  \global\advance\hlst_nest -1
% Fermer le groupe semi simple ouvert avec \hlist et manger les espaces
  \endgroup
  \ignorespaces
}

% Voici à présent des macros traduisant un <nombre> en un <label>.
% La première, qui est appelée par toutes les autres, converti le <nombre> en un
% nombre arabe.
\def\hlst_arabic#1{%
% Si le compteur (au modèle latex, c'est-à-dire "\c@<nom>") existe
  \hlst_ifcsname{c@\detokenize{#1}}
% Donner le nombre arabe avec \number, sinon, message d'erreur
    {\number\csname c@\detokenize{#1}\endcsname}
    {\errmessage{Unkonwn counter named "\detokenize{#1}"}}%
}
% Ensuite, ces macros convertissent un <nombre> en nombre romain ou en lettre,
% avec choix de la casse
\def\hlst_roman#1{\romannumeral\hlst_arabic{#1}\relax}
\def\hlst_Roman#1{\uppercase\expandafter\expandafter\expandafter{\hlst_roman{#1}}}
\def\hlst_alpha#1{\ifcase\hlst_arabic{#1}\relax\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
  k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or
  y\or z\else\errmessage{No letter for a value of \number\numexpr#1\relax}\fi}
\def\hlst_Alpha#1{\ifcase\hlst_arabic{#1}\relax\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
  K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or
  Y\or Z\else\errmessage{No letter for a value of \number\numexpr#1\relax}\fi}

% Voici maintenant la macro \hitem, qui marque le début d'un nouvel <item>
% Tout d'abord, on s'intéresse à ce qui suit, c'est-à-dire aux argument
% optionnels.
% On teste si elle est suivie d'une "*" (qui commande un retour à la ligne pour
% cet item) et on transmet le token 0 ou 1 selon l'issue du test
\def\hitem{\hlst_ifstar{\hlst_hitem_grabopts1}{\hlst_hitem_grabopts0}}

% Ensuite, on teste si elle est suivie de ">" (qui commande un multicol maximal
% pour cet item) et on transmet le token 0 ou 1  selon l'issue du test
\def\hlst_hitem_grabopts#1{\hlst_ifnxttok>{\hlst_first{\hlst_hitem_grabopts_i#11}}{\hlst_hitem_grabopts_i↙
    #10}}

% La macro \hlst_hitem_grabopts_i reçoit les 2 premiers arguments concernant la
% présence ou pas de "*" et de ">" ; ces 2 tokens seront transmis de proche en
```

```
% proche jusqu'à la la macro \hitem_ii qui est la macro qui fait le boulot.
\def\hlst_hitem_grabopts_i#1#2{%
% On teste si une parenthèse (pour demander un multicol) est le prochain token
  \hlst_ifnxttok(
% Si oui, aller lire ce qui est à l'intérieur avec \hlst_hitem_grabopts_ii
    {\hlst_hitem_grabopts_ii#1#2}
% Sinon, on teste si un crochet (pour forcer un label pour cet item) est le
% prochain token
    {\hlst_ifnxttok[
% Si oui, aller lire ce qu'il contient
      {\hlst_hitem_grabopts_iii#1#2}
% Si ni parenthèse ni crochet, transmettre "()[<quark>]" à \hitem_i
      {\hitem_i#1#2()[\hlst_quark]}%
    }%
}

% Ici, une parenthèse a été lue en premier, tester si la suite est un crochet
\def\hlst_hitem_grabopts_ii#1#2(#3){\hlst_ifnxttok[{\hitem_i#1#2(#3)}{\hitem_i#1#2(#3)[\hlst_quark]}}

% Ici, un crochet a été lu en premier, tester si la suite est une parenthèse
\def\hlst_hitem_grabopts_iii#1#2[#3]{\hlst_ifnxttok({\hlst_hitem_grabopts_iv#1#2[#3]}{\hitem_i#1#2()[#3]}}

% Au cas où le crochet est lu en premier, remettre les argument optionnels dans
% le bon ordre : parenthèse _puis_ crochet
\def\hlst_hitem_grabopts_iv#1#2[#3](#4){\hitem_i#1#2(#4)[#3]}

% Cette macro mange tous les espaces _explicites_ qui se trouvent éventuellement
% après les arguments optionnels
% Pour résumer, \hitem peut être suivi de "*" _puis_ ">" (ordre imposé)
% Ensuite on peut trouver (<multicol>) et/ou [<label>] dans n'importe quel ordre
\def\hitem_i#1#2(#3)[#4]{%
  \hlst_ifnxttok\hlst_sptok
    {\def\__temp{\hitem_i#1#2(#3)[#4]}\expandafter\__temp\hlst_gobspace}
    {\hitem_ii#1#2(#3)[#4]}%
}

% Voici la /vraie/ macro qui va faire le boulot et ses arguments :
%     - #1 et #2 valent 0 ou 1 selon la présence de "*" et ">"
%     - #3 est le nombre de multicolonnes demandé (si vide, comprendre 1)
%     - #4 est le label demandé (si <quark>, comprendre absence d'argument
%       optionnel)
\def\hitem_ii#1#2(#3)[#4]{%
% si ce n'est _pas_ le 1er de la liste
  \hlst_ifnum{\hlst_colcnt>0 }
% Exécuter la routine de fin d'item
    {\hlst_enditem
    }
% Sinon, fermer la boite ouverte par \hlist
    {\egroup
% Si toutes ses dimensions ne sont pas nulles, message d'erreur
    \ifnum0\ifnum\wd0=0 1\fi\ifnum\dp0=0 1\fi\ifnum\ht0=0 1\fi<111
      \errmessage{\string\hitem\space expected}%
    \fi
% comme il s'agit du 1er item de la liste, initialiser les profondeurs
    \hlst_maxdepth0pt \hlst_maxprevdepth0pt
    }%
% Incrémenter le compteur de label associé à cette imbrication
  \hlst_argcsname\global\advance{c@hlist\romannumeral\hlst_nest} 1
% Incrémenter le compteur de colonnes du nombre de colonnes sautées précédemment
  \advance\hlst_colcnt\hlst_askedmulticol\relax
% Si "*", mettre le compteur de colonnes au maximum pour forcer retour à la
% ligne
  \ifnum#1=1 \hlst_colcnt\numexpr\hlst_nbcol+1\relax\fi
  \ifnum\hlst_colcnt>1
% Si ce n'est pas le 1er item de la liste
    \hlst_ifnum{\hlst_colcnt=\numexpr\hlst_nbcol+1\relax}
```

```
% Si le nombre d'item sur la ligne a atteint son maximum,
% former le paragraphe, sauter l'espace verticale "item sep"
      {\par\vskip\useKV[\hlstname]{item sep}\relax
% Débogage
      \ifhlstdebugmode
        \message{\debuginfoB\space\space\space\space TOTAL = \the\hlst_debugdim/\the\hlst_textwidth^^J
          \space\space\space\space Depassement = \number\numexpr\hlst_debugdim-\hlst_textwidth\relax sp^^J}%
        \ifnum\numexpr\hlst_debugdim-\hlst_textwidth\relax>\hlst_colcnt \errmessage{Depassement !}\fi% ↙
             dépassement si plus que <nb colonnes>sp
        \edef\hlst_debuglineno{\number\numexpr\hlst_debuglineno+1}%
        \def\debuginfoB{\space\space Ligne d'items no \hlst_debuglineno\space:^^J}%
        \hlst_debugdim0pt
      \fi
% Tromper TeX : ne pas tenir compte des profondeurs des \vtop, mais utiliser la
% profondeur du dernier élément de la \vtop la plus profonde
      \prevdepth\hlst_maxprevdepth
% Passer en mode horizontal, initialiser les variables pour commencer une
% nouvelle ligne d'items
      \leavevmode
      \hlst_maxdepth0pt \hlst_maxprevdepth0pt \hlst_colcnt=1
      }
% S'il reste des items à loger sur la ligne, insérer espace inter-item
      {\kern\hlst_colsep\relax
% Débogage
      \ifhlstdebugmode
        \edef\debuginfoB{\debuginfoB\space\space\space\space colsep (\the\hlst_colsep)^^J}%
        \advance\hlst_debugdim\hlst_colsep
      \fi
      }%
  \fi
% Calculer le nombre de multicolonne pour l'item courant
  \edef\hlst_askedmulticol{\number\numexpr
    \hlst_ifnum{#2=1 }
% Si ">", prendre le maximum disponible
      {\hlst_nbcol+1-\hlst_colcnt}
% Sinon, prendre l'argument #3 (s'il est vide ce qui correspond à une absence
% d'argument optionnel, prendre 1)
      {\hlst_ifempty{#3}{1}{(#3)}}%
    \relax}%
  \hlst_ifnum{\hlst_askedmulticol<1 }
% Si l'utilisateur est un joueur ou un abruti !
    {\errmessage{Invalid multicol "(#3)", "(1)" used instead}\def\hlst_askedmulticol{1}}
% Si trop de multicol demandé
    {\hlst_ifnum{\numexpr\hlst_colcnt+\hlst_askedmulticol-1\relax>\hlst_nbcol\relax}
% Prendre le maximum et avertir
      {\edef\hlst_askedmulticol{\number\numexpr\hlst_nbcol+1-\hlst_colcnt\relax}%
      \hlst_warning{too much multicol (#3) asked, (\hlst_askedmulticol) used instead}}%
      {}%
    }%
% Calculer la largeur de la \hbox contenant l'<items>
  \hlst_itemboxwidth\dimexpr\hlst_colwidth*\hlst_askedmulticol+\useKV[\hlstname]{col sep}*(\↙
       hlst_askedmulticol-1)\relax
% Définir la boite  contenant <label> + <texte>
  \setbox\hlst_currenthbox\hbox to \hlst_itemboxwidth\bgroup
% Initialiser le débordement du <label>
    \xdef\hlst_labeloverwidth{0pt}%
% Si on affiche les labels ET si #4 n'est pas vide (un label explicitement
% demandé "[]" doit être compris comme "ne pas afficher de label du tout pour
% cet item").
    \hlst_ifnum{\ifboolKV[\hlstname]{show label}10\hlst_ifempty{#4}01=11 }%
% Ouvrir la boite des labels
      {\hbox to \useKV[\hlstname]{item offset}{%
% Pousser à droite au cas où "item offset">"label sep"+"label width"
        \hfill
% Ouvrir la \hbox contenant le label et forcer sa dimension horizontale à la
% <valeur> "label width"
```

```
        \hbox to \useKV[\hlstname]{label width}{%
% Trouver la première lettre de la <valeur> "label align", sachant qu'une
% consigne vide sera comprise comme un alignement "l"
          \edef\hlst_align{\hlst_firstcarof{\useKV[\hlstname]{label align}l}}%
% Pousser à droite si cette lettre est "r" ou "c"
          \if      \string r\hlst_align\hfill
          \else\if \string c\hlst_align\hfill
          \fi\fi
% Localement, (re)définir les macros de formatage pour le label afin qu'elles
% deviennent accessible à l'utilisateur
          \let\arabic\hlst_arabic\let\roman\hlst_roman\let\Roman\hlst_Roman\let\alpha\hlst_alpha\let\Alpha\↙
              hlst_Alpha
% Mettre dans la boite \hlst_labelbox tout ce qui concerne le label (les codes
% avant/après et le label formaté selon la <valeur> "label")
          \setbox\hlst_labelbox\hbox{%
            \useKV[\hlstname]{pre label}%
            \hlst_ifx{\hlst_quark#4}{\useKV[\hlstname]{label}}{#4}%
            \useKV[\hlstname]{post label}%
          }%
% Mesurer le dépassement (signé) du label par rapport à sa dimension autorisée
% "label width"
          \xdef\hlst_labeloverwidth{\the\dimexpr\wd\hlst_labelbox-\useKV[\hlstname]{label width}\relax}%
% Afficher la boite contenant le label avec ou sans cadre
          \ifboolKV[\hlstname]{show frame}
            {\fboxrule.4pt  \fboxsep-\fboxrule \fbox{\box\hlst_labelbox}}
            {\box\hlst_labelbox}%
% Si on gère le dépassement de largeur
          \ifboolKV[\hlstname]{autoindent}
% Mettre le ressort \hss pour neutraliser l'avertissement de "overful hbox"
            {\ifdim\hlst_labeloverwidth>0pt \hss\fi}
            {}%
% Pousser à gauche si l'alignement demandé est "l" ou "c"
          \if      \string l\hlst_align\hfill
          \else\if \string c\hlst_align\hfill
          \fi\fi
          }%
% Séparation horizontale entre boite de <label> et boite de <texte>
        \kern\useKV[\hlstname]{label sep}\relax
        }%
% Diminuer la largeur disponible de la largeur de la boite de <label>, soit la
% <valeur> "item offset"
      \advance\hlst_itemboxwidth-\useKV[\hlstname]{item offset}\relax
      }
      {}%
% Maintenant que la boite du <label> est affichée (ou pas si c'est le choix de
% l'utilisateur), il faut passer à la boite qui contient le <texte> du label.
% Bien entendu, on utilise des accolades _implicites_ puisque la fin de la boite
% ne se situe pas dans cette macro.
% Cette boite sera une \vtop...
    \setbox\hlst_textbox\vtop\bgroup
% ...de largeur \hlst_itemboxwidth...
      \hsize\hlst_itemboxwidth
% ...que l'on copie vers \linewidth pour les utilisateurs de LaTeX, afin que
% cette macro garde sa cohérence
      \ifdefined\linewidth\linewidth\hlst_itemboxwidth\fi\relax
% Réglage du parindent à "list parindent"
      \parindent\useKV[\hlstname]{list parindent}\relax
% Passage en mode H et pas d'indentation pour le 1er paragraphe si on
% affiche les labels
      \ifboolKV[\hlstname]{show label}\noindent\indent
% Après être passé en mode horizontal, on insère l'indentation supplémentaire en
% cas de débordement de la largeur du <label>
      \ifboolKV[\hlstname]{autoindent}
        {\ifdim\hlst_labeloverwidth>0pt \kern\hlst_labeloverwidth\fi}
        {}%
% Puis, on exécute le code inséré juste avant le <texte> de l'item
```

```
      \useKV[\hlstname]{pre item}%
}
%            _____________________________________________
%           |                                             |
%           |   <texte> de l'item entre ces 2 macros      |
%           |_____________________________________________|
%
% Une fois que le <texte> de l'item est /librement/ composé (librement, cela
% autorise les changements de catcode, le "verbatim", etc), la macro de fin du
% <texte> est exécutée.
\def\hlst_enditem{%
% D'abord, exécuter le code de fin d'item
      \useKV[\hlstname]{post item}%
% Ensuite, former le paragraphe pour accéder à \prevdepth
      \par
% et développer "\the\prevdepth" _avant_ de fermer la \vtop et la \hbox (le
% double \egroup) pour que cette dimension soit le texte de remplacement de
% \hlst_currentdepth
      \hlst_exparg{%
% fermeture de la \vtop
      \egroup
      \ifboolKV[\hlstname]{show frame}{\fboxrule.4pt  \fboxsep-.4pt \fbox{\box\hlst_textbox}}{\box\↙
          hlst_textbox}%
    \egroup
    \def\hlst_currentdepth}{\the\prevdepth}%
% Si la profondeur de la \vtop qui vient d?être créé est supérieure à toutes
% celles rencontrées dans cette ligne,
    \ifdim\dp\hlst_currenthbox>\hlst_maxdepth
% retenir la valeur de \prevdepth
      \hlst_maxprevdepth\hlst_currentdepth\relax
% et mettre à jour la profondeur maximale des \vtop de la ligne en cours.
% Le but de cette manoeuvre est de trouver la profondeur du dernier élément de
% la plus profonde \vtop de la ligne en cours.
% Cette valeur de \prevdepth sera utilisée au prochain \par pour tromper
% TeX qui sinon, prendrait en compte la /grande/ profondeur des \vtop, menant à
% des lignes trop proches.
      \hlst_maxdepth\dp\hlst_currenthbox
    \fi
% Une fois tout ceci fait, afficher la boite contenant le <texte>
  \box\hlst_currenthbox
% Débogage
  \ifhlstdebugmode
    \edef\debuginfoB{\debuginfoB\space\space\space\space item (\the\hlst_itemboxwidth)^^J}%
    \advance\hlst_debugdim\hlst_itemboxwidth
  \fi
}%

\hlst_restorecatcode
% Création, si nécessaire d'une macro \fbox, équivalente à celle de LaTeX
\unless\ifdefined\fbox
% Prendre les réglages par défaut identiques à ceux de laTeX
  \newdimen\fboxrule \newdimen\fboxsep \fboxrule=.4pt \fboxsep=3pt
% Pour les explications, voir codes 251 \‘a 254 de ce fichier :
%      http://progtex.fr/wp-content/uploads/2014/09/code.txt
% et pages 271 \‘a 274 de "Apprendre à programmer en TeX"
  \def\fbox#1{%
    \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

```

```
% Les réglages par défaut (modifiables avec \setdefaulthlist)
\setKVdefault[\hlstname]{
  pre skip       = \medskipamount,% ressort vertical avant la liste mère
  post skip      = \medskipamount,% ressort vertical après la liste mère
  left margin    = 0pt,% décalage à droite des premières boites de chaque ligne de la liste mère
  col sep        = 2.5em,% espace de séparation entre colonnes
  item offset    = 1.75em,% distance entre bord gauche du label et bord gauche de l'item
  label sep      = 0.25em,% recul du label
  label width    = *,% largeur du label automatique (=1.75-0.25 em) car "item ofset" = "label width" + "label sep"
  label align    = left,% consigne d'alignement
  show label     = true,% afficher les numérotations et tenir compte de leur largeur
  pre label      = \bf,% code exécuté au début de la boite contenant le label
  label          = \arabic{hlisti}.,% code du label
  post label     = {},% code exécuté après le label
  item sep       = 0pt,% espace vertical supplémentaire entre lignes d'items
  list parindent= 0pt,% indentation des paragraphes des items (sauf le 1er)
  pre item       = {},% code exécuté avant le texte de l'item
  post item      = {},% code exécuté après le texte de l'item
  resume         = false,% pas de reprise de numérotation
  autoindent     = false,% pas d'indentation auto en cas de label trop large
  show frame     = false% ne pas afficher les cadres autour des boites <label> et <texte>
}

% La macro \inithlist remet toutes les <clés> à leur <valeur> par défaut
\expandafter\let\expandafter\inithlist\csname hlst_[\hlstname]\endcsname

% Macros permettant de modifier les <valeurs> des <clés> de hlist...
\def\sethlist#{\setKV[\hlstname]}

% ... ainsi que les <valeurs> par défaut
\def\setdefaulthlist#{\setKVdefault[\hlstname]}
\endinput

Versions :
 ____________________________________________________________________________
| Version |    Date    | Changements                                         |
|----------------------------------------------------------------------------|
|   0.1   | 08/05/2017 | Première version                                    |
|----------------------------------------------------------------------------|
```