Anpassen von Listen mit dem Paket enumitem*

Javier Bezos

Version 3.5.2
2011-09-28

Für Anregungen, um Kommentare und um Fehler zu melden, besuchen Sie `http://www.tex-tipografia.com/enumitem.html`.

(English is not my strong point, so contact me when you find mistakes in the manual.)

Weitere Pakete des Autors: gloss (mit José Luis Díaz), accents, tensind, esindex, dotlessi, titlesec, titletoc.

1 Einführende Bemerkungen

Als ich vor einigen Jahren begann LaTeX zu nutzen, gab es zwei Punkte, die ich als ganz besonders ärgerlich empfand: Kopf- bzw. Fußzeilen und Listen, denn es war sehr kompliziert, sie zu modifizieren. Erstere lassen sich mittels meines titlesec-Paket anpassen, aber für Letztere gab es bislang nur folgende Möglichkeiten:

- enumerate, ein Paket, welches es nur erlaubt, das Label einer Liste zu ändern.
- mdwlist, welches nach der zugehörigen Dokumentation lediglich einige eventuell nützliche listenbezogene Befehle und Umgebungen zur Verfügung stellt, aber keine in sich geschlossene Methode ist, mit Listen zu arbeiten.
- paralist, welches es erlaubt, Listen innerhalb eines Paragraphen anzuwenden (der eigentliche Zweck dieses Pakets) und einige sehr spezifische Änderungen und das optionale Argument von enumerate zur Verfügung stellt.

Einer der größten Schwachpunkte des Standardpakets list ist, dass die Bedeutung seiner Parameter nicht immer klar ersichtlich ist. Um eine benutzerfreundlichere Bedienung zu ermöglichen, gab es zwei gangbare Wege: Entweder neue Listen definieren oder eine neue Syntax etablieren, die es einfacher macht, die Standardlisten anzupassen. Für Marks habe ich in titlesec den ersten Ansatz gewählt, einfach weil ich keine befriedigende Lösung für dieses Problem mit den LaTeX-internen Makros gefunden habe, hier habe ich jedoch auf den zweiten Ansatz zurückgegriffen.

Das Paket verwendet eine Art „Vererbung“. Das Verhalten von Listen lässt sich global einstellen, Parameter eines bestimmten Pakets wie z. B. enumerate können von diesen Einstellungen aber abweichen, gleiches gilt auch für Parameter einer bestimmten Listenebene. Die jeweiligen Werte ergeben sich aus der Hierarchie.

2 Das Paket

Dieses Paket ist dafür gedacht, die Anpassung der drei Basislistenumgebungen enumerate,itemize und description an eigene Bedürfnisse zu erleichtern. Es erweitert ihre Syntax

*Version 3.3 **übersetzt von Matthias Ludwig (FSU Jena)**, auf die aktuelle Version 3.5.2 angepasst von Christine Römer.

dahingehend, dass sie ein optionales Argument erlauben, welches der Form Schlüssel=Wert entspricht.

- Vertikale Abstände
  - topsep
  - partopsep
  - parsep
  - itemsep
- Horizontale Abstände
  - leftmargin
  - rightmargin
  - listparindent
  - labelwidth
  - labelsep
  - itemindent

Zum Beispiel:

```
\begin{itemize}[itemsep=1ex,leftmargin=1cm]
```

Die oben aufgeführten Schlüssel sind äquivalent zu den bekannten Listen-Parametern, weitere Informationen zu diesen finden sich in einem LaTeX-Handbuch. Der nächste Abschnitt widmet sich den Erweiterungen, die enumitem bereitstellt.

3 Schlüssel

Dieser Abschnitt beschreibt Schlüssel in angezeigten Listen. Die meisten von ihnen sind in Inline-Listen[1] verfügbar, wo aber auch weitere Schlüssel verwendet werden können (siehe 4).

3.1 Label- und Querverweisformat

`label=`⟨*commands*⟩

Dieser Befehl setzt eine Markierung, die auf der aktuellen Ebene verwendet werden kann. Das Setzen einer gesternten Version von `\alph`, `\Alph`,`\arabic`,`\roman` und `\Roman` ohne Argument steht für den aktuellen Zähler von enumerate.[2] Demnach gibt

```
\begin{enumerate}[label=\emph{\alph*}]
```

a), *b)* usw. aus (dies ist der Standardstil in Spanisch und wurde ehemals auch von Chicago verwendet). Selbiges funktioniert auch mit `\value` (vorrausgesetzt, das weiteste Label soll nicht verarbeitet werden oder `widest*` wird benutzt, siehe unten). Ein ausgefalleneres Beispiel (welches hässlich aussieht, aber demonstrieren soll, was theoretisch möglich ist; benötigt `color` und `pifont`):

```
\begin{enumerate}[label=\protect\fcolorbox{blue}{yellow}
{\protect\ding{\value*}}, start=172]
```

[1] Anm. d. Ü.: Bei Inline-Listen handelt es sich um Listen, die im Fließtext erscheinen, jedoch ist „Fließtextliste“ kein konventionalisierter Ausdruck, weshalb hier der englische Begriff beibehalten wird.

[2] Präzise ausgedrückt, der Asterisk ist das Argument, aber das könnte sich ändern. Deshalb sollte man sie einfach als gesternte Variante betrachten und die zugehörige Syntax verwenden.

Der Wert von `label` ist ein bewegliches Argument und empfindliche Befehle müssen geschützt werden, was aber nicht für die Zähler gilt. Deswegen ist der Gebrauch von `value` einigermaßen problematisch, da `\the` oder `\ifnum` einen konkreten Wert verlangen, was aber nicht der Fall ist, wenn `label` verwendet wird, um intern * mit dem Zählerargument zu ersetzen. Die in der Regel beste Lösung für dieses Problem ist es, mit Hilfe von `\AddEnumerateCounter` die Logik in einem neuen Zähler einzukapseln.[3] Wenn man die Art Label zu setzen bevorzugt, die im Paket `enumerate` verwendet wird, kann man „Kurz-Label“benutzen (siehe Abschnitt 6).

`label*=`⟨*commands*⟩

Dies funktioniert ähnlich wie `label`, nur das dessen Wert dem übergeordneten Label hintenangestellt ist. So definiert beispielsweise der folgende Quelltext eine `legal`-Liste (1., 1.1, 1.1.1 usw.):

```
\newlist{legal}{enumerate}{10}
\setlist[legal]{label*=\arabic*.}
```

`ref=`⟨*commands*⟩

In der Default-Einstellung legt `label` auch die Form der Querverweise und von `the...` fest, wobei Letzteres die Einstellungen der vorhergehenden Hierarchie-Ebenen überschreibt, aber mit diesem Schlüssel kann man ein abweichendes Format definieren. Das folgende Beispiel zeigt, wie man die rechte Parenthese entfernen kann:

```
\begin{enumerate}[label=\emph{\alph*}),ref=\emph{\alph*}]
```

Sowohl bei `label` als auch bei `ref` können die Zähler wie gewohnt verwendet werden:

```
\begin{enumerate}[label=\theenumi.\arabic*.]
```

oder

```
\begin{enumerate}[label=\arabic{enumi}.\arabic*.]
```

(vorrausgesetzt, man befindet sich auf der zweiten Gliederungsebene).

Allerdings ist zu beachten, dass sich die Verweise nicht aus den Labeln der einzelnen Gliederungsebenen zusammensetzen. Möchte man beispielsweise mit 1.*a* auf einen Punkt in einer Liste verweisen, deren Gliederungsebenen mit 1) (erste Ebene) und *a)* (zweite Ebene) beschriftet sind, muss man dies über `ref` einstellen, beispielsweise mit `\ref{level1}.\ref{level2}` und den dazu passenden `ref`-Einstellungen. Jedoch stellte schon Robin Fairbairns in der TEX-FAQ fest:

> ... [that] would be both tedious and error-prone. What is more, it would be undesirable, since you would be constructing a visual representation which is inflexible (you could not change all the references to elements of a list at one fell swoop).[4]

Es wird empfohlen diesem Rat zu folgen, aber manchmal möchte man etwas wie das Folgende:

```
...subitem \ref{level2} of item \ref{level1} ...
```

Der Wert von `ref` ist ein bewegliches Argument und empfindliche Befehle müssen geschützt werden, was aber *nicht* für die Zähler gilt.

[3] Dies ist zugegebenermaßen ziemlich umständlich. An einer besseren Lösung wird gearbeitet.

[4] Beides wäre mühsam und fehleranfällig. Obendrein wäre es auch nicht wünschenswert, da man auf diesem Wege eine visuelle Repräsentation erstellt, die nicht flexibel ist (man kann nicht alle Verweise auf Elemente einer Liste mit einem gefühlten Wisch verändern).

`font=`⟨*commands*⟩ `format=`⟨*commands*⟩

Dies setzt „label font“. Das ist nützlich, wenn das Label mit `\item` oder in description verändert wird. Der letzte Befehl in ⟨*commands*⟩ kann ein optionales Argument mit dem Label des Items zu sich nehmen. In description sind die Klassen-Einstellungen aktiv, demnach kann es nützlich sein, mit `\normalfont` zu beginnen. Ein synonymer Befehl ist `format`.

`align=left` `align=right` `align=parleft` NEW 3.0

Dies legt die Ausrichtung des Labels in Relation zu den Labelbox-Kanten fest. Drei Werte sind möglich: `left`, der Default-Wert `right` und `parleft` (eine Parbox der Breite `\labelwidth` mit Text im linksbündigen Flattersatz). Die Parameter, die die Ausrichtung des Labels bestimmen, sollten richtig gesetzt werden, entweder per Hand oder bequemer mit der Einstellung * (siehe unten):

```
\begin{enumerate}[label=\Roman*., align=left, leftmargin=*]
```

(Wenn die Labelbox die ursprüngliche Breite haben soll, muss `left` verwendet werden.)

`\SetLabelAlign{`⟨*value*⟩`}{`⟨*commands*⟩`}` NEW 3.0

Neue Ausrichtungen können über `\SetLabelAlign` definiert und vorhandene geändert werden; die vordefinierten Werte sind äquivalent zu:[5]

```
\SetLabelAlign{right}{\hss\llap{#1}}
\SetLabelAlign{left}{#1\hfil}
\SetLabelAlign{parleft}{\strut\smash{\parbox[t]\labelwidth{\raggedright##1}}}
```

Wenn der letzte Punkt in der Definition ein Zeilenumbruch ist (typischerweise `\hfil`), wird er manchmal von description entfernt. Wenn man dies aus irgendwelchen Gründen verhindern will, muss am Ende lediglich `\null` angefügt werden.

Obwohl primär für die Ausrichtung bestimmt, sind diese Befehle anders zu verwenden (wie das mitgelieferte `parleft`). Beispielsweise mit dem folgenden `align=right` werden alle Labels hochgestellt gesetzt:

```
\SetLabelAlign{right}{\hss\llap{\textsuperscript{#1}}}
```

(Ein neuer Name ist ntürlich auch möglich.)

Wenn Sie möchten, dass die internen Einstellungen für `align` und `font` ignoriert werden, können Sie die enumitem-Definition für `\makelabel` in `zuvor` überschreiben:

```
\begin{description}[before={\renewcommand\makelabel[1]{\ref{##1}}}]
```

(Alternativ können Sie ein Makro definieren und `\let` nehmen.)

3.2 Horizontale Ausrichtung von Labeln

`labelindent=`⟨*length*⟩
`\labelindent`

Dieser Parameter sorgt in enumitem für einen Leerraum zwischen Liste/Text und dem linken Rand der Labelbox. Dies bedeutet eine gewisse Redundanz, da ein Parameter auf den anderen aufbaut, also auf Basis anderer Werte errechnet werden muss, wie unten beschrieben. Es gibt eine neue Zählerlänge `\labelindent`, deren Defaultwert bei 0pt liegt. Die fünf Parameter verhalten sich untereinander wie folgt:

$$\texttt{\textbackslash leftmargin} + \texttt{\textbackslash itemindent} = \texttt{\textbackslash labelindent} + \texttt{\textbackslash labelwidth} + \texttt{\textbackslash labelsep}$$

[5] Vor Version 3.0 war die linksbündige Ausrichtung fehlerhaft und Text und Label konnten sich überlappen.

`leftmargin=!` `itemindent=!` `labelsep=!` `labelwidth=!` `labelindent=!` NEW 3.0

Legt fest, welcher Wert von den anderen errechnet wird. Dies geschieht, nachdem *alle* Werte gelesen wurden. Explizit festgelegte Werte gehen mit den folgenden Einstellungen nicht verloren:

```
leftmargin=2em
labelindent=1em,leftmargin=!
labelindent=!
```

`leftmargin` beträgt 2em und `\labelindent` ist der zu errechnende Wert. Die Default-Einstellung ist `\labelindent=!`. Es ist jedoch zu beachten, dass einige Schlüssel einen anderen Wert setzen (`wide` und description-Stiles). Eine Einstellung mit den Werten `align=right` (der Default), `labelindent=!` und `labelwidth=!` verhält sich in der Praxis ähnlich.

`leftmargin=*` `itemindent=*` `labelsep=*` `labelwidth=*` `labelindent=*`

Wie vorher, nur dass `\labelwidth` auf die Breite des aktuellen Labels angepasst wird. Dabei wird der Default-Wert von *0* bei `\arabic*`, *viii* bei `\roman*`, *m* bei `\alph*` verwendet, die großgeschriebenen Varianten funktionieren ähnlich. (Diese Werte können mittels `\widest` verändert werden, siehe unten). Beispiele dafür sind:

```
\begin{itemize}[label=\textbullet, leftmargin=*]
\begin{enumerate}[label=\roman*), leftmargin=*, widest=iii]
\begin{itemize}[label=\textbullet,leftmargin=2pc, labelsep=*]
\begin{enumerate}[label=\arabic*., leftmargin=2\parindent,
labelindent=\parindent, labelsep=*]
```

Am nützlichsten sind dabei `labelsep=*` und `leftmargin=*`. Mit Ersterem beginnt der Textkörper an einer fest definierten Stelle (namentlich `leftmargin`), mit Letzterem an einer variablen, basierend auf der Breite des Labels (aber innerhalb einer Liste natürlich immer an der gleichen). In der Regel wird es am günstigsten sein, `leftmargin=*` zu verwenden.

Leider definiert LaTeX keinen Default-Wert für `labelsep`, der in allen Listen angewendet werden könnte, es wird einfach immer nur der momentane Wert genutzt. Mit enumitem lassen sich Default-Werte für jede Liste festlegen, wie unten beschrieben wird, und um sicherzugehen, dass man volle Kontrolle über `labelsep`hat, benötigt man lediglich etwas wie:

```
\setlist{itemsep=.5em}
```

`labelwidth=*` und `labelwidth=!` funktionieren analog.

`widest=`⟨*string*⟩ `widest*=`⟨*integer*⟩ NEW 3.0 `widest`

Wenn man es wünscht, lassen sich diese Befehle in Verbindung mit den *-Werten verwenden. Sie überschreiben den Default-Wert des weitesten ausgegebenen Zählers. Wenn Listen nicht sehr lang sind, ist ein Wert *a* für `\alph` manchmal passender als der Default-Wert *m*:

```
\begin{enumerate}[leftmargin=*,widest=a] % zweite Gliederungsebene
```

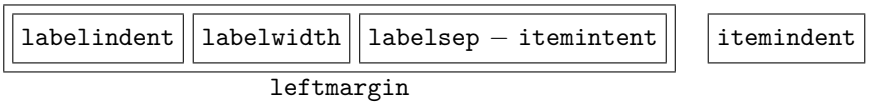
Wird kein Wert festgelegt, tritt wieder der Default-Wert in Kraft. Mit `widest*` wird die Zeichenkette unter Verwendung von ⟨*integer*⟩ als Zählerwert gebildet (beispielsweise sind `\roman`, `widest=viii` und `widest*=8` gleichbedeutend).

Da `\value` keine Zeichenketten sondern nur Nummern verarbeiten kann, funktionieren `widest` und die *-Werte nicht. Es geht aber mit `widest*`, da dessen Wert eine Nummer ist.

3.3 Mehr zur horizontalen Ausrichtung

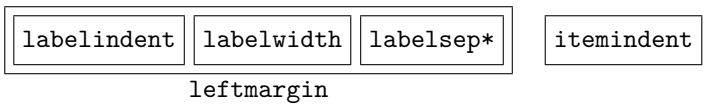
Da `\parindent` als solches innerhalb von Listen keine Anwendung findet, sondern nur intern für `\itemindent` oder `\listparindent` gesetzt wird, gibt enumitem, wenn `\parindent` als Parameter verwendet wird, den globalen Wert aus, also den Wert, den `\parindent` außerhalb der Liste besitzt.

Der horizontale Leerraum des linken Rands in der momentanen Gliederungsebene verteilt sich wie folgt:[6]



`labelsep*=`⟨*length*⟩ NEW 3.0

Zu beachten ist, dass `labelsep` Teile von `leftmargin` und `itemindent` umfasst, wenn Letzteres nicht 0 ist. Das ist oftmals verwirrend, deshalb wird hier ein weiterer Schlüssel eingeführt. Mit `labelsep*` wird der Wert vom linken Rand aus berechnet (es setzt `labelsep` und addiert `itemindent` hinzu, zusätzlich werden auch spätere Änderungen von `itemindent` berücksichtigt).



`labelindent*=`⟨*length*⟩ NEW 3.0

Dies funktioniert wie `labelindent`, wird aber vom linken Rand der momentanen Liste berechnet und von dem der übergeordneten Liste bzw. des Texts.

3.4 Nummerierung, Unterbrechung und Wiederaufnahme

`start=`⟨*integer*⟩

Legt die Nummer des ersten Items fest.

`resume`

Der Zähler der letzten `enumerate`-Liste wird fortgeführt und nicht wieder auf 1 gesetzt.

```
\begin{enumerate}
\item First item.
\item Second item.
\end{enumerate}
Text.
\begin{enumerate}[resume]
\item Third item
\end{enumerate}
```

Dies funktioniert aber nur lokal. Globale Wiederaufnahme wird im folgenden Abschnitt über Serien erläutert.

`resume*`

Wie `resume`, nur dass zusätzlich noch die Optionen der vorherigen Liste übernommen werden. Diese Option muss auf das optionale Argument in einer Umgebung beschränkt sein

[6] Zugegebenermaßen hat die Abbildung nicht die gewünschte Aussagekraft, aber dies soll sich in einer zukünftigen Version ändern.

(denn nur da ist es sinnvoll). Es sollte sparsam eingesetzt werden – verwendet man es oft, wird man wohl eher eine neue Liste definieren (siehe Abschnitt 7). Es ist möglich weitere Schlüssel zu definieren, in diesem Fall werden die gespeicherten Optionen von denen der aktuellen Liste überschrieben. Die Position von `resume*` spielt dabei keine Rolle. Zum Beispiel:

```
\begin{enumerate}[resume*,start=1] % oder [start=1,resume*]
```

Es nutzt die Schlüssel vom vorherigen `enumerate`, startet jedoch den Zähler. Wenn es eine Reihe einer bestimmten Liste mit `resume*` fortsetzt, werden die Optionen aus der vorherigen Liste übernommen, mit Ausnahme von `start`.

3.5 Serien

`series=`⟨*series-name*⟩ NEW 3.0
⟨*series-name*⟩ `resume*=`⟨*series-name*⟩ `resume=`⟨*series-name*⟩ NEW 3.0

Eine neue Möglichkeit (3.0) Listen weiterzuführen, besteht in der Verwendung des Schlüssels series. Dabei fungiert eine Liste mit dem Schlüssel series als Startliste und ihre Einstellungen werden *global* gespeichert, so dass die spätere Wiederaufnahme mittels `resume`/`resume*` erfolgen kann. Alle Schlüssel verlangen einen Namen als Wert, der sich von allen bereits vergebenen Namen unterscheiden muss.

- `resume=`⟨*Serien-Name*⟩ führt einfach die Nummerierung der Bezugsliste weiter
- `resume*=`⟨*Serie*⟩ übernimmt auch die Einstellungen der Startliste
- ⟨*Serie*⟩, d. h. die Verwendung des Seriennamens als Schlüssel, ist synonym zu `resume*=`⟨*Serie*⟩

Zum Beispiel gibt

```
\begin{enumerate}[label=\arabic*(a),leftmargin=1cm,series=lafter]
\item A
\item B
\end{enumerate}
```

eine Nummerierung 1(a), 2(a) aus. Daran kann man mit

```
\begin{enumerate}[label=\arabic*(b),resume*=lafter]
                % or [label=\arabic*(b),lafter]
\item A
\item B
\end{enumerate}
```

anschließen, was zu 3(b), 4(b) führt. (Man kann aber auch `start=1` verwenden.)

Zu beachten ist, dass man noch weitere Argumente hinzufügen kann, die dann nach denen aus der Startliste ausgeführt werden und deshalb bevorzugt behandelt werden. So wird insbesondere auch `resume*` gegenüber einem `start=1` aus der Startliste bevorzugt.

Jedes Mal, wenn eine Serie gestartet wird, werden einige Befehle intern gespeichert, um Ressourcenverschwendung und die Verwendung des gleichen Namens für sich nicht überlappende Serien zu vermeiden.

3.6 Penalties (Strafpunkte)

`beginpenalty=`⟨*integer*⟩ `midpenalty=`⟨*integer*⟩ `endpenalty=`⟨*integer*⟩

Setzt einen Penalty an den Beginn einer Liste, zwischen die Items oder ans Ende. Man sollte sich aber unbedingt in einem TEX- oder LATEX-Handbuch erkundigen, wie sich Penalties auf Seitenbrüche auswirken. Wird eine neue Liste gestartet, werden, im Gegensatz zu anderen Parametern, deren Werte nicht zurück auf Default gesetzt und finden folglich Anwendung in den Töchterlisten.

`before=`⟨*code*⟩ `before*=`⟨*code*⟩

Führt den Kode aus, bevor die Liste gestartet wird (oder präziser, im zweiten Argument der list-Umgebung, welches zu deren Definition genutzt wird). Die ungesternte Variante bewirkt, dass der Kode ausgeführt wird und dabei jeden vorherigen Wert überschreibt, die gesternte fügt den Kode zu dem bereits existierenden hinzu (unter Berücksichtigung der Befehlshierarchie, ohne Beeinflussung von übergeordnetem Text oder den Listen). Es kann Regeln und Text enthalten, dies wurde aber noch nicht ausführlich getestet. Alle Berechnungen werden zuerst beendet und man hat Zugriff auf die Listenparameter und kann diese verändern. Folgendes Beispiel zeigt, wie man beide Ränder (rechts und links) auf das weiteste Label setzen kann:

```
\setlist{leftmargin=*,before=\setlength{\rightmargin}{\leftmargin}}
```

`after=`⟨*code*⟩ `after*=`⟨*code*⟩

Wie zuvor, nur kurz vor dem Ende der Liste.

3.7 `description`/Beschreibungsstile

Hierbei handelt es sich um einen Schlüssel, der in `description` verfügbar ist.

`style=`⟨*name*⟩

Dies legt den description-Stil fest. Für ⟨*name*⟩ können folgende Stile eingesetzt werden:

- `standard`: Wie description in Standard-Klassen, wenn das Ergebnis mit anderen Klassen etwas anders ausfallen kann. Das Label wird eingerahmt. Setzt `itemindent=!`.
- `unboxed`: Ähnlich wie das Standard-description, nur ohne gerahmtes Label, um zu vermeiden, dass die Ausrichtung uneinheitlich wird oder dass sehr lange Labels nicht umgebrochen werden. Setzt `itemindent=!`.
- `nextline`: Wenn das Label nicht in den Rand passt, wird der Text in der nächsten Zeile weitergeführt. Andernfalls wird es in einem Kasten der Breite `\leftmargin` − `\labelsep` plaziert, d. h. der Textkörper der Items reicht niemals bis in den linken Rand hinein. Setzt `labelwidth=!`.
- `sameline`: Wie bei `nextline`, nur dass, wenn das Label nicht in den Rand passt, der Text auf der gleichen Zeile weitergeführt wird. Synonym zu `style=unboxed,labelwidth=!`.
- `multiline`: Das Label erscheint in einer Parbox, deren Breite dem linken Rand (`leftmargin`) entspricht. Falls nötig, reicht sie auch über mehrere Zeilen. Synonym zu `style=standard,align=parleft,labelwidth=!`.

  Drei Warnungen: (1) Die Mischung von gerahmten und umgerahmten Labels weist kein wohldefiniertes Verhalten auf. (2) wenn Listen verschachtelt werden, sind zwar alle Kombinationen erlaubt, aber nicht alle ergeben einen Sinn und (3) verschachtelte Listen mit `nextline` werden nicht unterstützt (zwar funktioniert es, aber das könnte sich in Zukunft ändern, da ihr momentanes Verhalten nicht dem entspricht, was man erwarten könnte.)

3.8 Kompakte Listen

`noitemsep` `nolistsep`

Der Schlüssel `noitemsep` entfernt den Leerraum zwischen Items und Paragraphen (d. h. `itemsep=0pt` und `parsep=0pt`), während `nolistsep` alle vertikalen Leerräume entfernt.[7]

[7] Der Schlüssel `nolistsep`, jetzt veraltet, führte einen kleinen Leerraum (*thin stretch*) ein, was nicht das Gewünschte war.

3.9 „Weite“ Listen

`wide` NEW 3.0
`wide=`⟨*parindent*⟩

Mit diesem bequemen Schlüssel wird der linke Rand (`leftmargin`) auf Null gesetzt und das Label wird Teil des Texts, d. h. die Items sehen wie gewöhnliche Paragraphen aus.[8] `labelsep` sorgt dabei für den Abstand zwischen dem Label und dem ersten Wort. Der Schlüssel ist äquivalent zu

```
align=left, leftmargin=0pt, labelindent=\parindent,
listparindent=\parindent, labelwidth=0pt, itemindent=!
```

Über `wide=`⟨*parindent*⟩ lässt sich auf einmal ein anderer Wert setzen (anstatt von `parindent`). Natürlich können diese Werte nach `wide` auch außer Kraft gesetzt werden, beispielsweise sei daran erinnert, dass mit linksbündigen Etiketten der Text verschoben wird, wenn sie breiter als `labelwidth` sind. Sie können `labelwidth=1.5em` für eine minimale Breite setzen oder anstelle von `itemindent=! itemindent=*` bevorzugen. Das setzt die minimale Breite in Bezug zum breitesten Label. Auf der zweiten Stufe können Sie `labelindent=2\parindent` vorziehen, und so weiter und so fort. Vielleicht möchten Sie es auch mit `noitemsep` oder `nolistsep` kombinieren.

4 Inline-Listen

NEW 3.0

Inline-Listen sind „horizontale“Listen, die als normaler Text in einem Absatz auftreten. Mit diesem Paket lönnen Sie Inline-Listen, wie im Folgenden erläutert wird, mit `\newlist`, die ihre eigen Labels und Zähler haben, kreieren. Jedoch sind dies in den meisten Fällen Inline-Versionen von Standardlisten, mit dem gleichen Labelschema. Es wird ausreichen, die Paketoption `inline` anzuwenden.

`inline` (package option)
`enumerate*` `itemize*` `description*` (environments)

Mit der Paket-Option `inline` werden drei Umgebungen für Inline-Listen definiert: `enumerate*`, `itemize*` und `description*`. Sie emulieren das Verhalten von `paralist` und `shortlst` dahingehend, dass Labels und Einstellungen mit den angezeigten (d. h. „normalen“) Listen `enumerate`, `itemize` und `description` geteilt werden (man sollte im Kopf behalten, dass die Wiederaufnahme von Listen vom Umgebungs-Namen abhängt und nicht von den Listen-Typen). Das gilt aber nur für diejenigen Listen, die mit `inline` erzeugt wurden, Inline-Listen, die wie unten beschrieben, mittels `\newlist` gebildet wurden, sind unabhängig und nutzen ihre eigenen Labels und Einstellungen. Merken Sie sich, dass `inline` nicht erforderlich ist, wenn die Inline-Versionen der Standardlisten nicht benötigt werden.

`itemjoin=`⟨*string*⟩ `itemjoin*=`⟨*string*⟩ `afterlabel=`⟨*string*⟩

Das Format wird mit den Schlüsseln `itemjoin` (der Default ist ein Leerzeichen) und `afterlabel` (der Default ist `\nobreakspace`, also `~`) gesetzt. Ein zusätzlicher Schlüssel ist `itemjoin*`, welches, wenn es gesetzt wird, anstatt `itemjoin` vor dem letzten Item genutzt wird. Das ergibt mit

```
before=\unskip{: }, itemjoin={{; }}, itemjoin*={{, and }}
```

die folgende Zeichensetzung zwischen den Items:

Blah blah: (a) Eins; (b) Zwei; (c) Drei und (d) Vier. Blah blah

[8] `fullwidth` verliert seinen Wert.

`itemjoin` wird im vertikalen Modus ignoriert (also bei `mode=unboxed`, gleich nach einem Zitat, einer angezeigten Liste und Ähnlichem).

`mode=unboxed`	`mode=boxed`

Die Items sind gerahmt, weshalb sie keine Gleitobjekte sein können und verschachtelte Listen nicht erlaubt sind (man sollte sich daran erinnern, dass viele dargestellte Elemente als Listen definiert sind). Wenn man Gleitobjekte und Listen innerhalb von Inline-Listen verwenden möchte, muss man einen alternativen „Modus“ verwenden, welcher sich mittels `mode=unboxed` aktivieren lässt (der Default ist `mode=boxed`). Damit lassen sich Gleitobjekte frei verwenden, aber fehlplatzierte `\item`s werden nicht eingefangen und `itemjoin*` wird ignoriert (es wird auch eine Warnung in die Log-Datei geschrieben).

5 Globale Einstellungen

Globale Veränderung, die auf alle diese Listen angewendet werden, sind ebenfalls möglich.

`\setlist[enumerate,`⟨*levels*⟩`]{`⟨*format*⟩`}`
`\setlist[itemize,`⟨*levels*⟩`]{`⟨*format*⟩`}`
`\setlist[description,`⟨*levels*⟩`]{`⟨*format*⟩`}`
`\setlist[`⟨*levels*⟩`]{`⟨*format*⟩`}`

Dabei ist ⟨*level*⟩ die Listenebene (eine oder mehrere) in list und die korrespondierende Ebene in enumerate und itemize.[9] Wird der optionale Parameter ⟨*levels*⟩ nicht spezifiziert, gilt das Format für alle Ebenen. Listen meint an dieser Stelle nicht alle Listen, sondern nur die drei, die von diesem Paket verarbeitet werden, und diejenigen, die vom Paket umdefiniert oder über `\newlist` definiert werden (siehe unten). Zum Beispiel:

```
\setlist{noitemsep}
\setlist[1]{\labelindent=\parindent} % < Usually a good idea
\setlist[itemize]{leftmargin=*}
\setlist[itemize,1]{label=$\triangleleft$}
\setlist[enumerate]{labelsep=*, leftmargin=1.5pc}
\setlist[enumerate,1]{label=\arabic*., ref=\arabic*}
\setlist[enumerate,2]{label=\emph{\alph*}),
                      ref=\theenumi.\emph{\alph*}}
\setlist[enumerate,3]{label=\roman*), ref=\theenumii.\roman*}
\setlist[description]{font=\sffamily\bfseries}
```

Diese Einstellungen werden in folgender Reihenfolge gelesen: list, list auf der momentanen Ebene, enumerate/itemize/description und enumerate/itemize/description auf der momentanen Ebene. Erscheint ein Schlüssel mehrere Male mit unterschiedlichen Werten, wird der letzte, also der spezifischste, angewendet. Wird eine Serie oder eine Liste weitergeführt, finden die gespeicherten Schlüssel Anwendung. Als Letztes wird das optionale Argument (außer `resume*`) gelesen.

LaTeX stellt eine Reihe Makros zur Verfügung, um viele dieser Parameter zu ändern, aber es ist konsistenter und manchmal flexibler, sie mit diesem Paket zu setzen, auch wenn dies zu Lasten der Klarheit (und *verbose*) geht.

Die Spezifikation der Liste kann Variablen und Zähler enthalten, vorrausgesetzt, dass sie erweiterbar und die Zähler calc-savvy sind. Wenn man das Paket lädt, kann man also etwas wie das Folgende schreiben:

```
\newcount{toplist}
\setcount{toplist}{1}
\newcommand{\mylistname}{enumerate}
\setlist[\mylistname,\value{toplist}+1]{labelsep=\itemindent+2em]
```

[9] `\setenumerate`, `\setitemize` und `\setdescription` verlieren ihre Gültigkeit als Befehle.

Damit lassen sich Listen in Schleifen definieren.

Momentan gibt es keine Möglichkeit, die Größe des Fonts zu unterscheiden (`\normalsize`, `\small`...).

6 enumerate-artige Label

`shortlabels` (Paket-Option)

Mit der Paket-Option `shortlabels` kann eine Syntax wie bei enumerate verwendet werden, wo `A`, `a`, `I`, `i` und `1` für `\Alph*`, `\alph*`, `\Roman*`, `\roman*` und `\arabic*` stehen. Die Hauptabsicht dahinter ist, dass dies als eine Art Kompatibilitätsmodus mit dem Paket enumerate dient, weshalb die folgende Regel Anwendung findet: Wird die allererste Option (auf jeder Ebene) nicht als gültiger Schlüssel erkannt, dann wird davon ausgegangen, dass es sich um ein Label mit enumerate-artiger Syntax handelt. Zum Beispiel:

```
\begin{enumerate}[i), labelindent=\parindent]
...
\end{enumerate}
```

Auch wenn es vielleicht nicht allzu nützlich ist, kann `label=` auch in der itemize-Umgebung unter ähnlichen Bedingungen weggelassen werden:

```
\begin{itemize}[\textbullet]
...
\end{itemize}
```

`\SetEnumerateShortLabel{`⟨*key*⟩`}{`⟨*replacement*⟩`}`

Mit diesem Befehl können neue Schlüssel definiert bzw. umdefiniert werden, was besonders nützlich ist, um enumerate an spezifische typographische Regeln anzupassen oder dessen Funktionsumfang auch auf nicht-lateinische Schriften zu erweitern. ⟨*replacement*⟩ enthält an dieser Stelle eine der gesternten Varianten der Zähler. Zum Beispiel definiert

```
\SetEnumerateShortLabel{i}{\textsc{\roman*}}
```

`i` in der Art um, dass Items mit kleinen römischen Zahlen nummeriert werden. Der Schlüssel muss ein einzelnes Zeichen sein.

7 Klonen der Basisliste

`\newlist{`⟨*name*⟩`}{`⟨*type*⟩`}{`⟨*max-depth*⟩`}`
`\renewlist{`⟨*name*⟩`}{`⟨*type*⟩`}{`⟨*max-depth*⟩`}`

Die drei Listen können geklont werden, um „logische“ Umgebungen so zu definieren, dass sie sich wie diese verhalten. Um eine neue Liste zu definieren oder eine existierende umzudefinieren, benutzt man `\newlist` oder `\renewlist`. Dabei ist ⟨*type*⟩ entweder enumerate, itemize oder description.

`\setlist[`⟨*names*⟩`,`⟨*levels*⟩`]{`⟨*keys/values*⟩`}` `\setlist*[`⟨*names*⟩`,`⟨*levels*⟩`]{`⟨*keys/values*⟩`}`

Nachdem man eine Liste erstellt hat, kann man eine neue Liste mit `\setlist` setzen (eigentlich muss man sogar, zumindest das Label):

```
\newlist{ingredients}{itemize}{1}
\setlist[ingredients]{label=\textbullet}
\newlist{steps}{enumerate}{2}
\setlist[steps,1,2]{label=(\arabic*)}
```

Die Namen im optionalen Argument von `\setlist` besagen, auf welche Listen die Einstellungen angewendet werden und die Nummer drückt aus, bis zu welcher Ebene. Mehrere Listen und/oder mehrere Ebenen können angegeben werden und alle Kombinationen sind gesetzt; z. B. setzt

```
\setlist[enumerate,itemize,2,3]{...}
```

enumerate/2, enumerate/3, itemize/2 und itemize/3. Keine Nummer (oder 0) bedeutet „alle Ebenen“ und kein Name „alle Listen“; kein optionales Argument bedeutet „alle Listen auf allen Ebenen“.

Die drei Inline-Listen haben die Typen enumerate*, itemize* und description*, die immer verfügbar sind, selbst ohne die Paket-Option `inline` (welche lediglich drei Umgebungen mit diesen Namen definiert).

Die gesternte Form `\setlist*` fügt die aktuellen Einstellungen zu den vorher gesetzten hinzu.

`\setlistdepth{`⟨*integer*⟩`}` NEW 3.0

Standardmäßig hat LaTeX ein Limit von 5 verschachtelten Ebenen, aber wenn man Listen klont, kann dieser Wert zu klein sein, weshalb man einen neuen setzen möchte. In Ebenen unterhalb der 5ten (oder der tiefsten von einer Klasse definierten) finden die Einstellungen der letzten Ebene Anwendung (d. h. `\@listvi`).

8 Mehr zu Zählern

8.1 Neue Zählerreprs̈entation

`\AddEnumerateCounter{`⟨*LaTeX command*⟩`}{`⟨*internal command*⟩`}{`⟨*widest label*⟩`}`

Dieser Befehl „registriert“ eine Repräsentation eines Zählers, so dass enumitem ihn erkennen kann. Er ist hauptsächlich für nicht-lateinische Schriften gedacht, aber auch für lateinische Schriften nützlich. Zum Beispiel:

```
\makeatletter
\def\ctext#1{\expandafter\@ctext\csname c@#1\endcsname}
\def\@ctext#1{\ifcase#1\or First\or Second\or Third\or
Fourth\or Fifth\or Sixth\fi}
\makeatother
\AddEnumerateCounter{\ctext}{\@ctext}{Second}
```

Die gesternte Variante erlaubt es, anstatt einer Zeichenkette eine Nummer als weitestes Label anzugeben; z. B. wenn das weiteste Label jenes ist, welches mit dem Wert 2: NEW 3.0 korrespondiert:

```
\AddEnumerateCounter*{\ctext}{\@ctmoreext}{2}
```

Diese Variante sollte bevorzugt werden, wenn die Repräsentation keine bloße Zeichenkette sondern mit einem gewissen Stil versehen ist, z. B. mit Kapitälchen. (Der Zählername kann ein @ enthalten, auch wenn es kein Buchstabe ist.)

8.2 enumerate neu starten

`\restartlist{`⟨*list-name*⟩`}` NEW 3.0

Momentan kann man mit

```
\setlist[enumerate]{resume}
```

eine kontinuierliche Nummerierung innerhalb eines Dokuments erreichen. Zusätzlich wurde eine neuer Befehl eingefügt, um den Zähler mitten im Dokument neu zu starten:

`\restartlist{`⟨*list-name*⟩`}`

Er basiert lediglich auf dem Listennamen, nicht dem Listentyp, was heißt, dass `enumerate*`, wie es mit der Paket-Option `inline` definiert wurde, nicht gleichbedeutend mit `enumerate` ist, da die Namen verschieden sind.

9 Generische Schlüssel und Werte

`\SetEnumitemKey{`⟨*key*⟩`}{`⟨*replacement*⟩`}` NEW 3.0

Mit diesem Befehle können eigene Schlüssel (ohne Wert) erzeugt werden. Zum Beispiel:

```
\SetEnumitemKey{midsep}{topsep=3pt,partopsep=0pt}
```

Derartig erzeugte Schlüssel können wie die anderen verwendet werden. Ein weiteres Beispiel ist eine `multicolumn`-Liste, mit multicol:

```
\SetEnumitemKey{twocol}{
  itemsep=1\itemsep,
  parsep=1\parsep,
  before=\raggedcolumns\begin{multicols}{2},
  after=\end{multicols}}
```

(Die Einstellungen für `itemsep` und `parsep` vernichten die gedehnten und geschrumpften Teile. Natürlich können Sie sich wünschen, eine neue Liste zu definieren.)

Ein Hinweis: Das Paket kann in der Zukunft neue Schlüssel einführen, so ist `\SetEnumitemKey` eine potenzielle Quelle für Vorwärtsinkompatibilitäten. Allerdings ist es sicher, Nicht-Buchstaben-Zeichen zu nehmen, anders als mit Bindestrich oder Stern im Schlüsselnamen (beispielsweise `:name` oder `2_col`).

`\SetEnumitemValue{`⟨*key*⟩`}{`⟨*string-value*⟩`}{`⟨*replacement*⟩`}` NEW 3.0

Dieser Befehl stellt eine weitere Abstraktionsebene für die Schlüssel-Wert-Paare (⟨*key*⟩=⟨*value*⟩ bereit. Mit ihm kann man logische Namen definieren, die dann zum aktuellen Wert übertragen werden. Zum Beispiel kann mit

```
\SetEnumitemValue{label}{numeric}{\arabic*.}
\SetEnumitemValue{leftmargin}{standard}{\parindent}
```

Folgendes gesetzt werden:

```
\begin{enumerate}[label=numeric,leftmargin=standard]
```

Damit lässt sich bis zum Ende offenhalten, was `label=numeric` bedeutet.

10 Paketoptionen

Neben `inline`, `ignoredisplayed` und `shortlabels` wird noch die folgende Option bereitgestellt:

`loadonly`

Mit dieser Option wird zwar `enumitem` geladen, die drei Listen aber nicht umdefiniert. Man kann aber trotzdem eigene Listen erstellen oder sogar bereits existierende umdefinieren.

11 Drei Vorlagen

Es gibt gute Gründe dafür anzunehmen, dass drei Listenlayouts besonders häufig sind. Im Folgenden wenden wir die obigen Parameter auf diese an, um sie umzudefinieren. (Beispiele finden sich unten.)

Die erste Vorlage richtet die Labels nach dem umgebenden `\parindent` aus, während der Textkörper der Items von dem Label und einem fixierten `labelsep` abhängt:

```
labelindent=\parindent,
leftmargin=*
```

Eine ziemlich häufige Variante ist es, das Label am umgebenden Text auszurichten (es ist zu beachten, dass der Default von `labelindent` 0pt ist):

```
leftmargin=*
```

Ersteres sieht auf der ersten Ebene besser aus, Zweiteres wäre eventuell auf den untergeordneten Ebenen vorzuziehen. Dies kann sehr leicht mittels

```
\setlist{leftmargin=*}
\setlist[1]{labelindent=\parindent} % Nur Ebene 1
```

gesetzt werden.

Die zweite Vorlage richtet den Textkörper der Items an dem umgebenden `\parindent` aus. In diesem Fall:

```
leftmargin=\parindent
```

Die dritte Vorlage würde schließlich das Label an `\parindent` und den Textkörper der Items an `2\parindent` auszurichten:

```
labelindent=\parindent,
leftmargin=2\parindent,
itemsep=*
```

Eine weitere Variante davon ist es, dass man das Label am umgebenden Text und den Textkörper der Items an `\parindent` ausrichtet.

```
leftmargin=\parindent,
itemsep=*
```

Anzumerken ist, dass sich `\parindent` an dieser Stelle auf den globalen Wert bezieht, der auf die normalen Paragraphen angewendet wird.

12 Das trivlist-Problem

LaTeX verwendet eine vereinfachte Version von list, die trivlist genannt wird, um angezeigtes Material wie `center`, `verbatim`, `tabbing`, `theorem` usw. zu setzen, auch wenn es sich dabei konzeptuell gar nicht um Listen handelt. Unglücklicherweise greift trivlist auf die momentanen Listeneinstellungen zurück, so dass es zu dem ungewollten Nebeneffekt kommen kann, dass Veränderungen bei den vertikalen Abständen zwischen Listen manchmal zur Veränderungen der Abstände in diesen Umgebungen führen.

Dieses Paket modifiziert trivlist dahingehend, dass die Default-Einstellungen für die aktuelle Ebene (d. h. diejenigen, die durch die zugehörigen clo-Dateien bestimmt werden) wiederhergestellt werden. Im Standard-LaTeX ist dies für gewöhnlich redundant, aber wenn man Listen feinabstimmen möchte, kann sich eine Nicht-wiederherstellung der Default-Werte als problematisch erweisen (besonders dann, wenn man die Option `nolistsep` verwenden möchte).

Ein Minimum an Kontrolle der vertikalen Abstände wird möglich durch[10]

[10] `\setdisplayed` verliert seinen Wert.

`\setlist[trivlist,⟨level⟩]{⟨keys/values⟩}`

Aber trivlist selbst, welches eher selten direkt genutzt wird, erlaubt kein optionales Argument. Diese Funktion ist nicht als umfassender trivlist-Formatierer gedacht.

Wenn man es aus irgendwelchen Gründen nicht möchte, dass die Veränderungen an trivlist in Kraft treten, kann dies über die Option `ignoredisplayed` erreicht werden.

13 Beispiele

Installieren Sie bitte erst das Paket und setzen Sie das Dokument nochmal.

In diesen Beispielen ist `\setlist{noitemsep}` gesetzt:

```
En un lugar de la Mancha, de cuyo nombre no quiero acordarme,
no ha mucho tiempo que viv\'{\i}a un hidalgo de los de
\begin{enumerate}[labelindent=\parindent,leftmargin=*]
  \item lanza en astillero,
  \item adarna antigua,
  \item roc\'{\i}n flaco, y
  \item galgo corredor.
\end{enumerate}
Una olla de algo m\'{a}s vaca que carnero, salpic\'{o}n las m\'{a}s
noches, duelos y quebrantos los s\'{a}bados...
```

Diese Regel zeigt `labelindent`.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

1. lanza en astillero,
2. adarna antigua,
3. rocín flaco, y
4. galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Mit `\begin{enumerate}[leftmargin=*] % labelindent=0pt by default`.
Diese Regel zeigt `labelindent`.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

1. lanza en astillero,
2. adarna antigua,
3. rocín flaco, y
4. galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Mit `\begin{enumerate}[leftmargin=\parindent]`.
Diese Regel zeigt `leftmargin`.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

1. lanza en astillero,
2. adarna antigua,
3. rocín flaco, y
4. galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Mit `\begin{enumerate}[labelindent=\parindent, leftmargin=*, label=\Roman*., widest=IV, align=left]`.
Diese Regel zeigt `labelindent`.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

I. lanza en astillero,
II. adarna antigua,
III. rocín flaco, y
IV. galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Mit `\begin{enumerate}[label=\fbox{\arabic*}]`. Ein Verweis auf das erste Item ist 1

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

1 lanza en astillero,
2 adarna antigua,
3 rocín flaco, y
4 galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Mit verschachtelten Listen.

```
En un lugar de la Mancha, de cuyo nombre no quiero acordarme,
no ha mucho tiempo que viv\'{\i}a un hidalgo de los de
\begin{enumerate}[label=(\alph*), labelindent=\parindent,
     leftmargin=*, start=12]
\item lanza en astillero,
\begin{enumerate}[label=(\alph{enumi}.\roman*), leftmargin=*, start=7]
\item adarna antigua,
\end{enumerate}
\item roc\'{\i}n flaco, y
\begin{enumerate}[label=(\alph{enumi}.\roman*), leftmargin=*, resume]
\item galgo corredor.
\end{enumerate}
\end{enumerate}
Una olla de algo m\'{a}s vaca que carnero, salpic\'{o}n las m\'{a}s
noches, duelos y quebrantos los s\'{a}bados...
```

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

(l) lanza en astillero,

(l.vii) adarna antigua,

(m) rocín flaco, y

(m.viii) galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

```
En un lugar de la Mancha, de cuyo nombre no quiero acordarme,
no ha mucho tiempo que viv\'{\i}a un hidalgo de los de
\begin{description}[font=\sffamily\bfseries, leftmargin=3cm,
    style=nextline]
  \item[Lo primero que ten\'{\i}a el Quijote] lanza en astillero,
  \item[Lo segundo] adarna antigua,
  \item[Lo tercero] roc\'{\i}n flaco, y
  \item[Y por \'{u}ltimo, lo cuarto] galgo corredor.
\end{description}
Una olla de algo m\'{a}s vaca que carnero, salpic\'{o}n las m\'{a}s
noches, duelos y quebrantos los s\'{a}bados...
```

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

Lo primero que tenía el Quijote lanza en astillero,
Lo segundo adarna antigua,
Lo tercero rocín flaco, y
Y por último, lo cuarto galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Genauso, nur mit `sameline`.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

Lo primero que tenía el Quijote lanza en astillero,
Lo segundo adarna antigua,
Lo tercero rocín flaco, y
Y por último, lo cuarto galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

Genauso, aber mit `multiline`. Es ist zu beachten, dass sich der Text überlappt, wenn der Textkörper für die Items zu kurz ist.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

Lo primero lanza en astillero,
Lo segundo adarna antigua,
Lo tercero rocín flaco, y
Y por último, lo cuarto galgo corredor.

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados...

14 Nachwort

14.1 LaTeX-Listen

Wie bereits bekannt ist, gibt es in LaTeX drei vordefinierte Listen: enumerate, itemize und description. Dabei handelt es sich um eine gängige Klassifikation, die so beispielsweise auch in HTML zu finden ist. Es existiert aber noch ein allgemeineres Modell, welches sich in drei Felder einteilen lässt: Label, Titel und Körper. So haben enumerate und itemize zwar Label (nummeriert und unnummeriert), aber keine Titel, während description Titel hat, aber keine Label. In diesem Modell kann man eine description-Liste haben, deren Einträge mit Labels markiert sind, wie zum Beispiel (natürlich ist diese einfache Lösung nicht zufriedenstellend):

```
\newcommand\litem[1]{\item{\bfseries #1,\enspace}}
\begin{itemize}[label=\textbullet]
\litem{Lo primero que ten\'{\i}a el Quijote} lanza en astillero,
... etc.
```

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de

- **Lo primero que tenía el Quijote,** lanza en astillero,
- **Lo segundo,** adarna antigua,
- **Lo tercero,** rocín flaco, y
- **Y por último, lo cuarto,** galgo corredor.

Diese Formatierung ist nicht selten und ein Werkzeug, um es zu definieren, ist in Arbeit und schon sehr weit fortgeschritten. Es ist in Version 3.0 noch nicht enthalten, da ich mir nicht sicher bin, ob dieses Paket der richtige Platz dafür ist oder ob nicht doch titesec besser geeignet wäre; außerdem arbeitet es noch nicht stabil genug.

14.2 Bekannte Probleme

- Die Wiederaufnahme von Listen basiert auf Umgebungsnamen. Wenn eine `\newenviroment` eine Liste enthält, möchte man vielleicht `\begin` und `\end` verwenden. Die Verwendung der korrespondieren Befehle ist kein Fehler, aber man ist selbst dafür verantwortlich, dass das Resultat korrekt ist.
- Es scheint keinen Weg zu geben, falsch geschriebene Namen in `\setlist` aufzufangen und der Compiler gibt die sinnlose Fehlermeldung „Missing number, treated as zero“ aus.
- Das Verhalten der gleichzeitigen Anwendung von gerahmten Labels (einschließlich enumerate und itemize) und ungerahmten Labels ist nicht wohldefiniert. Ähnlich verhält es sich mit der gleichzeitigen Weiterführung von Serien und Listen: Es ist zwar möglich, aber ihr Verhalten ist ebenfalls nicht wohldefiniert.
- (3.5.2) Eine Inkompatibilität mit 2.x ist aufgetaucht – wenn ein optionales Argument genommen wurde, um den Wert von `\ref` zu übertreffen, oder andere Makros erforderten expandierende Makros – dann wird ein Fehler ausgelöst. Eine schnelle Lösung ist das `\makelabel` nach `\descriptionlabel` zu lassen.

14.3 Neuerungen in Version 3.0

- Inline-Listen mit Schlüsseln, um zu bestimmen, wie die Items verbunden werden (d. h. die Zeichensetzung zwischen ihnen). Es sind zwei Modi erlaubt: `boxed` und `unboxed`.
- `\setlist` ist calc-savvy (z. B. zur Benutzung in Schleifen) und es lassen sich unterschiedliche Listen und Ebenen auf einmal festlegen.
- Alle Längen, die mit Labels zu tun haben, können den Wert `*` annehmen, nicht nur `labelsep` und `leftmargin`. Ihr Verhalten wurde vereinheitlicht und es gibt einen neuen Wert `!`, mit dem das weiteste Label nicht verarbeitet wird.
- Mit `\restartlist{`⟨*list-name*⟩`}` können Listenzähler neu gestartet werden (falls `resume` verwendet wird).
- `resume*` kann zusammen mit anderen Schlüsseln verwendet werden.
- Listen können global durch die Verwendung von Serien zusammengefasst werden, so dass sie wie eine einzige Liste behandelt werden. Um eine Serie zu starten, verwendet man einfach nur `series=`⟨*series-name*⟩, die Wiederaufnahme erfolgt durch `resume=`⟨*series-name*⟩ oder `resume*=`⟨*series-name*⟩.
- Das „experimentelle“ `fullwidth` wurde durch den neuen Schlüssel `wide` ersetzt.
- `\SetLabelAlign` definiert neue Werte zur Ausrichtung.
- Es lassen sich „abstrakte“ Werte (z. B. `label=numeric`) und neue Schlüssel definieren.

- (3.2) `start` und `widest*` sind calc-savvy.
- (3.2) `\value` kann zusammen mit `widest*` verwendet werden.
- (3.2) Eine interne Einschränkungen in `\arabic` und dergleichen wurden entfernt. Es ist flexibler zu Lasten einer „entspannteren“ Fehlersuche.

14.4 Bug-Fixes

- Gesternte Werte (z. B. `leftmargin=*`) konnten nicht außer Kraft gesetzt werden und neue Werte wurden ignoriert.
- `nolistsep` wurde als erster von mehreren Schlüsseln nicht immer erkannt und als Kurzlabel behandelt (d. h. `nol\roman*stsep`).
- `labelwidth` funktionierte nicht immer (wenn es vorher ein `widest` und `*` gab).
- Mit `align=right` konnten sich Label und nachfolgender Text überlappen.
- In description gab es Probleme mit der korrekten Listenebene.
- Ab einem bestimmten Punkt (Version 2.x) hörte `value*` auf, richtig zu funktionieren.
- (3.1) Unglücklicherweise „vernichtet“ `xkeyval` `keyval`, so dass Letzteres in enumitem nachgebildet werden musste.
- (3.3) Ein schwerwiegender Fehler wurde entfernt: Weder itemize noch description funktionierten in ihrer gesternten Variante.
- (3.4) Ein schlechter Abstand wurde entfernt (fehl platziert `\unskip` vor dem ersten Element und ein falscher *space*-Faktor zwischen Elementen).
- (3.4) `nolistsep` funktioniert nicht, wie vorgesehen, wurde vor mehreren Jahren festgestellt, deshalb wurde ein neuer Schlüssel `nosep` zur Verfügung gestellt.
- (3.4) Das Problem von `nolistsep` mit `shortlabels` (siehe oben) war nicht für alle Fälle behoben. Hoffentlich ist es jetzt so.
- (3.5.0) Es wurde ein Problem mit dem Abstandsfaktor zwischen Items behoben.
- (3.5.0) Es wurde ein Problem mit verschachtelten Boxen in Inline-Listen beseitigt.
- (3.5.1) `resume*` arbeitete nur einmal und die folgenden verhielten sich nur wie `resume`.
- (3.5.2) Beseitigte Fehler mit `\setlist*`, weil es nicht funktionierte.