revnumerate — a reverse enumerate-environment*

Jörn Wilms†

May 12, 1997

1 Introduction

While writing on a style-file for the list of references section of my CV I realized that there is no environment in LaTeX that enumerates items in reverse order. In my case I needed such an environment to present my list of publications in descending temporal order, while keeping them numbered in ascending temporal order. In other words, the first publication should be labeled with "1." while still being the last in the list, since the most recent publications should be presented first.

While this problem is easily stated at first, a solution of it was more difficult: Since the number of items in the list is not known at the beginning, they need to be counted first, then the number of items in the environment needs to be written in LaTeXs `aux`-File, to be used in the next LaTeX-run. In addition, the environment should fit smoothly into the already existing environments of LaTeX and the interface should just behave like the normal `enumerate`-environment, so that style-changes to `enumi`...`enumvi` would also affect the `revnumerate` environment.

My solution to the problem is documented below, if you have comments, if you have found bugs, or if you have extensions to the environment, send me email at `wilms@astro.uni-tuebingen.de`.

2 The User-Interface

`revnumerate` The user-interface to the `revnumerate`-environment is very simple. For example,

```
\begin{revnumerate}
   \item Point three is the most important,
   \item Point two is not as nice,
   \item and Point one is uninteresting.
\end{revnumerate}
```

and running LaTeX twice gives

3. Point three is the most important,

*This file has version number v1.0, last revised 1997/05/10.

†Universität Tübingen, Institut für Astronomie und Astrophysik, Abt. Astronomie, Waldhäuser Str. 64, D-72076 Tübingen, Germany, email: wilms@astro.uni-tuebingen.de

2. Point two is not as nice,
1. and Point one is uninteresting.

The `revnumerate`-environment has an optional argument that gives the starting number:

```
\begin{revnumerate}[10]
\item One
\item Two
\item Three
\item Four
\end{revnumerate}
```

results in

10. One
9. Two
8. Three
7. Four

It is possible to nest the environment with the other LaTeX-environments without any problems, as well as using the normal redefinition of `\theenumi` to `\theenumiv` to get a different behaviour of the output:

```
\renewcommand{\labelenumi}{\S\theenumi.}
\renewcommand{\theenumii}{\roman{enumii}}
\begin{enumerate}
   \item This is point one
   \item This section has two subpoints:
   \begin{revnumerate}
     \item Subpoint two\label{pt22}
     \item Subpoint one
   \end{revnumerate}
   \item and Point three.
\end{enumerate}
Where point~\ref{pt22}\ldots
```

The above text gives

§1. This is point one

§2. This section has two subpoints:

(ii) Subpoint two

(i) Subpoint one

§3. and Point three.

Where point 2ii...

Pretty nice, isn't it?

3 The Code

First we identify the style-file.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{revnum}[1997/05/10 v1.0 reverse enumerate, jw]
```

Next, we need to create four new counters, `rev@cnti` to `rev@cntiv`. These counters are used in the environment to count the number of `revnumerate`-environments present at each possible level. Since we need to keep track on how many entries each environment contains, we need a way to create counters such that they uniquely define each environment. The solution to this is to use an "index", provided by `rev@cnti` to `rev@cntiv` that gets appended to the counter for each environment. If that confuses you, see below.

```
\newcounter{rev@cnti}   \def\therev@cnti{i\arabic{rev@cnti}}
\newcounter{rev@cntii}  \def\therev@cntii{ii\arabic{rev@cntii}}
\newcounter{rev@cntiii} \def\therev@cntiii{iii\arabic{rev@cntiii}}
\newcounter{rev@cntiv}  \def\therev@cntiv{vi\arabic{rev@cntiv}}
```

`revnumerate` Now let's start with the `revnumerate`-environment. We have one option, the start number. If it is not given we initialize the start-number to -1, to indicate to the code that we want to use the value from the `.aux`-file.

```
\newenvironment{revnumerate}[1][-1]%
{%
```

As in the case of generic LaTeX-environments, only four nestings are possible. The current depth is given by the counter `\@enumdepth`, and if it is larger than three we need to output an error-message. If not, we increase the nesting-depth by one.

```
\ifnum\@enumdepth >\thr@@\@toodeep\else
  \advance\@enumdepth\@ne
```

As the generic environments, we use one of the counters `enumi` to `enumiv` as the counter for our environment. This way, all changes made to the counters `\theenumi`...`\theenumiv` will also affect the `revnumerate` environment.

```
  \edef\@enumctr{enum\romannumeral\the\@enumdepth}
```

To count uniquely identify the current `revnumerate`-environment, we use one of the counters `rev@cnti` to `rev@cntiv`. We first define, which counter to use, call it `\@revcnt` and then increase it by one to get the number of the current environment.

```
  \edef\@revcnt{rev@cnt\romannumeral\the\@enumdepth}
  \stepcounter{\@revcnt}
```

The number of items in the current list will be counted with a counter called something like `revi1`, `revii1`, `revii2`, etc. The name of this counter is created by appending the value of `rev@cnti` to `rev@cntiv` to the string `rev`.

```
  \edef\the@revcnt{rev\csname the\@revcnt\endcsname}
```

If the current counter is *not* known yet, i.e. if the counter has not been defined in the `.aux`-file, then we need to define it with `\newcounter` and initialize it. Just using `\newcounter` is not enough, since that will initialize the counter to 0. Since the `revnumerate`-environment counts the counter backwards, this would give negative counter-values, which would result in bogus error-messages if the counter is output, e.g. in alphabetical manner. We therefore initialize it to 26, which should give positive numbers for all `revnumerate`-environments where alphabetical output is desired. The strange \ifx...\relax sequence is an \if@undefined. I don't know why I didn't use \if@undefined directly...

```
  \expandafter\ifx\csname c@\the@revcnt\endcsname\relax%
    \newcounter{\the@revcnt}
    \setcounter{\the@revcnt}{26}
  \fi
```

In the next step we need to initialize \@enumctr, which will produce the labels to be output, to its starting value. This value is either the value given by the counter \the@revcnt, or it is the argument of the environment. In the default-case, when the argument did not have an argument, #1 gets set to -1.

```
  \ifnum #1 <0
    \setcounter{\@enumctr}{\value{\the@revcnt}}
  \else
    \setcounter{\@enumctr}{#1}
  \fi
```

Since we need to add -1 to \@enumctr before we output the label. Only in this case the \label-command will work correctly. I don't know why, but at least the current implementation works. In addition we need to reset the \the@revcnt-counter back to zero to count the number of items in the current environment.

```
  \stepcounter{\@enumctr}%
  \setcounter{\the@revcnt}{0}%
```

The revnumerate-environment is defined via the list-environment, thus making it easily changeable. For outputting the label we first reduce the environment-counter \@enumctr by one and then add one to \the@revcnt. Then the "variable" \@currentlabel, which contains the way the current position in the text will be referenced with \ref gets set to something like \p@enumi \the@enumi. The commands \p@enumi to \p@enumiv are usually defined in the class and are used to make labels more meaningful. For example, a reference to subpoint "b" of item number 3 will result in a "3b", and not just "b". Finally, the label is output using one of the commands \labelenumi to \labelenumiv.

```
  \begin{list}%
     {\addtocounter{\@enumctr}{-1}%
      \stepcounter{\the@revcnt}%
      \global\edef\@currentlabel
          {\csname p@\@enumctr\endcsname\csname the\@enumctr\endcsname}%
      \csname label\@enumctr\endcsname%
     }{}%
\fi
}{%
```

At the end of the environment, we need to write the number of items in the current environment to the aux-file. Since the aux-file is read twice during the processing of a LaTeX-file, once at the beginning and once at the end of the document, it is not sufficient to just output a \newcounter and a \setcounter. Instead, we have to output the \if@undefined-sequence to only do a \newcounter if necessary (i.e. during the first read on the aux-file). What I have not figured out yet is how to warn the user that the file needs to be processed twice. This could be done using a label, but that seems unelegant. So far, we'll just hope that everybody is TeXing the file at least twice...

```
\end{list}
```

```
\protected@write\@auxout{}{%
  \string\expandafter%
  \string\ifx\string\csname\space c@\the@revcnt\string\endcsname\relax%
  \string\newcounter {\the@revcnt}\string\fi
}
\protected@write\@auxout{}{%
   \string\setcounter {\the@revcnt}%
     {\number\csname c@\the@revcnt\endcsname}
   }
}
```