…three, two, one… *a quest for the number of numbers*

Keywords
enumerate, numbered items, lists, counting backwards, etaremune

Abstract
This article briefly describes the seemingly trivial task of numbering items in a list with decreasing numbers, starting with the number equal to the amount of items in the list.

Introduction

Some time ago, someone on the TEX-NL mailinglist asked whether it would be possible to have items in an `enumerate` environment numbered with decreasing numbers instead of increasing ones. Hendri presented a quick hack on the LaTEX `enumerate` environment as he didn't know the revnum [3] package yet. At first, his solution required submitting the amount of items to the environment as a parameter, but later versions of his solution didn't need that input anymore.

After the existence of the revnum package was brought to his attention, Hendri looked into the way this package operates. This package consumes numerous TEX counter registers. Hendri found that his own solution, with some extra work, could be an alternative to achieve the goal with a low consumption of counter registers. His solution would evolve to be the etaremune package [1], which is the subject of this small article.

Isn't this trivial?

The original idea does sound trivial indeed, but it really isn't when you realize that once LaTEX has read and typeset input, there is no way to change the typeset version anymore. So, once we have typeset the first item in the list and gave it, for instance, number '1', we can't change that number to a '3' anymore if we find 2 more items in the list. So, the basic problem is to already know the number of items in a list before starting and typesetting this list.

Two possible solutions spring to mind.

2. Extract the entire body of the environment, that should number items backwards, to an external file, parse it to find the number of `\item`'s in the body and insert the body back into the input stream. This is very tricky for several reasons, one being the fact that LaTEX `enumerate` environments can be nested and hence our new environment should also support that. For other possible problems related to this technique, see for instance the extract package [2].
1. Apply a two-step procedure as is usual in LaTEX (for instance to construct the table of contents). In the first LaTEX run, we typeset the item numbers in a default way and at the same time, count the number of items and save this to the `.aux` file. At the second run, we use this number to properly set the starting number (the number of the first item) and all following items.

Technique 1 seems the easiest and hence the most appealing one. This is the technique employed by the revnum and etaremune packages.

How to implement it then?

So now we know what technique we will use. But how do we implement it? First, we need to keep in mind that the environment should be nestable, just as `enumerate` is. Secondly, we need a counter register to do the actual counting of the items in the environment. Unfortunately, we cannot use the fact that the TEX group inside `\begin...\end` will do all counting locally to the environment and not touch the counting of items in an environment in a higher level (in case we are in a nested environment). This doesn't work[1] as the code needed to count items can only be inserted into `\item` via `\@itemlabel` which will be typeset in a box, keeping counting local to a single `\item` and resetting the counter again after leaving the item label box. Adding a `\global` isn't a solution as it makes counting escape all the way to the top level, hence also modifying the counted number of items in higher level lists.

How to proceed? The revnum package tries to work around this problem by using a counter register for every new `revnum` environment that the user starts (plus four to keep track of the number of the environment in the document etcetera). However, when using a lot of `revnum` environments and of course other packages that consume the necessary counters as well, one might end up asking a TEXpert on the TEX-NL mailinglist to extend TEX.

But we really do not need to use that many counters. We only need a counter for counting the items and one for counting the environments in the document, so we can say in the next LaTEX run: "list 113 has 37 items". And when we encounter a nested list, we need to make sure that, when leaving this nested list, we can reset the two counters and continue counting the items in the list in the higher level (which has a lower number than the nested list). We do this by temporarily saving counter values to macros when entering a nested list.

Besides counting items on the first LaTEX run, we need to provide some output. The revnum package then starts at 26, so that second and fourth level lists can use the alphabet without having problems of trying to access letter -1. However, when one adds items to an existing list, the package will still try to typeset letter -1 from the alphabet, resulting in problems. The etaremune package typesets an `etaremune` environment as an `enumerate` environment on the first run and makes sure that counters do not get lower than 0 to avoid errors when adding entries to nested lists.

New perspectives

The package opens a hoard of new possibilities. So far, counting backwards has been the 'modus operandi' in just a few organizations. But they use it well and we could all benefit from their expertise. At NASA, people count backwards toward the launching moment of rockets and space shuttles. Engineers operating explosives for demolition of buildings also count backwards toward the toppling of the tower. In general, controlled processes that go 'boom' are counted backwards as they advance. But low-tech destructive and high-tech astro-pyrotechnics aren't the only industries using the countdown method of advancement. In popular music it's customary to count backwards to the presentation of the current best selling ditty on the airwaves. And last but not least, Agatha Christie's play 'Ten Little Indians' about ten strangers in a secluded place uses the thrill-enhancing device of countdown as the characters are knocked off one at a time.

With the basic concept of this new package, countdown can change the way we create documents and even improve the way we read and meet.

For instance, how relevant is it to read a novel and see the page number incline? Unless we know the page number of the last page, there's little use for the page number other than a way to refer to a given section. But if the novel starts at the highest page number, those tiny numbers at the corners of the page indicate how far we are from arriving at the final page. The first page has its own unmistakable prominence at the very start of the book, after the title page and the table of contents, with a high number and the final page is at the very back of the book but it's proudly adorned with the number one! Using the ideas of this article, one could perform this task. It would require some work[2], but it would be worth it. There would even be the option of having a page zero, for 'zeroing in' on the epilogue!

And meetings can have an agenda starting with the highest number, so attendants can work towards a fixed end and not be bothered about a seemingly endless conference. Meetings could get snappier and more concise, making the organization more productive just by counting more sensibly.

This package could change the way we meet, the way we read and the way we think. And we can reserve the count-upwards method for those enigmatic increments towards ends unknown, like our life cycle. Adding one to our age every year does provide us with the mild illusion that in principle we could go on living forever, until the numbers run out. At least we give it a try...

Footnotes

1. Well, it might work if we would redefine `\item` and friends, but that is too risky in relation to other packages.
2. In fact, one would need to change the command `\stepcounter{page}` to `\addtocounter{page}{-1}` in LaTEX's output routine additionally to finding the number of pages (which could also be done with the lastpage package).

References

[1] Hendri Adriaens. etaremune package, v1.1, 2005/04/15. CTAN:/macros/latex/contrib/etaremune.

[2] Hendri Adriaens. extract package, v1.7, 2005/03/31. CTAN:/macros/latex/contrib/extract.

[3] Jörn Wilms. revnum package, v1.0, 1997/05/10. CTAN:/macros/latex/contrib/revnum.

Frans Goddijn
frans[at]goddijn.com
Hendri Adriaens
hendri[at]uvt.nl