The tabularx package*

David Carlisle
carlisle@cs.man.ac.uk

1995/03/20

Abstract

A new environment, `tabularx`, is defined, which takes the same arguments as `tabular*`, but modifies the widths of certain columns, rather than the inter column space, to set a table with the requested total width. The columns that may stretch are marked with the new token `X` in the preamble argument.

This package requires the `array` package.

1 Introduction

This package implements a version of the `tabular` environment in which the widths of certain columns are calculated so that the table is is a specified width. Requests for such an environment seem to occur quite regularly in `comp.text.tex`.

`tabularx` `\begin{tabularx}{`⟨*width*⟩`}{`⟨*preamble*⟩`}`
The arguments of `tabularx` are essentially the same as those of the standard `tabular*` environment. However rather than adding space between the columns to achieve the desired width, it adjusts the widths of some of the columns. The columns which are affected by the `tabularx` environment should be denoted with the letter `X` in the preamble argument. The `X` column specification will be converted to `p{`⟨*some value*⟩`}` once the correct column width has been calculated.

2 Examples

The following table is set with `\begin{tabularx}{250pt}{|c|X|c|X|}` ....

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.[1]	three	Column four will act in the same way as column two, with the same width.

If we change the first line to `\begin{tabularx}{300pt}{|c|X|c|X|}` we get:

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.

*This file has version number v2.02, last revised 1995/03/20.
[1]You can now use `\footnote` inside `tabularx`!

3 Differences between `tabularx` and `tabular*`

These two environments take the same arguments, to produce a table of a specified width. The main differences between them are:

- `tabularx` modifies the widths of the *columns*, whereas `tabular*` modifies the widths of the inter-column *spaces*.
- `tabular` and `tabular*` environments may be nested with no restriction, however if one `tabularx` environment occurs inside another, then the inner one *must* be enclosed by `{ }`.
- The body of the `tabularx` environment is in fact the argument to a command, and so certain constructions which are not allowed in command arguments (like `\verb`) may not be used.[2]
- `tabular*` uses a primitive capability of TeX to modify the inter column space of an alignment. `tabularx` has to set the table several times as it searches for the best column widths, and is therefore much slower. Also the fact that the body is expanded several times may break certain TeX constructs.

4 Customising the behaviour of `tabularx`

4.1 Terminal output

`\tracingtabularx` If this declaration is made, say in the document preamble, then all following `tabularx` environments will print information about column widths as they repeatedly re-set the tables to find the correct widths.

As an alternative to using the `\tracingtabularx` declaration, either of the options `infoshow` or `debugshow` may be given, either in the `\usepackage` command that loads `tabularx`, or as a global option in the `\documentclass` command.

4.2 The environment used to typeset the `X` columns

By default the `X` specification is turned into `p{`⟨*some value*⟩`}`. Such narrow columns often require a special format, this may be achieved using the `>` syntax of `array.sty`. So for example you may give a specification of `>{\small}X`. Another format which is useful in narrow columns is ragged right, however LaTeX's `\raggedright` macro redefines `\\` in a way which conflicts with its use in a tabular or array environments. For this reason this package introduces the command `\arraybackslash`, this may be used after a `\raggedright`, `\raggedleft` or `\centering` declaration. Thus a `tabularx` preamble may specify
`>{\raggedright\arraybackslash}X`.

`\arraybackslash`

`\newcolumntype` These preamble specifications may of course be saved using the command, `\newcolumntype`, defined in `array.sty`. Thus we may say
`\newcolumntype{Y}{>{\small\raggedright\arraybackslash}X}`
and then use `Y` in the `tabularx` preamble argument.

`\tabularxcolumn` The `X` columns are set using the `p` column which corresponds to `\parbox[t]`. You may want them set using, say, the `m` column, which corresponds to `\parbox[c]`. It is not possible to change the column type using the `>` syntax, so another system is provided. `\tabularxcolumn` should be defined to be a macro with one argument, which expands to the `tabular` preamble specification that you want to correspond to `X`. The argument will be replaced by the calculated width of a column.

The default is `\newcommand{\tabularxcolumn}[1]{p{#1}}`. So we may change this with a command such as:
`\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}`

[2]Since Version 1.02, `\verb and \verb*` may be used, but they may treat spaces incorrectly, and the argument can not contain an unmatched `{` or `}`, or a `%` character.

4.3 Column widths

Normally all `X` columns in a single table are set to the same width, however it is possible to make `tabularx` set them to different widths. A preamble argument of `{>{\hsize=.5\hsize}X>{\hsize=1.5\hsize}X}` specifies two columns, the second will be three times as wide as the first. However if you want to play games like this you should follow the following two rules.

- Make sure that the sum of the widths of all the `X` columns is unchanged. (In the above example, the new widths still add up to twice the default width, the same as two standard `X` columns.)
- Do not use `\multicolumn` entries which cross any `X` column.

As with most rules, these may be broken if you know what you are doing.