A LaTeX Tour, part 2: the Tools and Graphics distributions

David Carlisle

1 Introduction

In the previous article in this series I started by giving a description of the files in the 'base' LaTeX distribution. In part 2, I shall cover the 'tools' and 'graphics' distributions. These are distributed in the `tools` and `graphics` subdirectories of the CTAN directory `macros/latex/packages`. Although these files are not part of the minimal base distribution they should normally be included in the LaTeX installation at any site. The LaTeX book assumes that at least the graphics distribution is installed.

The primary source for LaTeX is the 'CTAN [1]' network of archives, so if I refer to path names of files this relates to the CTAN file structure. Note however that if you obtained LaTeX as part of a 'pre-packaged' TeX distribution, then these files may have been moved (typically documentation files may be separated from TeX source files). Hopefully this will not cause any confusion.

2 The Tools Distribution

The tools distribution consists of packages written by individual members of the LaTeX3 project. They are supported by the same mechanism as the base LaTeX distribution, that is any problems should be reported using `latexbug.tex` and the LaTeX bug report database, as described in part 1. Note that this bug report system should *not* be used for 'contributed' packages that one may find in the `macros/latex/contrib` area of the CTAN archives.

2.1 Packages Extending the array and tabular Environments

The first group of packages extend the functionality of the standard LaTeX array and tabular environments. These are all described in Chapter 5 of *The LaTeX Companion*, as well of course as in the source '`.dtx`' files which may be processed by LaTeX to produce typeset documentation, and optionally code listings.

`array` Extended versions of the array, tabular and tabular* environments. The principal advantage of the versions provided by this package is that you can specify typesetting instructions to apply to a whole column of the table. As well as the usual `clr` column specifiers, one may add commands at the beginning of each entry with `>` and at the end of each entry with `<`. So a column specifier of `>{\bfseries}c` would produce a bold, centred column of a table.

The array package also provides a `\newcolumntype` command for defining new column specifiers, in addition to the standard ones. This is used by some of the packages described below.

`dcolumn` Alignment on 'decimal points' in tabular entries. Requires array. This package provides a new column specifier `D` which may be used to produce columns of numbers aligned on a decimal point '.' or some other symbol, such as '·' or ','.

`delarray` This package requires the array package. It provides a mechanism for specifying 'large delimiters' around arrays. This is most convenient for putting brackets around arrays that are to be aligned on their top or bottom row (when the 'obvious' construction with `\left` and `\right` does not work). Compare the standard

```
\left(
  \begin{array}[t]{c}a\\b\end{array}
\right)
\left(
  \begin{array}{cc}a&b\end{array}
\right)
```

$$\left(\begin{array}{c}a\\b\end{array}\right)\left(\begin{array}{cc}a&b\end{array}\right)$$

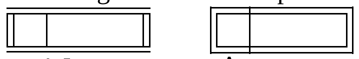
with the effect produced using the dcolumn syntax.

```
\begin{array}[t]({c})a\\b\end{array}
\begin{array}({cc})a&b\end{array}
```

$$\left(\begin{array}{c}a\\b\end{array}\right)\left(\begin{array}{cc}a&b\end{array}\right)$$

`hhline` Finer control over horizontal rules in tables. Requires array. Standard LaTeX's `\hline\hline` construction produces a double rule across a table, however the user has no control over how this rule interacts with vertical rules. Using the `\hhline` command provided by this package, one can make 'corners' where a double horizontal rule meets a double vertical rule, and other similar effects.

Compare the first, standard construction with the following `\hhline` sample:



`longtable` Standard LaTeX tables (i.e., the tabular environment) produce 'boxes' that can not be broken across a page. This has advantages in that

[1] `ftp.tex.ac.uk` in the UK

the table can then be positioned just like a large 'character' for instance centred by the center environment, but has the disadvantage that large tables need to be broken by hand to fit on the page. The longtable environment is essentially the same as tabular but produces tables that break at page boundaries, and has some additional commands to control 'head' and 'foot' lines of the table that are added to each page. If the array is also loaded, then the extra features may also be used in longtable column specifications. Note that longtable can deal with *very* long tables, longer than can be stored in memory by TeX's primitive `\hline` command. The longtable package has a few quirks and features that mean that it is not suitable in all cases. An alternative package (currently maintained by Johannes Braams, but as a contributed package, not as part of the tools distribution) is the package supertab which provides a similar supertabular environment.

`tabularx` Defines the tabularx environment which is similar to tabular* but modifies column widths, not inter-column space, to achieve a desired table width.

One common request is to combine the features of tabularx with longtable, i.e., have a table across multiple pages, in which the widths of the 'parbox' columns are calculated automatically. This functionality is not provided by the standard packages in the tools distribution, but the experimental contributed package ltxtable does provide such an environment. (ltxtable is written by the same author as longtable and tabularx; however, problems with ltxtable should *not* be addressed to the LaTeX bugs system.) An alternative to ltxtable is Anil Goel's contributed package, ltablex, a similar merger which is simpler to use than ltxtable, but not quite as powerful.

2.2 Missing File Error Control Files

Although these files (which are all generated from the same source file, `fileerr.dtx`) are distributed as part of the LaTeX distribution, they are possibly of more use when used with *other* formats. The primitive TeX behaviour if asked to input a non-existent file is to offer a prompt:

```
Please type another input file name:
```

You *must* type a valid file name to this prompt or TeX just repeats the request. On some systems you can use a mechanism to abort the job (e.g., control-C or control-Z) but there is no way to tell TeX to skip the input or do any other error recovery.

To avoid this unpleasant loop, LaTeX always checks that a file exists before trying to input it (unless you use the primitive `\input filename` syntax with no `{ }` around the argument).

If you do encounter this loop using a different format, or with LaTeX, by mistyping the file name on the command line, then the following `.tex` files provide valid filenames that you can easily remember which you can type to the missing file prompt. The actions that each of these `.tex` files takes is designed to mimic the actions that are possible after a TeX error.

`h` Typing `h` ⟨*return*⟩ to the missing file prompt will cause TeX to input `h.tex`, this produces a helpful message, and then produces the normal 'error prompt' i.e., `?` so you can hit ⟨*return*⟩ to move on, or `x` to quit, or whatever.

`s` Typing `s` ⟨*return*⟩ inputs `s.tex` which puts TeX into 'scroll mode'. This means that it will scroll past future errors without stopping.

`x` Typing `x` ⟨*return*⟩ causes the current TeX run to be aborted.

`e` The file `e.tex` is in fact the same as `x.tex` but allows `e` to be given as an answer to the missing file prompt similar to the `e` response to the error prompt (which is supposed to start up an editor but usually is the same as `x`).

␣ If the operating system allows there will also be a file `.tex` which does nothing, this will mean that just hitting ⟨*return*⟩ in response to the missing file prompt inputs `.tex` and allows TeX to proceed with the original file. Some operating systems object to a file with only an extension and no filename before the '`.`' so this option may not be available to you. Most TeX distributions include a file `null.tex` which is also empty, so if you do not have the option of installing the file `.tex` you may type `null` ⟨*return*⟩ in response to the missing file prompt, which will also allow TeX to proceed.

2.3 Miscellaneous Tools Packages

`afterpage` Defines an `\afterpage` command that saves up its argument and executes it after the current page (i.e., at the top of the next page). LaTeX's output routine was *not* designed with the idea that packages might want to play kind of trick, so this package is particularly fragile. In fact it was only written as a kind of private joke; I noticed the comment "*Output routines are always protected by enclosing them in groups, so that they do not inadvertently mess up the rest of TeX*" in the TeXbook, and wanted to answer[2] the question, "*Where do you end up if you jump out of that group with `\aftergroup`?*" Despite this, judging by comments in `comp.text.tex`, people do seem to find the package useful...

`enumerate` Extended version of the enumerate environment. The environment is given an optional argument which controls how the counter is printed.

[2]The answer incidentally is not at the top of the next page, but rather any of the places where the TeXbook uses the magic phrase "*exercises the page builder*".

For example `\begin{enumerate}[a)]` would produce items labelled 'a)' ' b)' ' c)'.

`ftnright` Place footnotes in the right hand column in two-column mode. Normally LaTeX places footnotes at the bottom of each column. This package causes the footnotes for both columns of a page to be set in the normal text area at the end of the second column on each page. It currently works only with the standard two column mechanism, not with the mechanism of the multicol package.

`indentfirst` Indent the first paragraph of sections etc. This very small package just suppresses the usual LaTeX mechanism which ensures that the first paragraph of each section is not indented.

`multicol` Typeset text in columns (up to 10 columns per page), with the length of the final columns 'balanced'. *Baskerville* uses this package to balance the columns at the end of every article. Unlike the standard `\twocolumn` command, this package allows changing the number of columns part way down a page. It does have some restrictions on the use of floats which means that it is not suitable for all purposes. Also (uniquely for the files in the core LaTeX distribution) this package has special conditions on commercial use.

`rawfonts` Preload fonts under the old internal font names of LaTeX 2.09. Not recommended for new packages, but may help when updating old files.

`somedefs` This package is not intended to be called directly by a document, but may by used (via `\RequirePackage`) to build a package in which you want the default behaviour to be to execute *all* possible options, but that the user may execute just some of the options by specifying options in the `\usepackage` call. This is used in the rawfonts package above to allow just some of the 'old' font names to be defined rather than all of them.

`theorem` Flexible definition of 'theorem-like' environments. The standard `\newtheorem` command gives some control over the title and numbering of 'theorem-like' declarations, but is not very flexible. The theorem package provides an enhanced declaration scheme which gives control over the fonts used in the heading and theorem body, and such details as whether the numbering is '**Theorem 1**' or '**1 Theorem**'. Recently this package has acquired a close cousin, the amsthm package, part of the 'AMS-LaTeX' collection. The AMS variant has perhaps slightly simpler user-syntax but is used in much the same way.

`varioref` Provides `\vref` and related commands. `\vref` is like a combination of `\ref` and `\pageref` which produces references such as 'Figure 2 on page 3'. However, it omits the page number if it is on the current page, and replaces it by phrases such as 'on the facing page' when appropriate.

`verbatim` Flexible version of verbatim environment. The standard LaTeX verbatim environment can not easily be used in the definition of other environments as typically the `\end` of the newly defined environment is not recognised as such, but is treated as verbatim text. This package re-implements verbatim such that (with some restrictions) it can be used in the definition of other environments and commands. It also defines some such derived environments, for inputting and writing files verbatim, and for adding line numbers, and also a comment environment that ignores all the environment body.

`xr` The xr (external references) package allows one LaTeX document to access the `.aux` file of another. So if file `fileA` has a section marked with `\label{xyz}` then file `fileB` may refer to that section using `\ref{xyz}` just as if it were part of the same document. This requires the file `fileA.aux` created when `fileA` was processed to be still available when `fileB` is processed. (This package was originally by Jean-Pierre Drucbert, but was recoded and adopted into the tools distribution.)

`xspace` One of the more common errors in TeXdocuments is to use a command such as `\TeX` within text, but forget to follow it with `\␣` or `{}`. This package defines a command `\xspace` which may be used at the end of the definition of such a 'text command'. It looks ahead at the next token and adds a space unless that token is a punctuation character.

2.4 Packages for Drafts and Tests

`fontsmpl` Package and test file for producing 'font samples'. The base distribution contains a file `nfssfont.tex` that shows some small samples, and a character table for a given font. `fontsmpl.tex` produces a much more extensive test showing examples of all the fonts in a given family. If you want to devise your own similar test suite you may use the fontsmpl package, following the examples in `fontsmpl.tex`.

`layout` Defines a `\layout` command that produces a half size 'picture' of the document page settings such as `\textwidth`, `\oddsidemargin`, ... together with a table of their values. This is quite useful when designing a new class file, as it gives a visual representation of how the various areas of the page for headlines, body text, marginal notes, etc., relate to each other.

`showkeys` LaTeX's automatic numbering and cross referencing feature is one of its strongest points, as it makes editing a document (and thus potentially changing the numbering throughout the rest of the file) quite painless. However, one disadvantage is that when reading a printed draft, one sees 'final' numbers rather than the symbolic names that are used in the source file's `\label` and `\ref` command. This package makes these symbolic names, or 'keys', appear in the margin in the case of

$\boxed{\text{lbl}}$

`\label` and `\bibitem` or raised above the number, like this $\overset{\texttt{lbl}}{1}$, in the case of `\ref` and `\cite`. Some people find the raised labels above cross references distracting and so a package option turns them off, just leaving the marginal notes showing the `\label` and `\bibitem` keys.

3 The Graphics Distribution

TeX (and the `dvi` format) is only designed to deal with rectangular boxes consisting of text, white space or rectangular rules. However it has an ‘escape mechanism’, the `\special` primitive command that allows processing instructions to be passed straight from TeX (via the `dvi` file) to the ‘driver’ program that is used to process (e.g., preview or print) the `dvi` file. TeX places essentially no restrictions on what instructions may be passed via `\special`, and so the possibilities are unlimited…

Most modern drivers can import ‘graphic’ files of various sorts. Those drivers that are producing POSTSCRIPT can often do more extensive manipulations of the typeset text, such as scaling or rotation of the text, or even writing text along an arbitrary curve. Many of these drivers can also support colour to some extent. Unfortunately as all these features require that the `dvi` file stores processing instructions for the driver, it means that the `dvi` file is not portable to a site that uses a different driver program. There have been many attempts over the years to coordinate the `\special` syntax used by the different drivers, so that they would all accept a common core of processing instructions, but there has been notable lack of success in such efforts to date…

As a ‘next best thing’ to having portability at the `dvi` level, LaTeX supplies a suite of standard graphics commands provided by the packages described in this section, so that at least TeX source files should be reasonably portable. At a given site the graphics packages will be customised to use a suitable ‘back end’ file that converts the LaTeX syntax into the form required by the local driver. This should mean that as long as both drivers support some feature, such as including POSTSCRIPT graphics, a file just needs to be re-processed with LaTeX to use the `\special`s at the new site; the LaTeX file does not need to be edited. Although this suite of programs was devised as part of LaTeX, users of other TeX formats may use them by way of the interface available from CTAN hosts in `macros/generic/graphics`.

3.1 Documentation

All the packages in this distribution are, as usual, distributed as documented sources in `dtx` form, however the documentation in these package sources is rather technical. A separate ‘User Guide’ is available as LaTeX source in `grfguide.tex` and also in pre-formatted form in the POSTSCRIPT file `grfguide.ps`.

The color package (which produces colours despite the strange spelling) and the graphics package are also described in Lamport’s LaTeX manual. An alternative to `grfguide.tex` as a free source of documentation is Keith Reckdahl’s *Using EPS Graphics in LaTeX2ε Documents* distributed from CTAN sites in the file `info/epslatex.ps` and published in TUGBOAT 17, no. 1–2. This document covers the graphicx package in some depth, and also related contributed packages for controlling figure placement and captions, and the psfrag system for overlaying LaTeX text over a POSTSCRIPT diagram.

3.2 Colour

`color` Produce coloured effects in your document. The `\color{red}` declaration would make all the following text red, the similar `\textcolor` command takes an extra argument that specifies the text to be coloured (by analogy with `\rmfamily` and `\textrm`).
One may also produce boxes with coloured backgrounds using the `\colorbox` command.
Accurate treatment of colour is probably the feature that requires the most ‘help’ from the driver program. If your driver was not specifically written to support colour then probably the color package will not work at all, or will be limited to regions of colour that fall on one page, and all the current colours will be ‘forgotten’ at a page break.

`pstcol` The pstricks package of Tim van Zandt provides a very powerful interface to POSTSCRIPT. Unfortunately the package has some slight incompatibilities with the color package. If a document loads this pstcol package, both color and pstricks are loaded, and then a few internal pstricks functions are redefined to repair the incompatibility.

3.3 Rotation, Scaling and Graphics Inclusion

`graphics` This is the core LaTeX package for rotation (`\rotatebox`), scaling (`\scalebox` and `\resizebox`) of text, and the inclusion of graphics images (`\includegraphics`). Unlike the old LaTeX 2.09 packages such as ps g the `\includegraphics` command is not restricted to POSTSCRIPT graphics, but can include any graphics formats that your driver supports.

`graphicx` Often when including graphics files one needs to specify combinations of scaling and rotation and other special effects. The graphics package uses standard LaTeX ‘positional’ optional arguments which means that it is not practical for any command to support more than a couple of optional arguments. The graphicx package calls the graphics package internally, but offers a more powerful and friendly ‘named argument’ interface in which an arbitrary number of optional keys may be set in one `[ ]` argument. For instance to include a graphic scaled to half size, and rotated through 90°, one can specify

```
\includegraphics[scale=.5, angle=90]{file}
```

To do the equivalent with the graphics package would require nested calls of

`\includegraphics` inside `\scalebox` inside `\rotatebox`.

`lscape` Provides a landscape environment within which the body of every page is rotated through 90°. The page head and foot are not rotated, but stay in their usual positions. It requires the graphics package which is used to handle the rotation.

`epsfig` The obsolete LaTeX 2.09 did not come with a standard graphics package. Two popular contributed packages to include POSTSCRIPT graphics were ps g (T. Darrell) and epsf (T. Rokicki). Sebastian Rahtz merged and extended these to produce the package eps g. The eps g package became very popular, especially after it was given extensive coverage in *The LaTeX Companion*. For this reason the current distribution contains a package called eps g so that old documents do not need converting to the new system. However this eps g is just a wrapper that converts the old syntax into calls to the new `\includegraphics` command and so should not now be used for new documents.

3.4 Driver Files

As mentioned above these packages all require customisation to a particular driver. This may be specified either in a site configuration file, or as a package option in the document. The code for these drivers is all stored in '`.def`' files, so for instance the code for the *dvips* driver (and also for *xdvi* which uses the same `\special` syntax) is stored in `dvips.def`. All these driver files are derived from the same source, `drivers.dtx`, except for the Textures file which is currently distributed as a separate `textures.def` contributed by Arthur Ogawa. One special `.def` file does not correspond to a driver, `dvipsnam.def` predefines the 60+ colours that are 'known' to the *dvips* driver. It may be used with other drivers as well, as described in the color package documentation.

3.5 Other Graphics Packages

The remaining two packages do not have code that is specific to dvi driver programs, and so in some sense do not really belong in the graphics distribution; however they are used by the graphics and graphicx packages. In fact the code of either of these packages may be extracted and used in any format based on plain TeX. They do not use any LaTeX specific features.

`keyval` The graphicx package makes use of a 'named argument' or 'key equals value' syntax as described above. Keyval provides a general parser for such a syntax, so this package is unlikely to be directly called within a document, but may be loaded by `\RequirePackage` by any package or class file that needs to define commands with such a syntax.

`trig` This package provides functions for calculating the trigonometric functions sin, cos and tan. These are used by the graphics package for determining the amount of space a rotated box will take up.

4 Coming Soon

Part 3 of this tour will describe the files of Johannes Braams' babel distribution of packages for multi-lingual typesetting, the psnfss distribution of POSTSCRIPT font related packages, and mfnfss distribution of packages for loading 'Pandora' and 'Old German' fonts.