Baskerville

The Annals of the UK TEX Users' Group **Guest Editor: Malcolm Clark** **Vol. 6 No. 2**

ISSN 1354–5930 **April 1996**

Articles may be submitted via electronic mail to `baskerville@tex.ac.uk`, or on MSDOS-compatible discs, to Sebastian Rahtz, Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, to whom any correspondence concerning *Baskerville* should also be addressed.

This reprint of *Baskerville* is set in Times Roman, with Computer Modern Typewriter for literal text; the source is archived on CTAN in `usergrps/uktug`.

Back issues from the previous 12 months may be ordered from UKTUG for £2 each; earlier issues are archived on CTAN in `usergrps/uktug`.

Please send UKTUG subscriptions, and book or software orders, to Peter Abbott, 1 Eymore Close, Selly Oak, Birmingham B29 4LB. Fax/telephone: 0121 476 2159. Email enquiries about UKTUG to `uktug-enquiries@tex.ac.uk`.

Contents

I Editorial

As you will have gathered from the last edition of *Baskerville,* the editorship of this august organ now rotates around the luckless committee. Whoever fails to avoid our esteemed chair's eye (Robin Fairbairns in Ancient Mariner mode) is deemed (or doomed) to serve. It is a demanding , but fascinating experience. With the resource of the rest of the committee to call on, nothing is insuperable. In the longer term it will build up a pool of expertise which can only be to the good. You begin to appreciate the Rahtz' triplets even more.

Sourcing material has not been difficult for this edition, since we are currently in the throes of two series – David Carlisle's tour of LaTeX, and Sebastian Rahtz' PSTricks exposition. In addition I had an abundance of riches and ended up having to pass on material. This does not imply that we do not continue to crave articles. Looking back over recent issues, the same faces/names pop up over and over again, although one or two contributors have sadly fallen by the wayside – perhaps exhausted by the strain of composition.

I was pleased with David Carlisle's other article, neatly solving a problem raised and discussed at last year's Bridewell meeting on Portable Documents – typesetting from HTML documents. Robin Fairbairns' article on CTAN is a welcome review and state of the art, together with some aspirations for the future. These two papers derive from the recent *TeX and the Internet* meeting. It is unlikely that the group's meetings will generate enough material to fill these pages, so let me paraphrase what I said earlier: "we need your contributions". The observant will also note that committee members continue to generate many of the articles, but by way of contradiction, I'm pleased to welcome Mark Wooding to the pantheon of stars. In my darker moods, I wonder whether *Baskerville* is merely an ego trip for its writers (i.e. mainly the committee) and is unread by its intended readership – yourselves. Is it wise to ask such questions?

As a partial answer, let me note that a pleasing feature of this edition (for me) is that I open it (with the editorial), and close it (with the *Gleanings*, whose final year you may be relieved to know this is).

I am indebted, as ever, but in particular for assistance with this edition to Robin Fairbairns (whose tele-presence was invaluable), David Carlisle and Sebastian Rahtz. Theirs is the glory, mine are the errors and misunderstandings.

II Production Problems

Robin Fairbairns

I had to apologise for the late production of *Baskerville* 5.6, which arose from my continuing distraction from the (several) jobs I had in hand; even when it had been printed, there was a delay before I was able to organise its posting. Because of that delay, we had to send you a separate mailing of the notice of Portable Documents (reprise) meeting.

Baskerville 6.1, produced by David Carlisle, arrived on time, and I was able to get discs to the printer in reasonable time. However, the next day I had to leave Cambridge, and spent only one day here for the next week and a half. Fortunately, I was able to alert the committee to the need to send out a special mailing for the TEX and the Internet meeting. Since they had been printed, I decided to include 'my' fliers for the meeting with the mailing which is now, finally (4 March) in the post. (I'm *sure* the green paper the fliers were printed on didn't look quite so violent when I chose it; I shan't select that colour again!)

One hopes such alarums and diversions won't happen again; I apologise to the membership for any inconvenience.

III 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). I hope 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)
```

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

$$\begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} a & b \end{pmatrix}$$

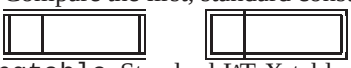
with the effect produced using the dcolumn syntax.

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

$$\begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} a & b \end{pmatrix}$$

`hhline` Finer control over horizontal rules in tables. Requires array. Standard LaTeX's `\hline\hline` construction produces a double rule across a table, but the user has no control over how this rule interacts with vertical rules. Using the `\hhline` command provided by this package, gives '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 cannot be broken across a page. This has advantages in that the table can then be positioned just like a large 'character' (say 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 package 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 `\halign` 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 to type at 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 reactions, 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. 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

[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*".

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 TeX documents 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 and adds a space unless the next 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 `\label` and `\bibitem` or raised above the number, like this lbl, 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 LaTeX 2ε 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 pstcol, 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 psfig 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 psfig (T. Darrell) and epsf (T. Rokicki). Sebastian Rahtz merged and extended these to produce the package epsfig. The epsfig 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 epsfig so that old documents do not need converting to the new system. However this epsfig 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; 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.

IV CTAN past and present — what next?

Robin Fairbairns

1 Introduction

The Comprehensive TEX Archive Network (CTAN) is a set of 'loosely consistent' archives of TEX material which (together with a large set of sites that mirror them) provide a single (logical) point for people to acquire material for their TEX-related work. I've been a user of CTAN since its inception at Aston, but since the move of the Aston node to Cambridge I've been involved in its management; this article derives from a talk I gave at the UK TEX Users' Group meeting on 'TEX and the Internet'.

As far as UK people are concerned, TEX archiving begins with the archive that Peter Abbott established at Aston back in the eighties. Aston was available to people who could talk coloured-book protocols over Janet.[3]

A while after the archive was established, the UKTEX mailing list was started, initially as a means of propagating information about the content of the archive. Volume '87' number 1 of UKTEX talked about accessions to the archive in the week ending 4 September 1987.

Peter's initiative was, at heart, a practical one: connection to the wider Internet was a major undertaking for most people in the UK, so it made sense to collect things available 'on the Internet' into one place for access over Janet. Peter wrote a paper about his experiences 1, which he gave at TUG'89 at Stanford University. I was struck, in re-reading Peter's paper, by the manpower he had to run the archive; the list of eight people reads like one of the great and the good of TEX in the UK.[4]

Peter found at Stanford that the Americans were wildly jealous of his achievement; at the time, whoever felt like it would establish a directory or two on his or her site's ftp server to promulgate their own favourite stuff. This was all very much in the spirit of the "anarchistic Internet", but it made for a terrible to-do to find any given TEX-related item you might be looking for.

Whatever was its real driving force, it was after the Stanford meeting that CTAN appeared. George Greenwade chaired a Technical Working Group on archive structures, and undertook to establish the first truly comprehensive USAn TEX archive at Sam Houston State University (SHSU) in Huntsville, Texas; he reported on the matter in 3. A German node was established at Stuttgart, and with the advent of the JIPS service, UK academics started to have access to the Internet, and the Aston archive became a CTAN node too.

The tale of Aston ended in 1994, when Peter Abbott was about to take early retirement from Aston University. Since we could no longer guarantee that the archive would have a protecting friend, the UK TEX Users' Group committee[5] sought alternative sites for it. Some obvious candidates for the archive site were examined and rejected. The HENSA archive would want to split the holdings between 'Unix' and 'PC' sets, which do not match the structure of CTAN's stuff (TEX is after all a portable program, *par excellence*). The national typesetting archive at Oxford couldn't offer CPU power and disc space in the required time frame. In either case, management of the archive by remote 'experts' could have been problematic. Finally, Prof. Roger Needham kindly agreed to a proposal prepared by Sebastian Rahtz and presented to him by Martyn Johnson, that Cambridge should host the archive. We completed the change-over *just* before Peter retired!

2 Archive mechanisms

I said at the start that CTAN is a weakly-consistent set of archives. What I mean is, that they are allowed to differ in the short term, but (in principle) they will sooner or later get themselves back into synchronisation. Weak consistency is a common thing to find: another TEX-related weakly consistent set is the Refdbms archives, which maintain a set of

[3] Those protocols were available to people such as myself over the public X.25 network; I worked at the time in a small firm, and had European money to connect to the network.

[4] Those on that list remark wryly about numbers of cooks and quality of broth...

[5] As private individuals: we don't consider the archive a service offered by UK TEX Users' Group, though we claim to 'support' it.

bibliographic references which can be updated anywhere with the knowledge that sooner or later every instance will become a pukkah copy — see, for example 2.

CTAN maintains its consistency by the 'archive user' sending email messages to the archive users at other archives when something new is installed, or something is replaced, moved or deleted.[6] The receiving archive user checks that the message is from a 'known' source, parses it, and translates it into a one-shot mirror operation. The Perl scripts to do all of this are maintained by Rainer Schöpf.

That procedure deals with the 'installations': when someone has submitted something to the `/incoming` directory of one of the archives, or when one of us has something to install (for example, all the updates to LaTeX 2ε itself are installed by Rainer in this way).

There are still archives of the 'old' sort: places where people make their own bits and pieces available for general access, but which don't offer a comprehensive collection. An example is Knuth's (and Tom Rokicki's, among others) stuff, kept on labrea.stanford.edu; there are many others, and there are many things that are kept in an archive related to another matter — for example, the NTeX implementation for Linux, lives on the general Linux archive at sunsite.unc.edu

For all these things, each of the CTAN nodes runs a considerable mirroring operation every night. Managing that mirror operation, and dealing with the installations and miscellaneous queries, takes a considerable portion of a person's time. For the three CTAN nodes at |ftp.tex.ac.uk|, |ftp.shsu.edu| and |ftp.dante.de|, there are three people to deal with all the work (the nominal manager of |ftp.shsu.edu|, George Greenwade, has taken no part in the work for more than a year). If the eight people running Aston at the start were spoiling a broth, I think it's reasonable to claim that three people for three archives are perhaps a little thinly spread (despite excellent systems support from Martyn Johnson at Cambridge, and a number of back-up workers helping Rainer at |ftp.dante.de|).

3 The archives from the users' viewpoint

I persist in viewing the archives primarily as sources of stuff to get by ftp, but the evidence suggests that they are as often as not accessed via Web browsers.

As anonymous ftp archives, CTAN nodes offer exactly the same structure. A rather deep tree of directories is accessed via the root |/tex-archive/|, and the structure of that tree is the same at all nodes. If a location on CTAN is quoted, such as |macros/eplain|, the common root is assumed.

CTAN stores very large numbers of individual files (for example, macro packages), which is good for those who want to browse, but bad for retrieving the files. Therefore, CTAN provides means to compress entire directories (or even directory trees) on-the-fly. Suppose, for example, I want to acquire the latest version of LaTeX for my DOS machine; I will (within ftp) change directory to |tex-archive/macros/latex|, and then `get base.zip`. There's no file called |base.zip|, but there is a directory |base|: CTAN will make a zip archive of the directory 'on the fly', and return that. The archive can also make |.tar.gz| and |.tar.Z| archives, but nothing specifically targeted at Mac users. It's worth noting that the most frequent cause of connections to CTAN failing is a user's optimism about how much he can pull in one go; a common one is to try and pull all of the `latex` tree, which gives you all the `packages` and (huge) `contrib` sub-trees, as well as `doc` and `unpacked` directories (which in a `.zip` archive simply repeat some or all of the `base` directory).

As the material on CTAN gradually approaches 2 Gbytes, how is the poor user to find her way around? Ideally, one would like some kind of advanced indexing software, but even within the rather restricted compass of an ftp connection CTAN can help. Each night, the archive examines its own navel: it produces sorted lists of all its files, and stores them in the archive itself; very often, one can gain an adequate clue to the location of a file by use of the command `quote site index` at the ftp prompt.[7] While a tool of more expressive power would be nice, the clever use of the arcane 'regular expressions' employed can find things with some precision (see question 23 in *Baskerville* 5.6).

Other means of accessing the files are:

- via NFS (fine for people with lots of bandwidth to the archive machine, such as sites on the SuperJanet backbone, but not otherwise terribly practical)
- via gopher; this is an entirely automatic mechanism — we've devoted little effort to it in Cambridge, and the evidence is that it's little used

[6] While this isn't as eccentric a proceeding as it would have been in the days when I first used email, one must admit it lacks a certain … fundamental sense. I am reminded of the Dilbert cartoon — which I failed to save from the Web — where the idiot manager suggests maintaining a database by email, and Dilbert and Co. fall about at his stupidity.

[7] Note that some ftp clients do not require the `quote`; check with the documentation.

This list is regularly updated. The link on the directory hierarchy (which is the relative location of the package on any CTAN archive/mirror) takes you to the CTAN mirror at Walnut Creek. The link on the flag (or the country code for browsers without images) takes you to the corresponding CTAN backbone. The link on the name of the package itself will load the packages actual documentation as a gziped postscript file (only for local users for now - although this will be fixed soon). Refer to my TeX Resources home page for other archives and information. Comments, corrections, enhancements, and additions are more than welcome (the HTML is built from a stylised database I maintain for myself but will probably evolve into the filehdr format). Similar lists (in English) that I am aware of are listed at the end.

A

abbrevs macros/latex/contrib/supported/monster/

Macros to expand abbreviations to text and insert the proper following space depending on context. These macros can also expand to one thing the first time they are used and another thing on subsequent invocations (to automatically spell-out abbreviations or acronyms at their first use). A generic facility is also provided for suffixes like 1900\BC and 6:00\PM, which correctly handles following periods.

Figure 1. The start of Graham Williams' Web page

- via mail: both `dante` and `shsu` offer an ftpmail interface; mail a message containing just '`help`' to `ftpmail@dante.de` (or at `shsu`) for details.

And there's the Web...

4 Access by WWW interfaces

The really big expansion of the media hype about the Internet has coincided with the explosion of the Web into people's consciousness. The Web is indeed a fine mechanism, particularly for those with lots of bandwidth, a big screen, and strong wrists and fingers, but it's nothing without information providers. Sadly, providing information in a useful and attractive form proves actually to be rather tricky; the manpower required to do it is not easily available to those of us running CTAN, so we tend to rely on people outside our numbers.

Norm Walsh 4 developed a mechanism that permits access to the 'normal' searching facilities of CTAN. People (particularly in the USA) speak well of it, but I've never had success with it (there's *never* enough bandwidth across the Atlantic). The URL is http://jasper.ora.com/ctan.html, and it has a series of menus for accessing the archive, as well as some searching mechanisms. The information content is derived automatically from directory listings of SHSU, so that (beyond the considerable effort of setting the thing up) there's little day-to-day work involved.

Another interesting mechanism is the (LA)TEX Navigator, which appears in three languages (French, English — http://www.loria.fr/tex/english/index.html — and German; the German portion is said not to be up to date). The range of information stored is enormous (it must represent a massive investment of effort); it's well worth a visit just to browse. One of its services is a CGI-script that searches the archives; this is a better interface than is `quote site index`, in that one can scroll or search through the information returned, but it's still not terribly informative.

More promising is Graham Williams' ftp://cbr.dit.csiro.au/staff/gjw/www/texpkgs.html; he is undertaking to index all the macro packages on CTAN to a fairly impressive level of detail (somewhat like David Jones' pre-CTAN index which is now, sadly, no longer maintained). I've included a short extract in figure 1, which gives some indication; that figure (though it contains some exciting flags at its top) is missing the flag-links to the CTAN sites themselves; by default, the package location points to `ftp.cdrom.com`

5 And for those with no network?

The Internet's a grand place . . . for those of us who are connected to it. Even when I'm at home, dialling in to a CTAN site, I find the archive tricky (and I know the layout better than most). How are the 'unconnected' to survive?

The obvious solution is to dump the archive's contents to CD-ROM. This is plainly do-able (though the whole archive now takes more even than two CDs). The problem is navigation: with the crazy restrictions of ISO 9660 CD directory format, file names lose what one might call 'expressive power'.

The problem is addressed by the 'targeted' CD — one which you can use as an installation source from the word go. The first of these was the NTG's excellent 4AllTEX CD for PCs, which combines a view of how the system ought to be run with a well-constructed, extensive set of backup archive material. A new one, whose structure is based on the TDS 'standard' and which uses the teTEX implementation of TEX, is being prepared for release in May 1996.

6 The future

CTAN, or something like it, will remain necessary for some time to come, but none of those involved would claim that it's entirely satisfactory as it stands. In particular, a 'world-wide' network of archives should consist of more than two European sites and one barely-functioning USAn one. Where are we on the Pacific rim?; why have we only in the last year gained our first mirror in Africa?

A straightforward first step would be to make Norm Walsh's mechanism available more widely; for wide usage, it should be available at all CTAN sites and at a representative selection of mirrors.

However, indexing and searching mechanisms have advanced massively over the last few years, and it would be nice to unleash the power of (say) Digital's AltaVista engine on the contents of CTAN. To do this, we need some criterion for indexing; even a well-documented TEX file can be expected largely to consist of gobbledygook, and finding the relevant stuff (in doc-package material, in running comments, or after `\endinput`) needs some careful heuristic work.

There is a comparable set of archives, called the CPAN, which holds Perl-related material (they do say that imitation is the sincerest form of flattery!). The CPAN people have constructed a script that will decide for you where you *should* have connected your Web-browser, and sends your connection off there. If you have access to a Web-browser, try connecting to |http://www.perl.com/CPAN/| — if I do it, I end up at a directory at |ftp://unix.hensa.ac.uk/mirrors/perl-CPAN//|, which is physically in a different continent. This is a neat trick, and we're working on something similar for CTAN. The script depends on your domain address, so that if you are one of those tricksy sites that is (say) in `.com` even though you're physically located in the UK and connected through UK service providers, you'll end up at the wrong place.

I mentioned above that there are better protocols to run what CTAN does. I (continue to) feel it would be nice to use some protocol other than email to maintain our consistency, but providing a complete suite of protocols (and getting it accepted!) is beyond the resources I have (notably the time resources. . .).

To close, at the meeting at Warwick, I asked for suggestions. I've presented all the ideas that came from the meeting; do readers have any?

References

Peter Abbott. UKTEX and the Aston archive. *TUGboat*, 10(1):59–60, April 1989.

Richard A. Golding, Darrell D. E. Long, and John Wilkes. The *refdbms* distributed bibliographic database system. In *Proceedings of the Winter Usenix Conference*, San Francisco, CA, January 1994.

George D. Greenwade. The Comprehensive TEX Archive Network (CTAN). *TUGboat*, 14(3):342–351, October 1993.

Norm Walsh. A World Wide Web interface to CTAN. *TUGboat*, 15(3):339–343, September 1994.

V typehtml: A LaTeX package to typeset HTML

David Carlisle

1 Introduction

This package enables the processing of HTML codes. The `\dohtml` command allows fragments of HTML to be placed within a LaTeX document,

```
\dohtml
<html>
html markup ...
</html>
```

The `<html>`...`</html>` is *required*. (It is anyway a good idea to have these tags in an HTML document.)

The `\htmlinput` command is similar, but takes a file name as argument. In that case the file need not necessarily start and end with `<html>`...`</html>`.

This package covers most of the HTML2 DTD, together with the mathematics extensions from HTML3.[8] The rest of HTML3 may be added at a later date.

Its current incarnation has not been extensively tested, having been thrown together during a couple of weeks in response to a question on `comp.text.tex` about the availability of such a package.

The package falls into three sections. Firstly the options section allows a certain amount of customisation, and enabling of extensions. Not all these options are fully operational at present. Secondly comes a section that implements a kind of SGML parser. This is not a real conforming SGML parser (not even a close approximation to such a thing!) The assumption (sadly false in the anarchic WWW) is that any document will have been validated by a conforming SGML parser before it ever gets to the stage of being printed by this package. Finally are a set of declarations that essentially map the declarations of the HTML DTD into LaTeX constructs.

2 Options

2.1 HTML Level

The options `html2` (the default) and `html3` control the HTML variant supported. Using the `html3` option will use up a lot more memory to support the extra features, and the math entity (symbol) names. Against my better judgement there is also a `netscape` option to allow some of the non-HTML tags accepted by that browser.

2.2 Headings

The six options `chapter`, `chapter*`, `section`, `section*`, `subsection` and `subsection*` determine to which LaTeX sectional command the HTML element `h1` is mapped. (`h2`–`h6` will automatically follow suit.) The default is `section*`.

2.3 Double Quote Handling

Most HTML pages use `"` as as a quotation mark in text, for example:

```
quoted "like this" example
```

This slot in the ISO latin-1 encoding is for ‘straight’ double quotes. Unfortunately the Standard TeX fonts in the OT1 encoding do not have such a character, only left and right quotes, “like this”. By default this package uses the `straightquotedbl` option which uses the LaTeX command `\textquotedbl` to render `"`. If used with the T1 encoded fonts `\usepackage[T1]{fontenc}` then the straight double quote from the current font is used. With OT1 fonts, the double quote is taken from the `\ttfamily` font, which looks "like this" which is fairly horrible, but better than the alternative which is ”like this”.

[8] The draft specification of HTML3 has expired, and the W3C group are currently devising a new proposed extension of HTML, so the mathematics typesetting part of this package may need substantial revision once a final specification of the HTML mathematics markup is agreed.

The `smartquotedbl` option redefines `"` so that it produces alternatively an open double quote “ then a close ”. As there is a chance of it becoming confused, it is reset to “ at the beginning of every paragraph, whatever the current mode.

Neither of these options affects the use of `"` as part of the SGML syntax to surround attribute values.

In principle the package ought to have similar options dealing with the single quote, but there the situation is more complicated due to its dual use as an apostrophe, so currently the package takes no special precautions: all single quotes are treated as a closing quote/apostrophe. Also the conventions of ‘open’ and ‘close’ quotes only really apply to English. If someone wants to suggest what the package should do with `"` in other languages...

2.4 Images

The default option is `imgalt` This means that all inline images (the HTML `img` element) are replaced by the text specified by the `alt` attribute, or [image] if no such attribute is specified. The `imggif` option[9] uses the `\includegraphics` command so that inline images appear as such in the printed version. The `imgps` option[9] is similar to `imggif` but first replaces the extension `.gif` at the end of the source file name by `.ps`. This will enable drivers that can not include GIF files to be used, as long as the user keeps the image in both PostScript and Gif formats.

2.5 Hyperref

Several options control how the HTML anchor tag is treated.

The default `nohyperref` option ignores `name` anchors, and typesets the body of `src` anchors using `\emph`.

The `ftnhyperref` option is similar to `nohyperref`, but adds a footnote showing the destination address of each link, as specified by the `src` attribute.

If the `hyperref` option is specified, the hypertext markup in the HTML file will be replicated using the hypertext specials of the HyperTEX group. If in addition the hyperref package is loaded, the extra features of that package may be used, for instance producing ‘native PDF’ specials for direct use by Adobe Distiller rather than producing the specials of the hyperTEX conventions.

The `dviwindo` option converts the hypertext information in the HTML into the `\special` conventions of Y&Y’s *dviwindo* previewer for Microsoft Windows.

2.6 Big Integrals

LaTeX does not treat integral signs as variable sized symbols, in the way that it treats delimiters such as brackets. In common with summation signs and a few other operators, they come in just two fixed sizes, a small version for inline mathematics, and a large version used in displays. In fact by default LaTeX always uses the same two sizes (from the 10 pt math extension font) even if the document class has been specified with a size option such as `12pt`, or if a size command such as `\large` has been used.

The standard exscale package loads the math extension font at larger sizes if the current font size is larger than 10 pt.

The HTML3 math description explicitly states that integral signs should be treated like delimiters and stretch if applied to a large math expression. By default this package ignores this advice and treats integral signs in the standard way, however an option `bigint` does cause integral signs to ‘stretch’ (or at least be taken from a suitably large font). The standard Computer Modern fonts use a very ‘sloped’ integral which means that they are not really suitable for being stretched. Some other math fonts, for instance Lucida, have more vertical integral signs, and one could imagine in those cases making an integral sign with a ‘repeatable’ vertical middle section so that it could grow to an arbitrary size, in the way that brackets grow.

3 Latin-1 characters

The SGML character entities for the ISO-Latin1 characters such as `&eacute;` are recognised by this style, although as usual, some of them such as the Icelandic thorn character, `&thorn;`, `\th`, produce an error if the old ‘OT1’ encoded fonts are being used. These characters will print correctly if ‘T1’ encoded fonts are used, for example by declaring `\usepackage[T1]{fontenc}`.

HTML also allows direct 8-bit input of characters according to the ISO-latin1 encoding, to enable this you need to enable latin-1 input for LaTeX with a declaration such as `\usepackage[latin1]{inputenc}`.

[9] one day...

4 Mathematics

The HTML3 `math` element is fairly well supported, including the `box` and `class` attributes. (Currently only `chem` value for class is supported, and as far as I can see the `box` attribute is only in the report, not in the DTD.) The super and subscripts are supported, including the shortref maps, however only the default right alignment is implemented so far. The convention described in the draft report for using white space to distinguish superscript positioning is fairly *horrible*!

The documentation that I could find on HTML3 did not include a full list of the entity names to be used for the symbols. This package currently *only* defines the following entities, which should be enough for testing purposes at least.

- `gt` (>) `lt` (<) (Already in the HTML2 DTD)
- Some Greek letters.
  `alpha` (α) `beta` (β) `gamma` (γ) `Gamma` (Γ)
- Integral and Sum. ∫ grows large if the `bigint` package option is given.
  `int` (∫) `sum` (∑)
- Braces (The delimiters ( )[ ] also stretch as expected in the `box` element)
  `lbrace` ({) `rbrace` (})
- A random collection of mathematical symbols:
  `times` (×) `cup` (∪) `cap` (∩) `vee` (∨) `wedge` (∧) `infty` (∞) `oplus` (⊕) `ominus` (⊖) `otimes` (⊗)
- A Minimal set of trig functions:
  `sin` (sin) `cos` (cos) `tan` (tan)
- Also in the special context as attributes to `above` and `below` elements the entities:
  `overbrace` (⏜) `underbrace` (⏟) and any (TEX) math accent name.

5 SGML Minimisation features

SGML (and hence HTML) support various minimisation features that aim to make it easier to enter the markup 'by hand'. These features make the kind of 'casual' attempt at parsing SGML as implemented in this package somewhat error prone.

Two particular features are enabled in HTML. The so called `shorttag` feature means that the name of a tag may be omitted if it may be inferred from the context. Typically in HTML this is used in examples like

```
<title>A Document Title</>
```

The end tag is shortened to `</>` and the system infers that `title` is the element to be closed.

The second form of minimisation enabled in HTML is the `omittag` feature. Here a tag may be omitted altogether in certain circumstances. A typical example is the HTML list, where each list item is started with `<li>` but the closing `</li>` at the end of the item may be omitted and inferred by the following `<li>` or `</ol>` tag.

This package is reasonably robust with respect to omitted tags. However it only makes a half hearted attempt at supporting the `shorttag` feature. The `title` example above would work, but nested elements, with multiple levels of minimised end tags will probably break this package.

It would be possible to build a LaTeX system that had full knowledge of the HTML (or any other) DTD and in particular the 'content model' of every element. This would produce a more robust parsing system but would take longer than I was prepared to spend... If you need a fully conforming SGML parser, it probably makes sense to use an existing one (excellent parsers are freely available) and then convert the output of the parser to a form suitable for LaTeX. In that way all such concerns about SGML syntax features such as minimisation will have been resolved by the time LaTeX sees the document.

6 Examples

6.1 A section

This document uses the `subsection*` option.

```
<h1>HTML and LaTeX</h1>
```

HTML and LaTeX

6.2 An itemised list

```
<ul>
<li> something
```

```
<li> something else
</ul>
```

- something
- something else

6.3 Latin1 Characters

```
&eacute; &ouml;
```

é ö

6.4 Images

Currently only the `alt` attribute is supported.

```
An image of me <img alt="DPC"
                src="dpc.gif">
```

This is an image of me DPC

6.5 A Form

```
<form
action=
   "http://www.cogs/cgi-bin/ltxbugs2html"
method=get><hr>
You can search for all the bug reports
about:
<select name="category">
 <option>AMS LaTeX</option>
 <option>Babel</option>
 <option>Graphics and colour</option>
 <option>LaTeX</option>
 <option selected>Metafont fonts</option>
 <option>PostScript fonts</option>
 <option>Tools</option>
</select>
<hr>
</form>
```

You can search for all the bug reports about:

category
AMS LaTeX
Babel
Graphics and colour
LaTeX
• Metafont fonts
PostScript fonts
Tools

6.6 Styles of Mathematics

```
<math>
H_2_O + CO_2_
</math>
<math class=chem>
H_2_O + CO_2_
</math>
<math box>
H_2_O + CO_2_
</math>
```

```
<math class=chem box>
H_2_O + CO_2_
</math>
```

$$H_2O + CO_2$$

$$\mathrm{H_2O + CO_2}$$

$$\boxed{H_2O + CO_2}$$

$$\boxed{\mathrm{H_2O + CO_2}}$$

6.7 Integrals

Stretchy integrals with the `bigint` option.

```
<math>
{&int;^1^_3_<left>
1 <over>
{x+{1<over>x+{2<over>x+
{3<over>x+{4<over>x}}}}}
<right>dx}
</math>
```

$$\int_3^1 \frac{1}{x+\frac{1}{x+\frac{2}{x+\frac{3}{x+\frac{4}{x}}}}}\mathrm{d}x$$

And the same integral with the standard integral sign.

$$\int_3^1 \frac{1}{x+\frac{1}{x+\frac{2}{x+\frac{3}{x+\frac{4}{x}}}}}\mathrm{d}x$$

6.8 Oversized delimiters

```
<math>
<box>
  (<left>1 <atop> 2 <right>)
</box>
<box size=large>
  (<left>1 <atop> 2 <right>)
</box>
</math>
```

$$\binom{1}{2}\left(\begin{matrix}1\\2\end{matrix}\right)$$

6.9 Roots, Overbraces etc

```
<math>
<above sym=overbrace>  abc </above>
<sup>k</sup>
 
<root>3<of>x</root>
<sqrt>5</sqrt>
```

```
 
<below sym=underline>  abc </below>
<above sym=widehat>  abc </above>
</math>
```

$$\overbrace{abc}^{k} \quad \sqrt[3]{x}\sqrt{5} \quad \underline{abc}\widehat{abc}$$

6.10 Arrays

Most of the array specification is supported. Currently most of the effort has gone into writing the HTML parser, so currently the column spacing is not yet ideal, as may be seen by the following examples, but that is (hopefully!) a small detail that can be corrected in a later release.

```
<math>
<array align=top>
<row><item><text>col 1</text>
     <item><text>col 2</text>
     <item><text>col 3</text>
     <item><text>col 4</text>
<row><item><text>row 2</text>
     <item> a_22_ <item>a_23_<item>a_24_
<row><item><text>row 3</text>
     <item rowspan=3
           colspan=2>a_32_-a_53_
     <item>a_34_
<row><item><text>row 4</text>
     <item>a_44_
<row><item><text>row 5</text>
     <item>a_54_
<row><item><text>row 6</text>
     <item align=left>al_62_
     <item align=right>ar_63_
     <item>a_64_
</array>
</math>
```

col 1	col 2	col 3	col 4
row 2	a_{22}	a_{23}	a_{24}
row 3	$a_{32} - a_{53}$		a_{34}
row 4			a_{44}
row 5			a_{54}
row 6	al_{62}	ar_{63}	a_{64}

Repeat that element, but change the `array` attributes as follows:

```
<array ldelim="(" rdelim=")" labels>
```

$$\begin{array}{c} \\ \text{row 2} \\ \text{row 3} \\ \text{row 4} \\ \text{row 5} \\ \text{row 6} \end{array} \begin{array}{c} \begin{array}{cccc} \text{col 1} & \text{col 2} & \text{col 3} & \text{col 4} \end{array} \\ \left(\begin{array}{cccc} a_{22} & a_{23} & a_{24} \\ a_{32} - a_{53} & & a_{34} \\ & & a_{44} \\ & & a_{54} \\ al_{62} & ar_{63} & a_{64} \end{array} \right) \end{array}$$

and finally an example of `colspec`

```
<math>
<array colspec="R+C=L">
<row><item>abc_11_<item>abc_12_
     <item>abc_13_
<row><item>a_21_<item>a_22_<item>a_23_
<row><item>a_31_<item>a_32_<item>a_33_
</array>
</math>
```

$$\begin{array}{rcrcl} abc_{11} & + & abc_{12} & = & abc_{13} \\ a_{21} & + & a_{22} & = & a_{23} \\ a_{31} & + & a_{32} & = & a_{33} \end{array}$$

6.11 Tables

HTML3 tables are not yet supported, but there is a minimal amount to catch simple cases.

```
<table>
<caption>Simple Table</caption>
<tr><td>one <td> two
<tr><td>a <td> b
</table>
```

Simple Table

one	two
a	b

7 Concluding Remarks

Some parts of this package are still rather 'rough'. In particular some of the spacing in the mathematics examples above is not perfect. I plan to revise the package and improve such details when (if?) a mathematics proposal for HTML to replace the HTML3 draft is published. Considering that it started off as an example just to show that TEX is capable of processing markup languages that do not look like the traditional 'backslash' commands, the package has proved surprisingly capable of handling a wide variety of 'real world' HTML documents. Of the core HTML language the most noticeable feature not yet supported is graphics inclusion. I plan to support that better in a future release.

A more difficult conceptual problem is that it is hard to linearise a hypertext document automatically. A typical 'document' will consist of many HTML files interconnected by links. Currently one must invoke `\dohtml` or `\htmlinput` separately on each of these files, and manually order them into a page order for the typeset version. It would be nice to develop heuristics to traverse the HTML document and build up the linear typeset version automatically; however TEX may not be the ideal language for writing a web-crawler…

VI A date for your diary

David Carlisle

The Annual General Meeting of the UK TEX Users Group will be held on October 16th at the University of Warwick. The AGM will include a review of new developments, including a report back from the TUG Dubna conference, and the chance to win a UK TEX Archive 10th anniversary t-shirt (original and still the best!).

VII An introduction to PSTricks, part 2

Sebastian Rahtz
Elsevier Science Ltd
Email: s.rahtz@elsevier.co.uk

1 Introduction

In the first part of this description of PSTricks, we looked at the basic concepts of the package, a series of low-level building blocks, and the useful commands for dealing with text and the third dimension. Now it is time to look at the higher-level packages built in PSTricks, for drawing trees and graphs. There are a wide variety of applications, as I hope the examples show. This part of the package is, unfortunately, extremely rich, and readers should not be surprised if they find the plethora of new commands rather confusing.

2 Nodes and their connections, and trees

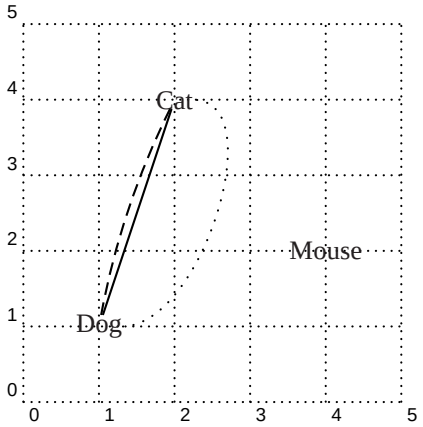
PSTricks offers sophisticated macros for setting up named nodes and joining them together in complicated ways, complete with labels.

The nodes can be created in three ways:

1. By placing them at arbitrary coordinates
2. By placing them on a regular grid or matrix
3. By using higher-level tree macros

The sort of effect we will create is like this:

```
\rput(1,1){\rnode{A}{Dog}}
\rput(2,4){\rnode{B}{Cat}}
\rput(4,2){\rnode{C}{Mouse}}
\ncline{A}{B}
\nccurve[linestyle=dotted]{A}{B}
\ncarc[linestyle=dashed]{A}{B}
```



Every node is given a symbolic name, which is used for node connectors (lines, curves and so on), and for node labels. The fact that nodes can occur *anywhere* (such as in running text), makes it possible to use them for surprising effects like linking two words one ~~to another.~~ When considering connectors, we have to distinguish between the position of the node they are pointing *towards* (the node reference point), and the actual extent of the connector. Most node creators have an invisible box around them, which determines the end of connector lines. We will first list all the commands, and then give practical examples.

Table 1 lists all the node creation commands and node connector commands, Table 3 lists the commands to label connectors and nodes, and Table 2 lists the extra graphical parameters which apply to node connectors.

Table 1: PSTricks node drawing commands

Node creators	
`\rnode`*[*refpoint*]*{name}{text}	create a node called *name*, consisting of *text*; connectors point to the *refpoint*
`\Rnode`*(x,y)*{name}{text}	the same as `\rnode`, but the reference point is the middle of the box's baseline, plus (*x* ,*y*)
`\pnode`*(x,y)*{name}	create a node at (*x*,,*y*,) which takes up no space
`\cnode`**[*settings*](x,y)*{radius}{name}	create a node consisting of circle of *radius*
`\Cnode`**[*settings*](x,y)*{name}	create a node consisting of circle, using the radius set by the *radius* graphical parameter (it is sometimes useful to set the radius for many circles)
`\cnodeput`**[*settings*]{*angle*}(x,y)*{name}{text}	
`\circlenode`**[*settings*]*{name}{text}	like `\pscirclebox`, but makes a node
`\ovalnode`**[*settings*]*{name}{text}	like `\psovalbox`, but makes a node
`\dianode`**[*settings*]*{name}{text}	like `\diabox`, but makes a node
`\dotnode`**[*settings*](x,y)*{name}{text}	like `\psdot`, but makes a node
`\fnode`**[*settings*](x,y)*{name}{text}	like `\psframe`, but makes a node
`\trinode`**[*settings*]*{name}{text}	like `\tribox`, but makes a node
Node connectors	
`\ncline`**[*settings*]{*arrows*}{firstnode}{secondnode}	straight line between nodes
`\ncLine`**[*settings*]{*arrows*}{firstnode}{secondnode}	straight line between nodes, but labels are placed as if the line went right to the center of the nodes
`\ncarc`**[*settings*]{*arrows*}{firstnode}{secondnode}	arc between nodes; uses parameter *arcangle*
`\ncdiag`**[*settings*]{*arrows*}{firstnode}{secondnode}	using the *arm* and *angle* parameters, 'arms' start out from each node and are then joined by a line; corner shape is controlled by the *linearc* parameter
`\ncdiagg`**[*settings*]{*arrows*}{firstnode}{secondnode}	as `\ncdiag`, but the second arm is not drawn
`\ncbar`**[*settings*]{*arrows*}{firstnode}{secondnode}	a line is drawn with arms coming off at a right angle to the nodes, at an angle of *angleA*; the arm length is adjusted if necessary
`\ncangle`**[*settings*]{*arrows*}{firstnode}{secondnode}	draws a connect line `\ncdiag`, but the angle between arm A and the connector line is forced to a right angle
`\ncangles`**[*settings*]{*arrows*}{firstnode}{secondnode}	like `\ncangle`, but arm A is joined to arm B by two line segments that meet at a right angle
`\ncloop`**[*settings*]{*arrows*}{firstnode}{secondnode}	like `\ncangles` but 5 line segments are used, the second and forth being *loopsize* long
`\nccurve`**[*settings*]{*arrows*}{firstnode}{secondnode}	bezier curve between nodes, using the *ncurv* parameters to determine the control point positions
`\nccircle`**[*settings*]{*arrows*}{node}{radius}	draws a circle or part circle of radius *radius* connecting the node to itself
Coil and zigzag node connectors	
`\nccoil`**[*settings*]{*arrows*}{firstnode}{secondnode}	
`\nczigzag`**[*settings*]{*arrows*}{firstnode}{secondnode}	

Table 2: PSTricks Graphical parameters for node connectors

Parameter	*Default*	*Explanation*
offset=dim	0pt	the offset of the connection point to a node
nodesep=dim	0pt	the border around nodes at which connectors stop
nodesepA=dim	0pt	the border around the first node
nodesepB=dim	0pt	the border around the second node
arcangle=angle	8	in `\ncarc`, the angle between the arc and a straight line drawn between the nodes
angle=angle	0	the angle at which connectors hit the nodes
angleA=angle	0	the angle at which a connector hits the first node
angleB=angle	0	the angle at which a connector hits the second node
arm=dim	10pt	the length of the line segment where the connector joins the nodes
armA=dim	10pt	the length of the line segment where the connector joins the first node
armB=dim	10pt	the length of the line segment where the connector joins the second node
loopsize=dim	1cm	the length of line segments for `\ncloop`
ncurv=num	0.67	the distance to Bezier control points in `\nccurve`; lower values give tighter curves; the distance from the node to the first control point is half *ncurv* $\times$ the distance between the two end points
ncurvA=num	0.67	as *ncurv* but for first node only
ncurvB=num	0.67	as *ncurv* but for second node only
boxsize=dim	0.4cm	half the width of the enclosing box of `\ncbox` and `\ncarcbox`
Parameters for node labels		
ref=ref	c	sets the reference point for labels
nrot=rot	0	the rotation of label text; if the angle is preceded by :, it is measured with respect to the connector line; the letter abbreviations we have already seen are available, so :U is commonly used to align text on the connector line
npos=num		the position along the length of the connector line where a label is placed; each connector line has one or more segments, and the value of *npos*+1 determines the segment on which the label is set; the default values for this parameter are given in the PSTricks manual, but can also be seen in the examples below
shortput=none/nab/tablr/tab	none	determines whether short codes are available for labelling connectors; see page 27.

PSTricks Graphical parameters for node connectors *cont.*

Parameter	*Default*	*Explanation*
tpos=num	0.5	the proportion of the distance between nodes at which labels are placed on a connector
mnode=type	R	(for matrices) the default node type; possibilities are R (`\Rnode`), r (`\rnode`), C (`\Cnode`), f (`\fnode`), p (`\pnode`), circle (`\circlenode`), oval (`\ovalnode`), dia (`\dianode`), tri (`\trinode`), dot (`\dotnote`), and none
emnode=type	none	(for matrices) the type of node created for empty cells in a matrix
name=name		(for matrices) the name of a node; parameters like this can set in square brackets in the cell
nodealign=true/false	false	(for matrices) whether baselines of nodes pass through the centre of nodes
mcol=l/r/c	c	(for matrices) the alignment of a node within a matrix cell
mnodesize=dim	−1pt	(for matrices) is positive, nodes are forced to be this size
rowsep=dim	1.5cm	(for matrices) the gap between rows
colsep=dim	1.5cm	(for matrices) the gap between columns

Table 3: PSTricks node connection labelling commands

Labelling based on connector length	
`\ncput*[`settings`]{`something`}`	place *something* on the connector line
`\naput*[`settings`]{`something`}`	place *something* above the connector line
`\nbput*[`settings`]{`something`}`	place *something* under the connector line
Labelling based on distance between nodes	
`\tvput*[`settings`]{`something`}`	working on the vertical distance between nodes, place *something* in the middle of the line
`\tlput*[`settings`]{`something`}`	working on the vertical distance between nodes, place *something* to the left of the line
`\trput*[`settings`]{`something`}`	working on the vertical distance between nodes, place *something* to the right of the line
`\thput*[`settings`]{`something`}`	working on the horizontal distance between nodes, place *something* in the middle of the line
`\taput*[`settings`]{`something`}`	working on the horizontal distance between nodes, place *something* above the line
`\tbput*[`settings`]{`something`}`	working on the horizontal distance between nodes, place *something* below the line
Labelling nodes	
`'[par]'{angle}{name}{something}`	place *something* next to the node, at a distance of *nodesep*, in the direction *angle* from the centre of the node

Table 4. PSTricks drawing commands comparable to node connectors

`\pcline*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pccurve*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pcarc*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pcbar*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pcdiag*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pcangle*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pcloop*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pczigzag*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`
`\pccoil*[`settings`]{`arrows`}(`*x1*`,`*y1*`)(`*x2*`,`*y2*`)`

There are several important concepts we need to bear in mind when looking at the myriad of node and connector commands:

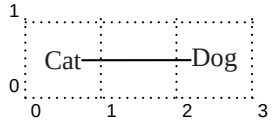
1. When joining two nodes with something like a curve, the connectors comes in by default on the right hand side of the object (at 0 degrees). If we have two boxes side by side, the *angleB* parameter has to be set to 180 if we want the connector to come into the second box on its left side. This may seem cumbersome at first, but it makes for a flexible system;
2. the connector labelling commands all place the label some proportion of the way along the connector, but they distinguish between the distance between nodes, and the length of the line. The former situation applies to constructions like matrices, where we want the label positions to be constant, regardless of the size of the nodes.

3. The node connectors are not drawn directly in TEX, but are done at the PostScript level; this means that TEX is not always quite sure how much space will be taken up by the object. Particularly when curving connectors are drawn, you might find that they protrude outside the area allowed by TEX— adjust by hand.

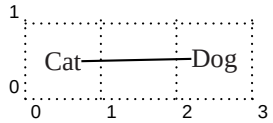
Because labelling node connectors is a very common thing to do, a short cut is provided to save all the `\naput` commands etc. If the parameter *shortput* is set to `nab`, the `^` is used instead of `\naput` and _ instead of `\nbput`. If it is set to `tablr`, the `^` stands for `\taput`, _ for `\tbput`, `<` for `\tlput` and `>` for `\trput`.

If all this were not enough, all the node connectors can also be used as ordinary drawing tools, by using the commands listed in Table 4, where the 'pc' version corresponds to the 'nc' node connector.

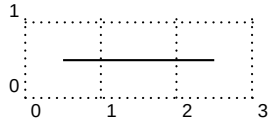
Let us first demonstrate the effects of these basic building blocks:



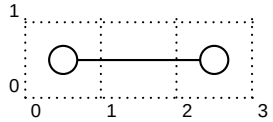

```
\rput(.5,.5){\rnode{A}{Cat}}
\rput(2.5,.5){\rnode{B}{Dog}}
\ncline{A}{B}
```



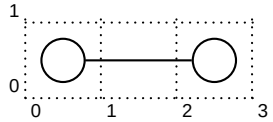

```
\rput(.5,.5){\Rnode{A}{Cat}}
\rput(2.5,.5){\Rnode{B}{Dog}}
\ncline{A}{B}
```



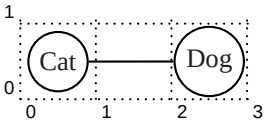

```
\pnode(.5,.5){A}
\pnode(2.5,.5){B}
\ncline{A}{B}
```



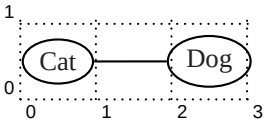

```
\cnode(.5,.5){.2}{A}
\cnode(2.5,.5){.2}{B}
\ncline{A}{B}
```



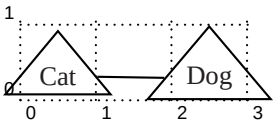

```
\psset{radius=.3}
\Cnode(.5,.5){A}
\Cnode(2.5,.5){B}
\ncline{A}{B}
```



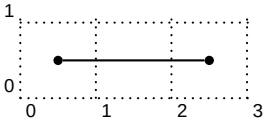

```
\rput(.5,.5){\circlenode{A}{Cat}}
\rput(2.5,.5){\circlenode{B}{Dog}}
\ncline{A}{B}
```



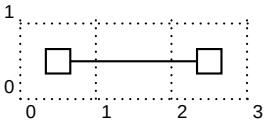

```
\rput(.5,.5){\ovalnode{A}{Cat}}
\rput(2.5,.5){\ovalnode{B}{Dog}}
\ncline{A}{B}
```



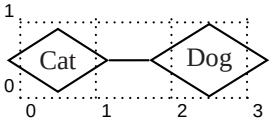

```
\rput(.5,.5){\trinode{A}{Cat}}
\rput(2.5,.5){\trinode{B}{Dog}}
\ncline{A}{B}
```



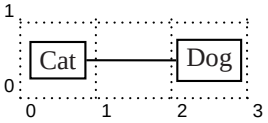
```
\dotnode(.5,.5){A}
\dotnode(2.5,.5){B}
\ncline{A}{B}
```



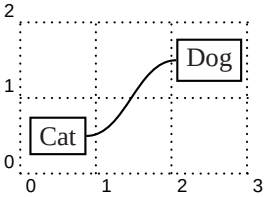
```
\fnode(.5,.5){A}
\fnode(2.5,.5){B}
\ncline{A}{B}
```



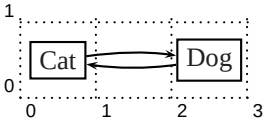

```
\rput(.5,.5){\dianode{A}{Cat}}
\rput(2.5,.5){\dianode{B}{Dog}}
\ncline{A}{B}
```



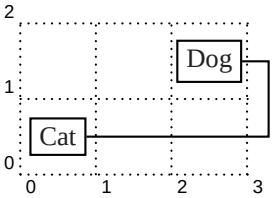

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,.5){\rnode{B}{\psframebox{Dog}}}
\ncline{A}{B}
```



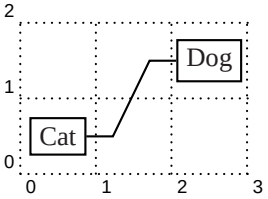

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\nccurve[angleB=180]{A}{B}
```



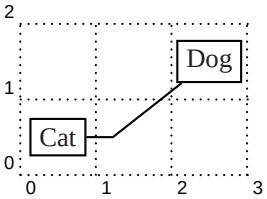

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,.5){\rnode{B}{\psframebox{Dog}}}
\ncarc{->}{A}{B}
\ncarc{->}{B}{A}
```



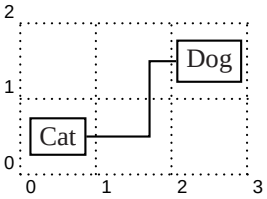

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\ncbar{A}{B}
```



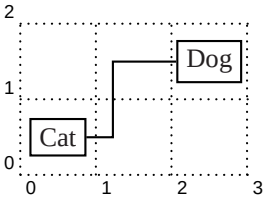

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\ncdiag[angleB=180]{A}{B}
```



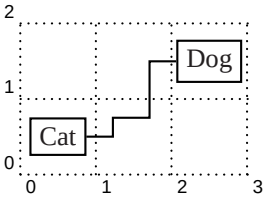

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\ncdiagg[angleB=180]{A}{B}
```



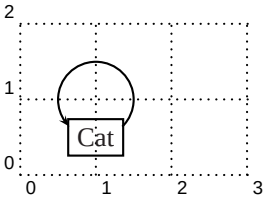

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\ncangle[angleB=180]{A}{B}
```



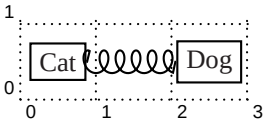

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\ncangles[angleB=180]{A}{B}
```



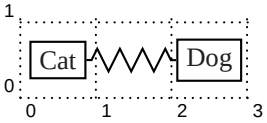

```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,1.5){\rnode{B}{\psframebox{Dog}}}
\ncloop[loopsize=.25,angleB=180]{A}{B}
```




```
\rput(1,.5){\rnode{A}{\psframebox{Cat}}}
\nccircle{->}{A}{.5}
```




```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,.5){\rnode{B}{\psframebox{Dog}}}
\psset{coilarm=.01,coilwidth=.3}
\nccoil{A}{B}
```

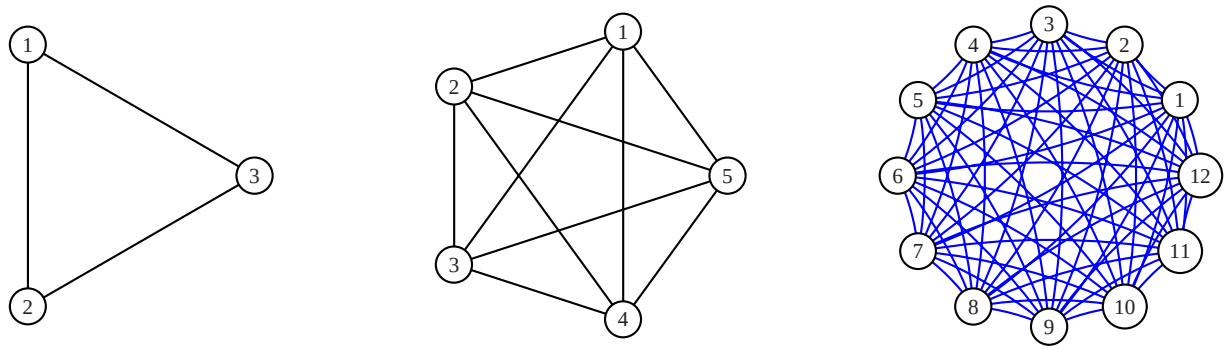



```
\rput(.5,.5){\rnode{A}{\psframebox{Cat}}}
\rput(2.5,.5){\rnode{B}{\psframebox{Dog}}}
\psset{coilarm=.01,coilwidth=.3}
\nczigzag{A}{B}
```

The effect of node connectors is demonstrated in the following, which uses the `\multido` macro (we will look at that in a future article) to place objects at regular intervals around a circle, and join them up

```
\newcount\CtA
\newcount\CtB
\newcommand{\Wheel}[3]{{%
\pspicture(-1,-1)(1,1)
\SpecialCoor
\degrees[#1]
\multido{\ia=1+1}{#1}{%
  \CtA=\ia
  \advance\CtA by 1
  \CtB=#1
  \advance\CtB by -\ia
  \multido{\ib=\CtA+1}
     {\CtB}{#3(1;\ia)(1;\ib)}}
```

```
\multido{\i=1+1}{#1}{%
  \rput(1;\i){%
   \pscirclebox[fillstyle=solid,
       fillcolor=white]%
    {\footnotesize\i}}}
\endpspicture}}
\makebox[\columnwidth][s]{%
\psset{unit=2cm}
\Wheel{3}{1.8}{\psline}
\Wheel{5}{1.8}{\psline}
\psset{arcangle=10}
\Wheel{12}{3}{\pcarc[linecolor=blue]}}
```



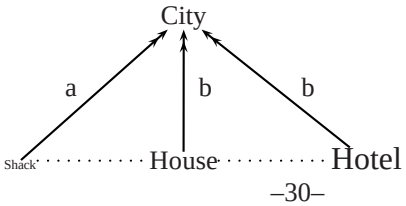
It should be clear that we can draw arbitrary diagrams, trees etc simply by working out the coordinates of each node; however, in practice, there are two higher-level environments for easier creation of nodes — matrices and trees.

2.1 Matrices — grid-based nodes

The existing LaTeX `tabular` or AMS LaTeX `matrix` can be used to place nodes, but PSTricks provides its own environment, `psmatrix`. This is like an easy form of table, since the number of columns does not have to be specified — we simply separate column items by `&` and rows by `\\` as normal, and PSTricks makes each cell a node, named as *rownumber*,*columnnumber*. Thus the first node in the first row is named `1,1` and the third node in the fourth row is `4,3`, and these are used by the node connectors.

`\psmatrix` has an optional parameter in which we can set *rowsep* and *colsep*, determining the gap between nodes. The parameter *shortput* is set to *tab* inside `\psmatrix` by default, so we can adopt a quite succinct notation:

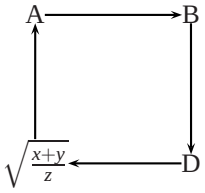
```
\begin{psmatrix}[rowsep=1.5cm]
&City&\\
{\tiny Shack} & House & {\Large Hotel}
\psset{arrows=<<-}
\ncline{1,2}{2,1}<{a}
\ncline{1,2}{2,2}>{b}
\ncline{1,2}{2,3}>{b}
\psset{arrows=-,linestyle=dotted}
\ncline{2,1}{2,2}
\ncline{2,2}{2,3}
\end{psmatrix}
```



Notice in this example that the shorthand '>' stands for `\trput`, which places labels according to the distance between node centres, not the connection length, which allows for the difference in sizes of nodes in the second row. Nodes can span multiple columns by using the `\psspan` command at the end of the cell, with a parameter of the number of columns to span.

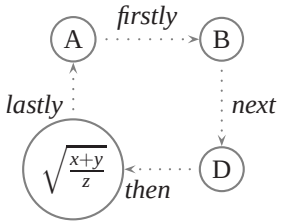
A simple example of a square matrix can be created as:

```
\begin{psmatrix}
A & B \\
$\sqrt{\frac{x + y}{z}}$ & D
\ncline{->}{1,1}{1,2}
\ncline{->}{1,2}{2,2}
\ncline{->}{2,2}{2,1}
\ncline{->}{2,1}{1,1}
\end{psmatrix}
```

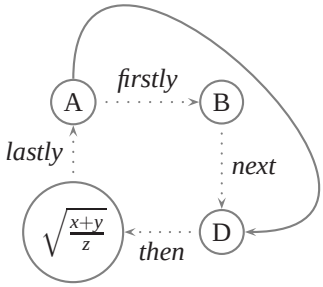


but by changing a few initial parameter settings, we can present a 'fancier' result, with nodes encircled, arrows on connectors, and a better spacing. The connector labels are added with the short-hand forms, which are those which are positioned in relation to node centres. In the second version, we redo the labels with the label types which relate to line length, which in this case gives a better result. The outer looping connector is an example of a construct whose extent TEX will probably guess incorrectly.

```
\psset{arrows=->,labelsep=3pt,
linecolor=gray,mnode=circle,shortput=nab}
\begin{psmatrix}[rowsep=20pt,colsep=28pt]
A & B \\
$\sqrt{\frac{x + y}{z}}$ & D
\psset{linestyle=dotted}
\ncline{1,1}{1,2}^{\emph{firstly}}
\ncline{1,2}{2,2}>{\emph{next}}
\ncline{2,2}{2,1}_{\emph{then}}
\ncline{2,1}{1,1}<{\emph{lastly}}
\end{psmatrix}
```

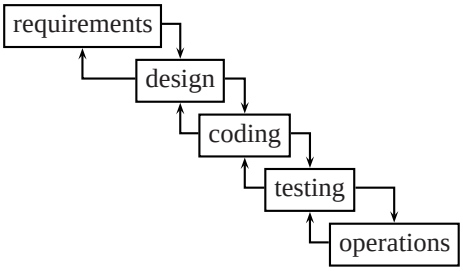



```
\psset{arrows=->,labelsep=3pt,
linecolor=gray,mnode=circle}
\begin{psmatrix}[rowsep=20pt,colsep=28pt]
A & B \\
$\sqrt{\frac{x + y}{z}}$ & D
\psset{linestyle=dotted}
\ncline{1,1}{1,2}\naput{\emph{firstly}}
\ncline{1,2}{2,2}\naput{\emph{next}}
\ncline{2,2}{2,1}\naput{\emph{then}}
\ncline{2,1}{1,1}\naput{\emph{lastly}}
\nccurve[ncurv=2,linestyle=solid,angleA=90]
         {1,1}{2,2}
\end{psmatrix}
```



Matrices can be nested, and it is possible to link nodes from two different matrices, if the nodes are given explicit names. Each `\psmatrix` wipes out the current set of *row,column* names.

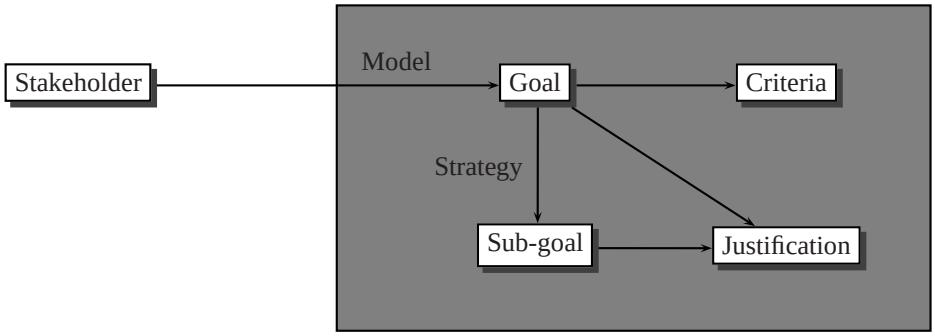
```
\psset{linearc=.2}
\begin{psmatrix}[rowsep=3pt,colsep=-10pt]
[name=A]\psframebox{requirements}\\
&[name=B]\psframebox{design}\\
&&[name=C]\psframebox{coding}\\
&&&[name=D]\psframebox{testing}\\
&&&&[name=E]\psframebox{operations}
\psset{linearc=0,arrows=->,armA=0pt,angleB=90}
\ncangle{A}{B}
\ncangle{B}{C}
\ncangle{C}{D}
\ncangle{D}{E}
\psset{angleB=-90,angleA=180}
\ncangle{B}{A}
\ncangle{C}{B}
\ncangle{D}{C}
\ncangle{E}{D}
\end{psmatrix}
```



A normal low-level PSTricks command, like `\framebox`, can be applied to a whole matrix. We have to take some care in this example with alignment to make the connecting line horizontal, so we place the single node on the left in its own matrix.

```
\psset{fillcolor=white,fillstyle=solid}
\def\Show#1{\psshadowbox{#1}}
\psset{arrows=->}
\begin{psmatrix}
  [mnode=r,ref=t]
   \psframebox[linestyle=none,framesep=.75]{%
    \psset{ref=c}
    \begin{psmatrix}
    [name=A]\Show{Stakeholder}
    \end{psmatrix}
       } &
  [mnode=r,ref=t]
   \psframebox[fillstyle=solid,framesep=.75,fillcolor=gray]{%
  \psset{ref=c}
  \rule{1cm}{0pt}
  \begin{psmatrix}
  [name=B]\Show{Goal} &\Show{Criteria}\\
          \Show{Sub-goal} & \Show{Justification}
          \ncline{1,1}{1,2}
          \ncline{1,1}{2,2}
```

```
          \ncline{1,1}{2,1}\tlput{Strategy}
          \ncline{2,1}{2,2}
  \end{psmatrix}
  }
\ncline[angleB=-180]{A}{B}\naput[npos=.7]{Model}
\end{psmatrix}
```



In addition, PSTricks has an extremely rich environment for drawing trees, permitting complex structures and presentation. This will form the subject of the next part of this series.

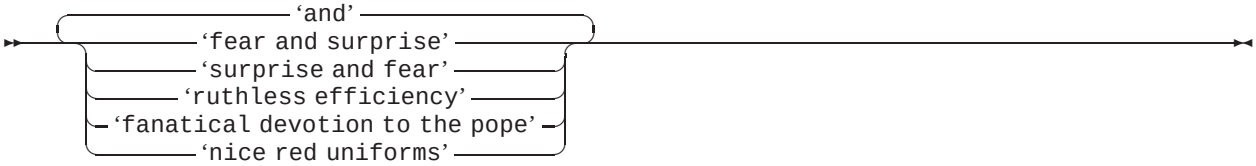
VIII Syntax diagrams and other fun

Mark Wooding
Email: mdw@straylight.co.uk

1 Introduction

Among other things, I write manuals for computer programs, and descriptions of syntax diagrams tend inevitably to creep in. Formal BNF grammars are relatively easy to typeset using some simple definitions and a list-based environment. However, they can be rather daunting for less technical readers, containing as they do all manner of funny metasymbols.[10] Books attempting to cater for such readers (and even some technical manuals about C programming) tend to present syntax using diagrams: the idea, if you haven't come across them already, is that you follow the lines on the diagram around writing any of the items you come across on your journey, until you reach the end.

For example, the diagram below attempts to present the various diverse elements comprising the arsenal of the Spanish Inquisition, according to the now legendary Monty Python sketch.



The diagram permits any sequence of one or more 'weapons', separated by the word '`and`'. Compare this with the BNF equivalent:

⟨*arsenal*⟩ ::= ⟨*weapon*⟩ | ⟨*arsenal*⟩ '`and`' ⟨*weapon*⟩

⟨*weapon*⟩ ::= '`fear and surprise`'
| '`surprise and fear`'
| '`ruthless efficiency`'
| '`fanatical devotion to the pope`'
| '`nice red uniforms`'

Which do you think is easier to read?

The syntax package provides some commands for typesetting syntax diagrams like the one above (and for typesetting BNF grammars).

2 Building syntax diagrams

Syntax diagrams are typeset using the syntdiag environment. This puts LaTeX into a special 'syntax diagram' mode: only syntax diagram commands should be used while in this mode: anything else will upset the typesetting and produce output which looks truly awful. To help you lay out your source nicely, spaces and newlines (including blank newlines) are totally ignored within syntax diagrams. (They will be enabled again when spaces and newlines are actually useful, so there's no need to worry about this.)

One important point about syntax diagrams must be made here: like the verbatim environment, the syntdiag environment cannot be used inside the argument of a command. The other point is that you can only use syntax diagrams when you're in paragraph mode – there's a syntdiag environment which is designed for use in LR mode, although that's more oriented towards presenting fragments of diagrams.

If you just write an empty syntdiag environment, you get a line across the current text column with a double headed arrow on each end, like this:

This is clearly not much good, and we need to learn how to make it more interesting.

[10] Even the word 'metasymbol' is a little scary.

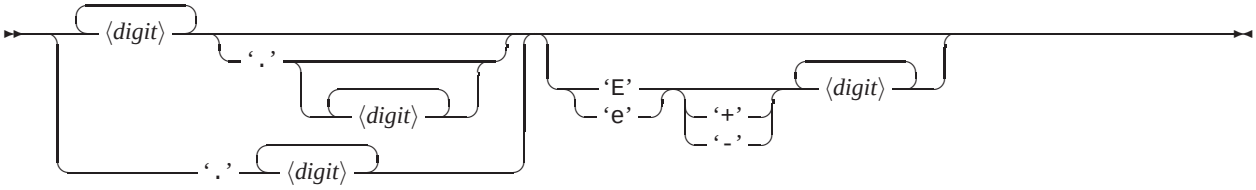


Figure 1. Syntax diagram for a floating point number

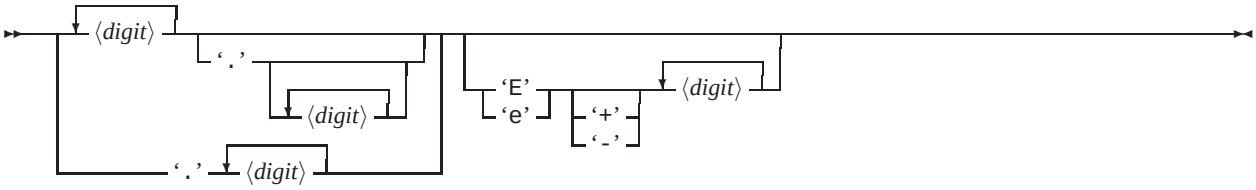


Figure 2. The same diagram with in the 'square' style

3 Simple bits of syntax diagrams

Syntax diagrams are built up from a small number of simple blocks. You just need to learn how to put these blocks together to build quite complicated looking diagrams.

The simplest things you can put in syntax diagrams are *syntax objects*. There are three types of syntax objects built in, and they have special abbreviations because they get used so much. The three types are:

Nonterminals stand for some (possibly fairly complicated) syntactic entity. They look like ⟨*this*⟩[11] in syntax diagrams. You type nonterminals by typing the text within '`<...>`', like `<this>`.

Terminals describe text which should be typed in exactly as shown. They look like '`this`' (or possibly like `this`) in the diagram. You can type a quoted terminal by just typing single quotes around the text, like `'this'`; unquoted terminals like the second example are obtained by using double quotes, like `"this"`.

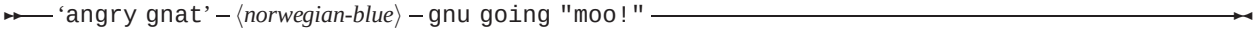
Composites can contain terminals and nonterminals, together with normal text typed in LaTeX's LR mode. You type a composite object by using the `\tok` command – the argument contains the text of object. Within this argument, you can type nonterminals and terminals using the abbreviations described above. Also, the '`|`' character will produce a | symbol, which is used to indicate alternatives.

The text within a nonterminal or terminal is read in way similar to LaTeX's `\verb` command. You type the text more or less exactly as you'd like it to look. Problems occur if you want to produce something like '`doesn't`' – if you type `'doesn't'`, then TeX will think that the text ends at the *first* '`'`' character. You can avoid this problem by preceding the offending character by a backslash, making `'doesn\'t'`, which is unfortunately less readable, but at least it actually works. Similar considerations apply to the '`"`' and '`>`' characters. There's no need to start escaping characters like this unless there's actually an ambiguity. You can also type '`\\`' to obtain a backslash character, and '`\␣`' to get the '␣' symbol.

Right: we can now put something into our empty syntax diagram. By saying

```
\begin{syntdiag}
  'angry gnat'  <norwegian-blue>
     "gnu going \"moo!\""
\end{syntdiag}
```

we get something which looks not dissimilar to this:



[11] Well, you can change the style so that they look like anything you want, although this is how they look by default, and I'd recommend that you don't change the style too radically, because you'll confuse your readers utterly. The styles of the other syntax objects can also be changed.

4 Building larger structures

Once you can write simple syntax objects, you need to know how to combine them in interesting ways. You've already seen that just writing objects next to each other inserts them in series (I managed to sneak that in above). Other contructions are described by environments nested within the main syntdiag environment.

A choice of one item from a list is represented by stacking the available options one above the other. Such a construction is typeset using the stack environment; the individual rows of the stack are separated by \\ commands.

An item which may be repeated any number of times is shown with a loop above it. Such a structure can be typeset with the rep environment; the text to be repeated forms the body of the environment. If the repeated texts are to be separated from each other, you can describe this separating text by preceding it with the \\ command.

These two environments can be nested within each other as required. Arbitrarily complicated structures can be built in this way. We now have enough equipment to start drawing real diagrams. For example, a floating point number might be represented like this:

```
\begin{syntdiag}
\begin{stack}
  \begin{rep} <digit> \end{rep}
  \begin{stack} \\
    '.' \begin{stack} \\
      \begin{rep} <digit> \end{rep}
    \end{stack}
  \end{stack} \\
  '.' \begin{rep} <digit> \end{rep}
\end{stack}
\begin{stack} \\
  \begin{stack} 'E' \\ 'e' \end{stack}
  \begin{stack} \\ '+' \\ '-' \end{stack}
  \begin{rep} <digit> \end{rep}
\end{stack}
\end{syntdiag}
```

(The output is shown in figure 1.)

You can probably think of all sorts of syntax diagrams which can't be drawn by this system. Such diagrams are rather unlikely to appear in practice, they tend to be harder to understand, and all of them can be transformed into something which *can* be represented, without altering their meanings.

If you find that you can't draw the diagram you want, you should first see if it can't be transformed fairly simply into something which you can draw. If that doesn't work, try complaining at the author of the package.

5 Other things

Syntax diagrams have a habit of getting rather large. When possible, this package will try to break long syntax diagrams over several lines. Usually, a diagram begins with a double arrow (like '⋯'), and ends with a funny pair of inward pointing arrows (like '⋯'). When a diagram is broken, single arrows are put on the broken ends as a visual clue that the lines aren't complete. Quite often, the TEX will find fairly good line breaks, and most of the time, the breaks will be fairly reasonable. If you want to take matters into your own hands, you can use the \\ command to force a break.

Finally, there are two different styles of syntax diagrams which can be drawn by the package – there's the 'rounded' style I used in figure 1, and the 'square' style shown in figure 2. You can choose which style to use in your document by passing the option '`square`' or '`rounded`' when you load the package. The default is to draw 'square' style diagrams.

IX Plug and play CD

Sebastian Rahtz

GUTenberg and UKTUG, in collaboration with TUG and NTG, are producing a plug-and-play CD-ROM based on Thomas Esser's teTEX distribution. It adheres to the TDS (TEX Directory Structure) tree structure (see, for instance, TUGboat 16(4), pages 401–412).

Presently Unix executables for the following platform/operating system combinations are included: Digital alpha-osf (2.0 and 3.2), Hewlett Packard hpux (9.01 and 10.01), Intel i386 bsdi2.0, freebsd (2.0.5 and 2.1.0) netbsd (1.0 and 1.1), Intel i486 (linux and linuxaout), m68k (linux, linuxoldld, and nextstep3), mips (irix 5.2, 5.3 and ultrix4.4) IBM rs6000 (aix3.2 and aix4.1.1) Sparc Solaris (2.4 and 2.5) and Sunos 4.1.3.

As the CD-ROM uses the ISO 9660 standard, the platform-independent files can, in principle be read on all operating systems which are compatible with that format. To preserve the complete Unix/Posix file system information the file tree was recorded with the Rock Ridge extensions, so that long filenames are honoured. Some operating systems, most notably MS-DOS, do not support these extensions, so that only the ISO 9660 filenames are used, but with this limitation, the files are readable on all systems. At present the complete LATEX Dec. 1995, patch level 2 base and all contributed files are included, plus the most-used font families (CM, DC 1.2, TC, AMS, Euler, psnfss). We plan to publish an updated CD-ROM twice a year (more or less coordinated with new LATEX releases), and we hope to add more packages and fonts in the next issues.

The complete source of the teTEX package is included, which has the source of TEX, METAFONT, and all the TEX utilities. Documentation for all of the macro packages on the CD-ROM is included in DVI and PDF format, as well as the original source. A large fraction of the documentation has also been converted to HTML.

The CD-ROM will be officially presented on May 29th in Nanterre (Paris), where GUTenberg organises a seminar on TEX distributions. The CD will be available to members of TEX user groups at £15 and to non-members at £25, through the normal channels.

X Malcolm's Gleanings

Malcolm Clark

1 Hints and allegations

Have you booked your trip to Dubna yet? Despite my own personal reservations I'm sure this will turn out to be one of the great TUG conferences, honest. Trust me. The sorts of efforts which are being made to ensure that all the delegates are welcomed when they arrive in Moscow and conveyed to Dubna seem to indicate that this is going to be a very effective and friendly affair. Go for it. It will be something to tell your grandchildren about.

A colleague (David Wright) recently spent a few weeks in Moscow. I asked him for his comments, since they may be applicable to those who manage to attend the conference. He writes:

"I had a great time – but I was lucky in having the option of staying with friends who have a good flat in one of the more privileged areas. I stayed for 2 nights in a student house – pretty scummy and with a bathroom replete with cockroaches and a rat/mouse which lived behind the toilet. I was there as a student however and they do look after non-student guests better – but you should be prepared for pretty basic accommodation and then anything better will be a pleasant surprise. This is a good rule of thumb for all areas of life in the former SU.

Something you may notice is that Russians have a casual approach to safety – the only way to be if you are permanently living in an environment where things are fairly out of control. For visitors it is worth being fairly cautious – particularly with respect to traffic which is anarchic. From observation and what I have heard, emergency services are rudimentary or non-existent and many Russian hospitals are pretty run down – so even with good medical insurance, you might find that initial treatment is tough. Having said that, life becomes pretty normal after a while and you can't spend your time wondering whether the Metro or buses are getting the requisite preventive maintenance – they clearly aren't. You could pack a geiger counter if you are staying on the site of former or current reactors. . .

Transport is very good actually – you can travel as far as you like on the Metro for 1500 roubles (about 5,000 roubles to the dollar, so about 20p) - to do so you buy 'jeton' which you feed into the barrier. On buses, trams, etc you can buy 10 tickets from the driver (a 'knitchka' *lit.* small book) for 19,000 roubles – valid on all versions (tram, trolleybus or bus) and for any length of journey. To validate one for a trip you stamp it in one of the coded punches in the vehicle.

An absolute must to visit is the refurbished Tretyakof Gallery which is the centrepiece of Russian art – but it is worth reading a bit or getting a guide unless you are the rare Westerner who knows anything about Russian art. The Pushkin also currently has two exhibitions (Moscow-Berlin 1900–1950 – excellent, covering art, architecture, drama, film, etc. *and* of course the new Trojan Gold (which opened the day I left). An annexe to the Pushkin which holds several private collections is also well worth a visit and for the really art mad there is also the House of Artists near Gorki Park (also a nice place to stroll in warm weather) which houses an extension to the Tretyakof and numerous other transient exhibitions.

The Kremlin is fairly essential (both for the churches and the treasures – which I didn't see, as I left some things for another trip), but if one wants a weird experience and is interested the Museum of the Revolution is it – two of us went – paid about 30p to get in and were the only people there (apart from the old ladies guarding every room). It is likely to disappear at least in this form so it will soon itself be history.

Other places which are worth a go are the Bolshoi Theatre – not the greatest in terms of performances but relatively cheap – best tickets about 75,000 roubles. Don't fall for the ticket touts (anywhere) – attendances by Russians have fallen off greatly because most have to prioritise expenditure fairly ruthlessly and the new rich have not yet matured into great patrons of the arts. The Conservatoire is interesting for its place in music history, but by modern standards the hall is not acoustically great (a bit like a school hall with pretensions). The music is good and varied, though with quite a lot of more modern Russian music.

For the really authentic Russian art experience, however, try the theatre – lots of them – a very good one is the Mali near the Bolshoi which has been refurbished recently. You have to be prepared not to understand much so a Russian version of a play you know is probably best. The *Cheek by Jowl* theatre company were in Moscow when I was there, so if you want a Russian/Western experience there are sometimes such options.

If you hate busy city life, try Suskova – a preserved palace in the Moscow suburbs (South East), which has an interesting palace (wooden!), a lake a park, many folly/pavilions a couple of which house glass and porcelain collections. It is, though, also well worth walking round the back streets of Moscow, especially near the North West quarter of the inner garden ring (road). In spite of years of destruction there is still quite a lot left of what must have been a rather beautiful city.

Food is a problem in Moscow – not many good places. To get Russian fast food (pies basically, but don't try the synthetic borsch) try the Russian Bistros. For a good Russian meal (100,000 roubles and upwards each) there is a restaurant in the basement of one of the old university building in the centre of Moscow (a bit hard to find though).

Changing money: you should take all your money ideally in dollars (in a money belt!) and change about $100–150 a time. There are many places to do this (all marked 'obmen valouti'). Some are pretty heavy places because they are protected against frequent raids, but they are ok. You should get a receipt of the transaction in case you are asked to show them at the airport when leaving (I wasn't). If they ask a question when you hand over you money, passport and visa – you should carry both of the latter at all times – just say 'da' – they are sure to be asking whether you want a receipt. *Very important* – take new notes (insist at the bank that they give you these) – many exchange places refuse even slightly worn/soiled notes and there is only one central official exchange place where one can be certain that any note will be changed. This is not a fantasy: some people in our group had real problems because they couldn't change some of their money.

At Sheremetevo airport expect the worst – this is surely amongst the least welcoming of all airports in the world. We had a long wait at immigration (over an hour) and baggage appears to come randomly off the carousels. After a while it is just dumped anywhere on the floor of the baggage hall. When you leave, you go through customs before checking in your bags – and you need to find the (unmarked) area where you get and fill in a customs declaration before you join a queue. In general most bureaucracy in Russia is just obstructive, however, rather than really threatening…

Good luck."

2 Urgent Newsflash: Origins of LaTeX

Another correspondent (RAR) sent me this towards the end of March, beginning of April. It has the uncomfortable ring of truth to it:

"U.S. Government Department of Defense papers released yesterday reveal the true source of the LaTeX document preparation tool. Under the twenty-five year rule, hitherto secret Military Intelligence papers have been released into the public domain, detailing the Government involvement in the development of a prototype software tool.

Major General Charles Schultz publicly apologised at a press meeting held to defuse what was described as a 'tense situation' as the unsavoury origins of the 'alleged software' were finally unearthed. Developed at the height of the Cold War, the prototype system was aimed at crippling the so-called 'information net' within the Soviet high command. Maj. Gen. Schultz said, 'We realised what would happen if an enemy organisation actually tried to create documents with this thing. Of course, it seems cold-blooded now, but you have to remember that this was at a time of war.' He added, 'We never expected it to get out into general circulation'.

Apparently, the software was released into a controlled environment under the cover of a beta-test at several academic sites in Europe and America. 'We put in place all safety measures. However, we didn't fully appreciate the tenacity of academic staff in using and then illegally copying it into the wider environment. With hindsight, we should have taken a warning from the fact that they still thought Fortran was a good idea.'

DoD boffins are rumoured to have panicked and released a beta-version of Emacs, the strangely popular editor-cum-operating-system in order to stem the spread of LaTeX. However, Maj. Gen. Schultz declined to comment."

Had Schultz gone further and revealed that Unix was a similar escaped virus, I would have been inclined to believe this. Still, a close friend maintains that the CIA was the major sponsor of Pollock and Rothko (among other major non-figurative artists) in order to confuse and alarm those pesky Russkies. It all depends on whether you believe in the conspiracy or cock-up theory of politics.

3 Fonts

Gleanings from the Web: Adobe and Microsoft have made an agreement to create a new font format called OpenType which will combine TrueType and Type 1 into a single new font format. This new format will use Adobe's new font compression to create compact fonts. Details can be found at the following URL: www.microsoft.com/truetype/fontpack/opentype.htm