Making PDF files from LaTeX

There are several schemes for producing PDF output from LaTeX, each having their own merits (and problems).

The main issues are fonts, which will scale correctly when viewed on screens and graphics support.

1 Using `pdflatex`

The simplest scheme is to process the source LaTeX file with `pdflatex`, which directly produces a PDF file with sensible and reasonable font selection which work equally well with text and equations. The main problem is with graphics, in particular *only* PDF graphics files are supported.

To include PDF graphics, the simplest scheme is to use the `graphicx` package, with

```
\usepackage{graphicx}
```

and them use `\includegraphics` to include the graphics, for example

```
\begin{center}
    \includegraphics[height=60mm]{GraphicsFile.pdf}
\end{center}
```

will centre the graphics file `GraphicsFile.pdf` scaled to 60 mm in height.

Note this scheme does *not* work `latex` or `pslatex`. Also `pdflatex` does not generate a `dvi` file, so the results cannot be viewed with `xdvi`, or processed with `dvips` etc.

2 Using `dvipdf`

The next alternative is to make a suitable `dvi` file and then convert it into `pdf`. The problem here is fonts, in particular the default bit-mapped `crm` fonts in LaTeX do not work well and result in `pdf` files that do not display well on the screen. The solution is to use `pslatex` which runs LaTeX but using scalable POSTSCRIPT fonts.

The route is now:

```
pslatex file
dvipdf file
```

will generate `file.pdf` using scalable POSTSCRIPT fonts.

This route works with included POSTSCRIPT graphics using the normal `epsfig` or `graphicx` packages, and produces good `pdf` files that work well both on-screen and printed.

The known problems are:

1. The scalable POSTSCRIPT fonts are *not* as prefect at the native LaTeX fonts in complex equations, and some equations need some manual spacing to make them look nice.
2. In equations, *v* (`$v$`) and ν (`$\nu$`) are *very* similar.
3. Some, fairly exotic, characters are missing for example **bold** capital Greek fails, ■ should be `$\bf\Psi$`. You will get an warning message about `missing glyph`.

This scheme used absolutely standard `tex` files that can be processed using `latex` and also generated standard `dvi` files that can be viewed using `xdvi`.

3 Using `dvips`

The final and most "long winded" scheme is to make a suitable `dvi` file, then convert to POSTSCRIPT and then finally convert to `pdf`. This is identical to `dvipdf` but allows you to pre-processing the POSTSCRIPT for example to generate "2-up" format with `psnup`. As above you *must* use `pslatex` to make the `dvi` file to force the use of the POSTSCRIPT scalable fonts.

The simples route is now

```
pslatex file
dvips -f file | ps2pdf - file.pdf
```

which uses `dvips` as a "filter", and piped this to `ps2pdf`. Note carefully the syntax of `ps2pdf` the `-` *must* have a space on either side.

Once the file is `pslatex`ed then you can then pre-process the POSTSCRIPT file between the `dvips` and `ps2pdf` stages, for example

```
dvips -f overheads | psnup -4 | ps2pdf - overheads.pdf
```

will produce a output `pdf` file in "4-up" format.

Again since we are initially using `pslatex` it works with totally standard LaTeX files and POSTSCRIPT graphics, however it does **not** fix the font problems 1 → 3 noted above.

Summary

Clearly there is no "ideal" scheme, my best advice is:

1. Complex mathematics and few/no diagrams, use scheme 1), but remember you will have to convert / save any diagrams as `pdf` before you start.
2. Large number of POSTSCRIPT diagrams, use scheme 2), purists will want to edit some formula, and watch out for black squares.
3. If 2) fails *or* want to do multi-page, then use 3), but watch out for the same font problems as 2).

Will Hossack
May 13, 2005