The pstool package

Will Robertson and Zebb Prime

v0.2   2008/08/03

Contents

Part I
Documentation

1   Introduction

While pdfLaTeX is a great improvement in many ways over the 'old method' of dvi→ps→pdf, it loses the ability to interface with a generic PostScript workflow, used to great effect in numerous packages, most notably PSTricks and psfrag.

Until now, the best way to use these packages while running pdfLATEX has been to use the pst-pdf package, which processes the entire document through a filter, sending each relevant PostScript environment through DVI→PS→PDF. The resulting PDF versions of each image are then included into the pdfLATEX document. The auto-pst-pdf package provided a wrapper to execute this separate process automatically.

The disadvantage in this route is that for every document compilation, *every* graphic must be re-processed. The pstool package uses a different approach to allow each graphic to be processed only as-needed, speeding up and simplifying the typesetting of the main document.

2   Processing modes

The main command provided by this package is
$$\psfig[\langle \textit{graphicx options}\rangle]\{\langle \textit{filename}\rangle\}\{\langle \textit{graphic data}\rangle\}.$$
By default it can be used in the following modes:

\psfig Process the graphic ⟨*filename*⟩ if no PDF of the same name exists, or the source EPS file is *newer* than the PDF;

\psfig* Always process this figure; and,

\psfig! Never process this figure.

The following package options override the above: [process=all], [process=none] (the default is [process=auto]).

3   Cropping graphics

Graphics are cropped to the appropriate size with the preview package. Sometimes, however, this will not be good enough when an inserted label protrudes from the natural bounding box of the figure. A good way to solve this problem is to use the pdfcrop program (requires a Perl installation under Windows). This can be activated in pstool with the [pdfcrop] package option.

4   Todo

1. Use dir /h & call echo ^%ERRORLEVEL^%.

2. Higher commands (\psfragfig, \matlabfig, \mathfig)

3. Generalise "olding" code for multiple files.

4. Basic EPS→PDF processing (no need to read in the document preamble).

5. Check for correct behaviour in shells other than bash.

6. More flexible usage (support things like \begin{postscript} in pst-pdf).

7. mylatex integration

Part II
Implementation

```
1  \ProvidesPackage{%
      pstool}[2008/08/03␣v0.2␣Wrapper␣for␣processing␣PostScript/psfrag␣figures]
```

External packages

```
2  \RequirePackage{%
      catchfile,color,ifpdf,ifplatform,inversepath,graphicx,suffix,xkeyval}
```

Initialisations

```
\if@pstool@always@    3  \newif\if@pstool@always@
 \if@pstool@never@    4  \newif\if@pstool@never@
\if@pstool@pdfcrop@   5  \newif\if@pstool@pdfcrop@
       \pstool@out    6  \newwrite\pstool@out
```

Package options

```
        pdfcrop    7  \DeclareOptionX{pdfcrop}{\@pstool@pdfcrop@true}

        process    8  \define@choicekey*{pstool.sty}{process}[\@tempa\@tempb]{%
                       all,none,auto}{%
                   9    \ifcase\@tempb\relax
                  10      \@pstool@always@true
                  11    \or
                  12      \@pstool@never@true
                  13    \or
                  14    \fi}

                  15  \ifshellescape\else
                  16    \ExecuteOptionsX{process=none}
                  17    \PackageWarning{pstool}{^^J\space\space%
                  18      Package␣option␣[process=none]␣activated^^J\space\space
                  19      because␣-shell-escape␣is␣not␣enabled.^^J%
                  20      This␣warning␣occurred}
                  21  \fi

                  22  \ProcessOptionsX
```

These are cute:

<code>
\OnlyIfFileExists   23 \providecommand\OnlyIfFileExists[2]{\IfFileExists{#1}{#2}{}}
\NotIfFileExists   24 \providecommand\NotIfFileExists[2]{\IfFileExists{#1}{}{#2}}
</code>

Command line abstractions between platforms:

<code>
25 \edef\pstool@cmdsep{\ifwindows\string&\else\string;\fi\space}
26 \edef\pstool@rm{\ifwindows␣del␣\else␣rm␣--␣\fi}
</code>

<code>
\pstool@try@rm   27 \newcommand\pstool@try@rm[1]{%
                 28   \begingroup
                 29   \@for\@tempa:=#1\do{%
                 30   \OnlyIfFileExists{\ip@directpath\@tempa}{%
                 31   \immediate\write18{%
                 32   cd␣"\ip@directpath"\pstool@cmdsep
                 33   \pstool@rm␣"\@tempa"}}}%
                 34   \endgroup}
</code>

Generic function to execute a command on the shell and pass its exit status back into LATEX. Any number of \pstool@exe statements can be made consecutively followed by \pstool@endprocess, which also takes an argument. If *any* of the shell calls failed, then the execution immediately skips to the end and expands \pstool@error instead of the argument to \pstool@endprocess.

<code>
\pstool@exe   35 \def\pstool@exe#1{%
              36   \immediate\write18{%
              37   cd␣"\ip@directpath"\pstool@cmdsep
              38   \pstool@writestatus{#1}%
              39   }%
              40   \pstool@retrievestatus\@tempa
              41   \ifnum\@tempa␣>␣0
              42   \PackageWarning{pstool}{%
                        Execution␣failed␣during␣process:^^J␣␣#1^^J}%
              43   \expandafter\pstool@abort
              44   \fi}
</code>

Edit this definition to print something else when graphic processing fails.

<code>
\pstool@error   45 \def\pstool@error#1{\fbox{\color{red}%
                      \ttfamily␣An␣error␣occured␣processing␣this␣graphic.}}
</code>

<code>
\pstool@abort   46 \def\pstool@abort#1\pstool@endprocess{\pstool@error}
                47 \let\pstool@endprocess\@firstofone
</code>

It is necessary while executing commands on the shell to write the exit status to a temporary file to test for failures in processing. #1 & echo %ERRORLEVEL%

4

doesn't return the correct value inside a \write18 in Windows, so we have to do something different there.

```
\pstool@statusfile    48  \def\pstool@statusfile{status-deleteme.txt}
\pstool@writestatus   49  \def\pstool@writestatus#1{%
                      50    \ifwindows
                      51      echo␣0␣>␣\pstool@statusfile\pstool@cmdsep
                      52      #1␣\detokenize{||}␣echo␣1␣>␣\pstool@statusfile
                      53    \else
                      54      #1\pstool@cmdsep␣echo␣$?␣>␣\pstool@statusfile
                      55    \fi
                      56  }
```

Read the exit status from the temporary file and delete it.

```
\pstool@retrievestatus  57  \def\pstool@retrievestatus#1{%
                        58    \pstool@touchstatus
                        59    \CatchFileEdef{#1}{\ip@directpath\pstool@statusfile}{}%
                        60    \pstool@try@rm{\pstool@statusfile}%
                        61  }
```

This ensures the file is written to disk properly (allowing it to be read by \CatchFileEdef). Not necessary on Windows, whose file writing is evidently more crude/immediate.

```
\pstool@touchstatus   62  \def\pstool@touchstatus{%
                      63    \ifwindows\else
                      64      \immediate\write18{%
                      65        cd␣"\ip@directpath"\pstool@cmdsep
                      66        touch␣\pstool@statusfile}%
                      67    \fi
                      68  }
```

4.1   File age detection

Use ls (or dir) to detect if the EPS is newer than the PDF:

```
\pstool@datefiles    69  \def\pstool@datefiles{%
                     70    \edef\pstool@filenames{\ip@lastelement.eps\space␣%
                                \ip@lastelement.pdf\space}%
                     71    \immediate\write18{%
                     72      cd␣"\ip@directpath"\pstool@cmdsep
                     73      \ifwindows
                     74        dir␣/T:W␣/B␣/O-D␣"\ip@lastelement.eps"␣%
                                  \ip@lastelement.pdf"␣>␣\pstool@statusfile
                     75      \else
```

```
76        ls␣-t␣"\ip@lastelement.eps"␣"\ip@lastelement.pdf"␣>␣%
                \pstool@statusfile
77      \fi
78    }%
79    \pstool@retrievestatus\@tempa
80    \ifx\@tempa\pstool@filenames
81      \@tempswatrue
82    \else
83      \@tempswafalse
84    \fi
85  }
```

5 Command parsing

User input is \psfig (with optional * or ! suffix) which turns into one of the
following three macros depending on the mode.

\pstool@alwaysprocess
```
86  \newcommand\pstool@alwaysprocess[3][]{%
87    \inversepath*{#2}% calculate filename, path & inverse path
88    \pstool@process[#1]{#2}{#3}}
```

\pstool@neverprocess
```
89  \newcommand\pstool@neverprocess[3][]{%
90    \includegraphics[#1]{#2}}
```

For regular operation, which processes the figure only if the command is
starred, or the PDF doesn't exist.

\pstool@maybeprocess
```
91   \newcommand\pstool@maybeprocess[3][]{%
92     \inversepath*{#2}% calculate filename, path & inverse path
93     \IfFileExists{#2.pdf}{%
94       \pstool@datefiles
95       \if@tempswa\expandafter\@firstoftwo
96         \else\expandafter\@secondoftwo\fi{%
97           \pstool@process[#1]{#2}{#3}%
98         }{%
99           \includegraphics[#1]{#2}}%
100    }{%
101      \pstool@process[#1]{#2}{#3}%
102    }}
```

5.1 User commands

Finally, define \psfig as appropriate for the mode:

```
103  \ifpdf
```

```
104    \if@pstool@always@
105      \let\psfig\pstool@alwaysprocess
```
\psfig `106      \WithSuffix\def\psfig!{\pstool@alwaysprocess}`
\psfig* `107      \WithSuffix\def\psfig*{\pstool@alwaysprocess}`
```
108    \else\if@pstool@never@
109      \let\psfig\pstool@neverprocess
```
\psfig `110      \WithSuffix\def\psfig!{\pstool@neverprocess}`
\psfig* `111      \WithSuffix\def\psfig*{\pstool@neverprocess}`
```
112    \else
113      \let\psfig\pstool@maybeprocess
```
\psfig `114      \WithSuffix\def\psfig!{\pstool@neverprocess}`
\psfig* `115      \WithSuffix\def\psfig*{\pstool@alwaysprocess}`
```
116    \fi\fi
117  \else
118    \let\psfig\pstool@neverprocess
```
\psfig `119    \WithSuffix\def\psfig!{\pstool@neverprocess}`
\psfig* `120    \WithSuffix\def\psfig*{\pstool@neverprocess}`
```
121  \fi
```

6   The figure processing

\pstool@process
```
122  \newcommand{\pstool@process}[3][]{%
123    \pstool@write@processfile{#1}{#2}{#3}%
124    \pstool@exe{latex␣-shell-escape␣-output-format=dvi
125        -interaction=batchmode␣"\ip@lastelement-process.tex"}%
126    \pstool@exe{dvips␣"\ip@lastelement-process.dvi"}%
127    \if@pstool@pdfcrop@
128      \pstool@exe{ps2pdf␣"\ip@lastelement-process.ps"␣"%
             \ip@lastelement-process.pdf"}%
129      \pstool@exe{pdfcrop␣"\ip@lastelement-process.pdf"␣"%
             \ip@lastelement.pdf"}%
130    \else
131      \pstool@exe{ps2pdf␣"\ip@lastelement-process.ps"␣"%
             \ip@lastelement.pdf"}%
132    \fi
133    \pstool@endprocess{\includegraphics[#1]{#2}}}
```

The file that is written for processing is set up to read the preamble of the
original document and set the graphic on an empty page (cropping to size is
done either here with preview or later with pdfcrop).

\pstool@write@processfile
```
134  \def\pstool@write@processfile#1#2#3{%
135      \immediate\openout\pstool@out␣#2-process.tex\relax
136      \immediate\write\pstool@out{%
137        \unexpanded{%
```

```
138          \pdfoutput=0% force DVI mode if not already
```

Input the main document; redefine the document environment so only the preamble is read:

```
          139          \let\origdocument\document
\document 140          \def\document{\endgroup\endinput}%
          141       }\noexpand
          142          \input{\ip@inversepath\jobname}%
```

Now the preamble of the process file: (restoring document's original meaning)

```
143          \if@pstool@pdfcrop@\else
144            \noexpand\usepackage[active,tightpage]{preview}
145          \fi
146          \unexpanded{%
147            \let\document\origdocument
148            \pagestyle{empty}% remove the page number
149            \begin{document}
```

And the document body to place the graphic on a page of its own:

```
150          \centering\null\vfill}%
151          \if@pstool@pdfcrop@\else
152            \noexpand\begin{preview}%
153          \fi
154            \unexpanded{#3}% this is the "psfrag" material
155            \noexpand\includegraphics[#1]{\ip@lastelement}%
156          \if@pstool@pdfcrop@\else
157            \noexpand\end{preview}%
158          \fi
159          \unexpanded{%
160            \vfill\end{document}}%
161          }%
162       \immediate\closeout\pstool@out}
```

⟨*eof*⟩