

The pstool package

Will Robertson and Zebb Prime

v0.5 2008/08/08

Contents

I	Documentation	1
1	Introduction	2
2	Processing modes	2
3	Package options	3
3.1	Cropping graphics	3
3.2	Temporary files & cleanup	4
4	Todo	4
II	Implementation	4
4.1	File age detection	7
5	Command parsing	8
5.1	User commands	9
6	The figure processing	9
7	User commands	12

Part I

Documentation

1 Introduction

While pdfL^AT_EX 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 pdfL^AT_EX has been to use the pst-pdf package, which processes the entire document through a filter, sending the relevant PostScript environments through a single pass of DVI→PS→PDF. The resulting PDF versions of each image are then included into the pdfL^AT_EX document. The auto-pst-pdf package provides a wrapper to perform all of this automatically.

The disadvantage with this method 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 generic command provided by this package is

$$\backslash\mathrm{pstool}[\langle\mathrm{graphicx\ options}\rangle]\{\langle\mathrm{filename}\rangle\}\{\langle\mathrm{input\ definitions}\rangle\}$$

It converts the graphic $\langle\mathrm{filename}\rangle.\mathrm{eps}$ to $\langle\mathrm{filename}\rangle.\mathrm{pdf}$ through a unique DVI→PS→PDF process for each graphic, using the preamble of the main document. The resulting graphic is then inserted into the document, with optional $\langle\mathrm{graphicx\ options}\rangle$. The third argument to $\backslash\mathrm{pstool}$ allows arbitrary $\langle\mathrm{input\ definitions}\rangle$ (such as $\backslash\mathrm{psfrag}$ directives) to be inserted before the figure as it is processed.

By default $\backslash\mathrm{pstool}$ can be used in the following modes:

$\backslash\mathrm{pstool}$ Process the graphic $\langle\mathrm{filename}\rangle$ if no PDF of the same name exists, or if the source EPS file is *newer* than the PDF;

$\backslash\mathrm{pstool}^*$ Always process this figure; and,

$\backslash\mathrm{pstool}!$ Never process this figure.

The package accepts options to override the above:

[process=auto] This is the default as described above;

[process=all] All \pstool graphics are processed regardless of suffix; and,

[process=none] No \pstool graphics are processed.¹

It is useful to define higher-level commands with \pstool for including specific types of EPS graphics that take advantage of psfrag. As an example, this package defines the following commands. These commands all support the * or ! suffices.

`\epsfig[⟨opts⟩]{⟨filename⟩}` Insert a plain EPS figure. It is more convenient than using, for example, the `epstopdf` package since it will regenerate the PDF if the EPS file changes.

`\psfragfig[⟨opts⟩]{⟨filename⟩}` Insert an EPS file with psfrag definitions contained within the file `⟨filename⟩-psfrag.tex`. (Accepts an optional braced argument as shown next.)

`\psfragfig[⟨opts⟩]{⟨filename⟩}{⟨input definitions⟩}` Insert an EPS file with psfrag definitions contained either/or within the file `⟨filename⟩-psfrag.tex` and supplied by the third argument `⟨input definitions⟩`.

`\laprintfig[⟨opts⟩]{⟨filename⟩}` Insert figures that have been produced with MATLAB's `laprint` package.

`\mathpsfragfig[⟨opts⟩]{⟨filename⟩}` Insert figures that have been produced with MATHEMATICA's `MathPSfrag` package. Automatically adds the `-psfrag` suffix to the `⟨filename⟩`.

3 Package options

3.1 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.

¹If `pstool` is loaded in a L^AT_EX document in DVI mode, this is the option that is used since no external processing is required for these graphics.

3.2 Temporary files & cleanup

Each figure that is processed spawns an auxiliary L^AT_EX compilation through $\text{DVI} \rightarrow \text{PS} \rightarrow \text{PDF}$. This process is named after the name of the figure with a suffix; the default is `[suffix={-process}]`. All of these suffixed files are “temporary” in that they may be deleted once they are no longer needed.

As an example, if the figure is called `ex.eps`, the files that are created are `ex-process.tex`, `ex-process.dvi`, The `[cleanup]` package option declares via a list of extensions which temporary files are to be deleted after processing.

By default, this is `[cleanup={tex,dvi,ps,pdf,log,aux}]`. Choose `[cleanup={}]` to delete none of the temporary files (useful for debugging).

4 Todo

1. Test `\laprint`, `\psfragfig`, `\matlabfig`, `\mathfig`, especially with figures in a relative path.
2. Generalise “process if older” code for multiple files.
3. Support optional *⟨input definitions⟩* for all user commands??
4. (Maybe) support `epstool` for cropping the graphics.
5. Direct support for `\includegraphics` with EPS files.
6. Check for correct behaviour in shells other than bash.
7. More flexible usage (support things like `\beginpostscript` in `pst-pdf`).
8. mylatex integration.

Part II

Implementation

```
1 \ProvidesPackage{pstool}[2008/08/08_v0.5
2   Wrapper_for_processing_PostScript/psfrag_figures]
```

External packages

```
3 \RequirePackage{%
4   catchfile,color,ifpdf,ifplatform,
```

```
5 inversepath,graphicx,suffix,xkeyval}
```

Initialisations

```
\if@pstool@always@ 6 \newif\if@pstool@always@
\if@pstool@never@ 7 \newif\if@pstool@never@
\if@pstool@pdfcrop@ 8 \newif\if@pstool@pdfcrop@
\if@pstool@nopreamble@ 9 \newif\if@pstool@nopreamble@
\if@pstool@nofig@ 10 \newif\if@pstool@nofig@
\pstool@out 11 \newwrite\pstool@out
```

Package options

```
pdfcrop 12 \DeclareOptionX{pdfcrop}{\@pstool@pdfcrop@true}

process 13 \define@choicekey*{pstool.sty}{process}[\@tempa\@tempb]{%
    all,none,auto}{%
14 \ifcase\@tempb\relax
15 \@pstool@always@true
16 \or
17 \@pstool@never@true
18 \or
19 \fi}

cleanup 20 \DeclareOptionX{cleanup}{\def\pstool@rm@files{#1}}
\pstool@rm@files 21 \ExecuteOptionsX{cleanup={%
    .tex,.dvi,.ps,.pdf,.log,.aux,-blx.bib,.nav,.out,.snm,.toc}}

suffix 22 \DeclareOptionX{suffix}{\def\pstool@suffix{#1}}
\pstool@suffix 23 \ExecuteOptionsX{suffix={-process}}

24 \ifshellescape\else
25 \ExecuteOptionsX{process=none}
26 \PackageWarning{pstool}{^^J\space\space%
27 Package_option_[process=none]_activated^^J\space\space
28 because_-shell-escape_is_not_enabled.^^J%
29 This_warning_occurred}
30 \fi

31 \ProcessOptionsX
```

These are cute:

```

\OnlyIfFileExists 32 \providecommand\OnlyIfFileExists[2]{\IfFileExists{#1}{#2}{}}
\NotIfFileExists 33 \providecommand\NotIfFileExists[2]{\IfFileExists{#1}{#2}{}}

```

Command line abstractions between platforms:

```

34 \edef\pstool@cmdsep{\ifwindows\string&\else\string;\fi\space}
35 \edef\pstool@rm@cmd{\ifwindows\del\else\rm--\fi}

```

Delete a file if it exists:

```

\pstool@rm 36 \newcommand\pstool@rm[1]{%
37   \OnlyIfFileExists{\ip@directpath#1}{%
38     \immediate\write18{%
39       cd"\ip@directpath"\pstool@cmdsep\pstool@rm@cmd"#1"}}%
40   }

```

Generic function to execute a command on the shell and pass its exit status back into L^AT_EX. 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.

```

\pstool@exe 41 \def\pstool@exe#1#2{%
42   \pstool@writestatus{#1}{#2}%
43   \pstool@retrievestatus{#1}{\@tempa}%
44   \ifnum\@tempa>\z@
45     \PackageWarning{pstool}{%
      Execution failed during process: ^^J\#2^^J}%
46   \expandafter\pstool@abort
47   \fi}

```

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

```

\pstool@error 48 \def\pstool@error#1{\fbox{\color{red}\ttfamily\scshape
49   An error occured processing this graphic.}}

\pstool@abort 50 \def\pstool@abort#1\pstool@endprocess{\pstool@error}
51 \let\pstool@endprocess\@firstofone

```

It is necessary while executing commands on the shell to write the exit status to a temporary file to test for failures in processing. (If all versions of pdf_lat_ex supported input pipes, things might be different.)

```

\pstool@writestatus 52 \def\pstool@writestatus#1#2{%
53   \immediate\write18{%
54     cd"#1"\pstool@cmdsep
55     #2\pstool@cmdsep
56     \ifwindows
57       call_echo
58       \string^\@percentchar_ERRORLEVEL\string^\@percentchar
59     \else
60       echo\detokenize{${}}
61     \fi
62     >\pstool@statusfile}%

```

That's the execution; now we need to flush the write buffer to the status file. 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.

```

63   \ifwindows\else
64     \immediate\write18{%
65       touch"#1\pstool@statusfile}%
66   \fi}
\pstool@statusfile 67 \def\pstool@statusfile{status-deleteme.txt}

```

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

#1 is the path

#2 is the command to record the status within.

```

\pstool@retrievestatus 68 \def\pstool@retrievestatus#1#2{%
69   \CatchFileEdef{#2}{#1\pstool@statusfile}{}%
70   \pstool@rm{\pstool@statusfile}%
71   \ifx#2\pstool@statusfail
72     \PackageWarning{pstool}{%
73       Status_of_process_unable_to_be_determined:^^J#1^^J%
74       Trying_to_proceed...}%
75   \def#2{0}%
76   \fi}
\pstool@statusfail 77 \def\pstool@statusfail{\par}

```

4.1 File age detection

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

```

\pstool@datefiles 78 \def\pstool@datefiles{%

```

```

79 \edef\pstool@filenames{\ip@lastelement.eps\space\%
    \ip@lastelement.pdf\space}%
80 \immediate\write18{%
81   cd\ip@directpath"\pstool@cmdsep
82   \ifwindows
83     dir/T:W/B/O-D"\ip@lastelement.eps"\%
        \ip@lastelement.pdf">\pstool@statusfile
84   \else
85     ls-t"\ip@lastelement.eps"\ip@lastelement.pdf">\%
        \pstool@statusfile
86   \fi
87 }%
88 \pstool@retrievestatus{\ip@directpath}{\@tempa}%
89 \ifx\@tempa\pstool@filenames
90   \@tempwattrue
91 \else
92   \@tempwafalse
93 \fi
94 }

```

5 Command parsing

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

```

\pstool@alwaysprocess 95 \newcommand\pstool@alwaysprocess[3] [] {%
96   \inversepath*{#2}% calculate filename, path & inverse path
97   \pstool@process[#1]{#2}{#3}}

```

```

\pstool@neverprocess 98 \newcommand\pstool@neverprocess[3] [] {%
99   \includegraphics[#1]{#2}}

```

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

```

\pstool@maybeprocess 100 \newcommand\pstool@maybeprocess[3] [] {%
101   \inversepath*{#2}% calculate filename, path & inverse path
102   \ifx\ip@directpath\@empty
\ip@directpath 103     \def\ip@directpath{./}%
104   \fi
105   \IfFileExists{#2.pdf}{%
106     \pstool@datefiles

```



```

107     \if@tempswa\expandafter\@firstoftwo
108     \else\expandafter\@secondoftwo\fi{%
109         \pstool@process[#1]{#2}{#3}%
110     }{%
111         \includegraphics[#1]{#2}}%
112 }{%
113     \pstool@process[#1]{#2}{#3}%
114 }}

```

5.1 User commands

Finally, define `\pstool` as appropriate for the mode:

```

115 \ifpdf
116     \if@pstool@always@
117         \let\pstool\pstool@alwaysprocess
\pstool 118     \WithSuffix\def\pstool!\{\pstool@alwaysprocess}
\pstool* 119     \WithSuffix\def\pstool*\{\pstool@alwaysprocess}
120     \else\if@pstool@never@
121         \let\pstool\pstool@neverprocess
\pstool 122     \WithSuffix\def\pstool!\{\pstool@neverprocess}
\pstool* 123     \WithSuffix\def\pstool*\{\pstool@neverprocess}
124     \else
125         \let\pstool\pstool@maybeprocess
\pstool 126     \WithSuffix\def\pstool!\{\pstool@neverprocess}
\pstool* 127     \WithSuffix\def\pstool*\{\pstool@alwaysprocess}
128     \fi\fi
129 \else
130     \let\pstool\pstool@neverprocess
\pstool 131     \WithSuffix\def\pstool!\{\pstool@neverprocess}
\pstool* 132     \WithSuffix\def\pstool*\{\pstool@neverprocess}
133 \fi

```

6 The figure processing

`\ip@lastelement` is the filename of the figure stripped of its path (if any)

```

\pstool@jobname 134 \def\pstool@jobname{\ip@lastelement\pstool@suffix}

\pstool@process 135 \newcommand{\pstool@process}[3][\%
136     \pstool@write@processfile{#1}{#2}{#3}%
137     \pstool@exe}{\latex

```

```

138     -shell-escape
139     -output-format=dvi
140     -output-directory="\ip@directpath"

-interaction=batchmode

141     "\pstool@jobname.tex"}%
142     \pstool@exe{\ip@directpath}{%
143         dvips "\pstool@jobname.dvi"}%
144     \if@pstool@pdfcrop@
145         \pstool@exe{\ip@directpath}{%
146             ps2pdf "\pstool@jobname.ps" "\pstool@jobname.pdf"%
147             \pstool@exe{\ip@directpath}{%
148                 pdfcrop "\pstool@jobname.pdf" "\ip@lastelement.pdf"%
149             \else
150                 \pstool@exe{\ip@directpath}{%
151                     ps2pdf "\pstool@jobname.ps" "\ip@lastelement.pdf"%
152                 \fi
153             \pstool@endprocess{%
154                 \pstool@cleanup
155                 \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 156 \def\pstool@write@processfile#1#2#3{%
157     \immediate\openout\pstool@out_\#2\pstool@suffix.tex\relax
158     \immediate\write\pstool@out{%
159         \noexpand\pdfoutput=0% force DVI mode if not already

```

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

```

160     \if@pstool@nopreamble@
161         \unexpanded{%
162             \documentclass{minimal}
163             \usepackage{graphicx}}
164     \else
165         \unexpanded{%
166             \let\origdocument\document
167         \document{\endgroup\endinput}}%
\document

```

```

168         \noexpand
169         \input{\ip@inversepath\jobname}%
170     \fi

```

Now the preamble of the process file: (restoring document's original meaning; empty \pagestyle removes the page number)

```

171     \if@pstool@pdfcrop@ \else
172         \noexpand\usepackage[active,tightpage]{preview}
173     \fi
174     \if@pstool@nopreamble@ \else
175         \unexpanded{%
176             \let\document\origdocument
177             \pagestyle{empty}}%
178     \fi

```

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

```

179     \unexpanded{%
180         \begin{document}
181         \centering\null\vfill}%
182     \if@pstool@pdfcrop@ \else
183         \noexpand\begin{preview}%
184     \fi
185     \unexpanded{#3}% this is the "psfrag" material
186     \if@pstool@nofig@ \else
187         \noexpand\includegraphics[#1]{\ip@lastelement}%
188     \fi
189     \if@pstool@pdfcrop@ \else
190         \noexpand\end{preview}%
191     \fi
192     \unexpanded{%
193         \vfill\end{document}}%
194     }%
195     \immediate\closeout\pstool@out}

```

```

\pstool@cleanup 196 \def\pstool@cleanup{%
197     \@for\@ii:=\pstool@rm@files\do{%
198         \pstool@rm{\pstool@jobname\@ii}%
199     }}

```

7 User commands

These all support the suffixes * and !, so each user command is defined as a wrapper to \pstool.

for plain EPS figures (no psfrag):

```
\epsfig 200 \newcommand\epsfig[2] [] {\pstool@epsfig{\pstool}[#1]{#2}}
\epsfig* 201 \WithSuffix\newcommand\epsfig*[2] [] {\pstool@epsfig{%
    \pstool*}[#1]{#2}}
\epsfig 202 \WithSuffix\newcommand\epsfig![2] [] {\pstool@epsfig{%
    \pstool!}[#1]{#2}}

\pstool@epsfig 203 \def\pstool@epsfig#1[#2]#3{%
    204     \beginpgroup
    205     \@pstool@nopreamble@true
    206     #1[#2]{#3}{}%
    207     \endpgroup
    208 }
```

for EPS figures with psfrag:

```
\psfragfig 209 \newcommand\psfragfig[2] [] {\pstool@psfragfig{\pstool}[#1]{#2}}
\psfragfig* 210 \WithSuffix\newcommand\psfragfig*[2] [] {\pstool@psfragfig{%
    \pstool*}[#1]{#2}}
\psfragfig 211 \WithSuffix\newcommand\psfragfig![2] [] {\pstool@psfragfig{%
    \pstool!}[#1]{#2}}

\pstool@psfragfig 212 \def\pstool@psfragfig#1[#2]#3{%
    213     \ifnextchar\bgroup{%
    214         \pstool@psfragfig{#1}[#2]{#3}%
    215     }{%
    216         \pstool@psfragfig{#1}[#2]{#3}{}%
    217     }%
    218 }

\pstool@@psfragfig 219 \def\pstool@@psfragfig#1[#2]#3#4{%
    220     #1[#2]{#3}{%
    221         \InputIfFileExists{#3-psfrag}{}{}%
    222     #4}%
    223 }
```

for Matlab's laprint:

```

\laprintfig 224 \newcommand\laprintfig[2] [] {\pstool@laprintfig{\pstool}[#1]{%
#2}}
\laprintfig* 225 \WithSuffix\newcommand\laprintfig*[2] [] {\pstool@laprintfig{%
\pstool*}[#1]{#2}}
\laprintfig 226 \WithSuffix\newcommand\laprintfig![2] [] {\pstool@laprintfig{%
\pstool!}[#1]{#2}}

\pstool@laprintfig 227 \def\pstool@laprintfig#1[#2]#3{%
228 \begingroup
229 \@pstool@nofig@true
\resizebox 230 \renewcommand\resizebox[3]{##3}%
\includegraphics 231 \renewcommand\includegraphics[2] [] {#1[#2]{##2}{}}%
232 \input{#3}%
233 \endgroup
234 }

```

for Mathematica's MathPSfrag:

```

\mathpsfragfig 235 \newcommand\mathpsfragfig[2] [] {\pstool@mathpsfragfig{%
\pstool}[#1]{#2}}
\mathpsfragfig* 236 \WithSuffix\newcommand\mathpsfragfig*[2] [] {%
\pstool@mathpsfragfig{\pstool*}[#1]{#2}}
\mathpsfragfig 237 \WithSuffix\newcommand\mathpsfragfig![2] [] {%
\pstool@mathpsfragfig{\pstool!}[#1]{#2}}

\pstool@mathpsfragfig 238 \def\pstool@mathpsfragfig#1[#2]#3{%
239 #1[#2]{#3-psfrag}{\input{#3-psfrag}}%
240 }

\langle eof \rangle

```