The newpax package, v0.1

Reinserting annotations from included pdf file

Ulrike Fischer*

2020-03-30

1 Introduction

When including PDF-files in a document – may it be with `\includegraphics` or with `\includepdf` – clickable links and other annotations of these documents are lost.

The `newpax` package offers some tools to reinsert these annotations. It is based in large parts on the `pax` package from Heiko Oberdiek. It consists of a lua script to extract annotations and a style to reinsert them in the target document.

2 Quick use instructions

2.1 Step 1: extract the annotations

The lua script offers two functions which take as argument the name of a PDF without the extension. The functions can be used in some lua scripts but also in some document which then must be compiled with lualatex. You should create a `.pax` files if you want to use `pax.sty`, and the `.newpax` files if you want to use `newpax.sty` to reinsert the annotations.

Listing 1: doc-extract.tex

```
\documentclass{article}
% load the lua code
\directlua{require("newpax")}
% write .newpax files for newpax.sty
\directlua
  {
    newpax.writenewpax("pax-input")
    newpax.writenewpax("figure/pax-input2")
    newpax.writenewpax("example-image-a")
  }
```

*fischer@troubleshooting-tex.de

```
% and/or write .pax files for pax.sty
\directlua
  {
    newpax.writepax("pax-input")
    newpax.writepax("figure/pax-input2")
    newpax.writepax("example-image-a")
  }
\begin{document}
\end{document}
```

2.2 Step 2: Using the `.pax`-file with `pax.sty`

Ensure that the `.pax` file created in step 1 can be found by your main document. You can then insert your PDF files together with their annotations like in the following listing.

- This works with pdflatex and lualatex. lualatex needs the extra code demonstrated in the document.
- It needs two or three compilations until every reference is correct.
- There is a small typo in `pax.sty` which affects clipping, the patch shown in the listing correct this.
- Don't include PDFs with destinations twice as this will lead to duplicate destinations and pdflatex will complain.
- If annotations should not be reinserted remove the `.pax`-file.
- If `hyperref` is loaded you can change the color and style of link borders with hyperref options.

Listing 2: doc-pax-test.tex

```
\documentclass{article}
\usepackage{ifluatex,etoolbox}
\usepackage{pdfpages}
%pax needs this to run with lualatex
\ifluatex
\usepackage{pdftexcmds}
\makeatletter
\let\pdfstrcmp\pdf@strcmp
\let\pdfescapename\pdf@escapename
\makeatother
\usepackage{luatex85}
\fi
%load pax
\usepackage{pax}
%correct a bug in pax affecting clipping
\makeatletter
```

```
\patchcmd\PAX@pdf@annot{\PAX@pagellx}{\PAX@page@llx}{}{\fail}
\makeatother
\begin{document}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=2]{pax-input}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=1]{pax-input}

\includepdf[pages=-]{figure/pax-input2}
\end{document}
```

2.3 Experimental!: Variant of Step 2: Using the `.newpax`-file with `newpax.sty`

The style `newpax` is based on `pax.sty` but extends it in various way. It is an experimental package which requires the experimental pdfresources code.

1. clone https://github.com/latex3/pdfresources
2. In the main folder run `l3build install .` This will install the package in your texmfhome.

Attention! Experimental means EXPERIMENTAL! The code is currently in full flow. And it is *not* compatible with every other package.

The following listing shows how to use `newpax.sty`.

- It should work with pdflatex, lualatex and xelatex. The latex/dvips route fails as this can't include PDF anyway.
- Some provision have been added to allow multiple inclusion of the same PDF, but if you insert different sets of pages from a PDF destinations can still be missing. So better avoid it.
- You can choose for every file if border color and styles of links are taken from the source PDF or from the hyperref settings. But you can't adjust or change colored links.

Listing 3: doc-newpax-test.tex

```
%improves compability with tikz etc ...
\PassOptionsToPackage{patches}{pdfresources}

\documentclass{article}
\usepackage{pdfpages,xcolor}

% use a custom hyperref driver:
\usepackage[customdriver=hgeneric-experimental]{hyperref}
\hypersetup{linkbordercolor=blue}
\hypupdateattribute %after \hypersetup

\usepackage{newpax}

%use the link border color and style of the imported pdf
```

```
%and not hyperref colors
\newpaxsetup{usefileattributes=true}

\begin{document}

\includegraphics[scale=0.5,trim=4cm 15cm 8cm 3cm,clip,page=1]{pax-input}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=2]{pax-input}

%set a unique suffix is the pdf is imported twice
\newpaxsetup{destsuffix=B}
\includegraphics[scale=0.5,trim=4cm 15cm 8cm 3cm,clip,page=1]{pax-input}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=2]{pax-input}

% suppress the adding of annotations
\newpaxsetup{addannots=false}
\includegraphics[scale=0.5,trim=4cm 15cm 8cm 3cm,clip,page=1]{pax-input}

%reactivate
\newpaxsetup{addannots=true}
\includepdf[pages=-]{figure/pax-input2}
\end{document}
```

2.4 Combining the steps

When using lualatex the code that extracts the annotations and writes the `.pax`/`.newpax`-files can be in the same document that uses the files later.

In the future it would be nice if the luacode could like e.g. epstopdf be called on the fly by a document.

3 Background

Clickable links in a PDF are one example of an annotation. Annotations are areas on a page which are associated with an action. A typical annotation object could look like this in the PDF:

```
15 0 obj
<<
/Type /Annot
/Subtype/Link
/Rect [147.716 654.025 301.887 665.15]
/Border[0 0 1]/BS<</S/U/W 1>>/H/I/C[0 1 1]
/A<</Type/Action/S/URI/URI(https://www.latex-project.org)>>
>>
endobj
```

This is an object of type `Annot` and subtype `Link`. The `/Rect` value describes the rectangle of this annotation. The coordinates are absolute coordinates related to the current page. It is important to understand that an annotation is not connected to some page content but only to a location! The `/Border` setting and the other values in this line describe the look and color of annotation. The `/A` value contains the action, in this case it is an url to an external website.

To "reactivate" the annotations of an included pdf one has to do a number of tasks.

- One must *retrieve and store* the annotations of the included pdf. For links to external url this requires to find only one object like the one shown above. But e.g. internal links point to a destination object and these must be found too.
- One must *recalculate* the rectangle coordinates to fit to the coordinate system of the target page: as the included pdf can be placed at various positions, scaled, rotated and even clipped this is not an easy task. Destinations have rectangles too that must be recalculated.
- One must *reinsert* the annotation and related objects. This has to take into account the a pdf is perhaps not included completly, a link shouldn't point to a missing page or a clipped annotation. It also has to take into account that a pdf is perhaps inserted more than once or in steps.

3.1 Retrieving and storing annotations

Theoretically one can do it manually: Uncompress the PDF (or when using LaTeX, create directly an uncompressed one), open it in an editor and copy and paste all needed objects. Practically one naturally want some tool.

The `pax` package from Heiko Oberdiek consists of a perl script and a java-jar file `PDFAnnotExtractor` which can extract the necessary objects. It writes the information to a file with the extension `pax`. When it has been successfully install it works quite fine. Problems with this approach are

- `PDFAnnotExtractor` requires an external, old version of the java library of PDFbox which must be installed manually.
- It requires a java installation and
- it is not extensible.

The `newpax` package comes with a lua-file. It uses the `pdfe` library embedded in luatex to extract the annotations and other needed information. `newpax` writes the information to a file with the extension `pax` or `newpax`. The content of the files is (nearly) identical to the content of the `pax`-file written by `PDFAnnotExtractor`. The lua code was written by looking at example outputs from `PDFAnnotExtractor` and reproducing it in lua. The ordering of element is a bit different and some strings are output in a different way but for the examples I used the resulting `pax`-files can be used together with the original `pax.sty`. But due to the fact that the code was written without real spec simply by looking at examples, it is quite probably that the lua code is not yet handling all objects or options that `PDFAnnotExtractor` outputs. But the code can rather easily be extended when the needs arises.

The code also doesn't handle structure elements, neither at the export nor at the import. I have yet no real idea what would be sensible here (and I'm quite sure that `PDFAnnotExtractor` doesn't handle this either.)

4 Importing annotations

The `pax` package from Heiko Oberdiek does the hard work to recalculate the annotation rectangles and to decide which annotation and which destination should be reinserted. It also patches the `\includegraphics` command to automate this.

`newpax` reuses the core commands of `pax`. It adds a number of switches and changed primitive so that more engines and backend can be supported.

5 Example