The xsavebox Package, v0.1

Alexander Grahn
a.grahn@web.de

25th February 2016

Abstract

This package defines commands for saving content that can be repeatedly placed into the document without replicating DVI/PDF code in the output file, allowing for smaller size of the final PDF file and improved content caching for faster display in certain PDF viewers. The user commands are modelled after the standard LaTeX commands `\savebox`, `\sbox`, `\usebox` and the '`lrbox`' environment. The package supports all common TeX engines and back-ends, including '`dvips`'.

1 Introduction

Whenever the standard LaTeX command `\usebox{`*save-box*`}` is issued to insert a previously defined *save-box* more than once, the typeset content stored therein is written as DVI or PDF code into the output file again. The redundant code adds to the overall file size and may impair the page caching facilities built into some PDF viewers.

The PDF file format defines a powerful mechanism for packing readily typeset content once into self-contained entities, so-called 'XObjects', that can be referenced at other places within the PDF document.

The '`xsavebox`' package makes this PDF feature accessible on the LaTeX level as a set of user commands which look similar to and are used in a similar way as the well-known *save-box* related LaTeX commands.

All common TeX engines and back-ends are supported, which are:

- pdfLaTeX, LuaLaTeX,
- LaTeX → `dvips` → `ps2pdf`/Distiller
- (XH)LaTeX → `(x)dvipdfmx`

To enable '`dvipdfmx`', pass it as a document class option.

It should be emphasized that 'XObjects' is a PDF feature. Content saved and referenced using 'XObjects' is only visible in the final PDF output, but not in intermediate formats of the work-flow if those are involved, namely DVI and

PostScript. Of course, PostScript converted back from PDF displays the content correctly.

2 User commands

Content saving

```
\xsbox{<xsbox name>}{<content>}
\xsavebox{<xsbox name>}[<width>][<position>]{<content>}
\xsavebox*{<xsbox name>}[<width>][<position>]{<content>}

\begin{xlrbox}{<xsbox name>}
  <content>
\end{xlrbox}

\begin{xlrbox*}{<xsbox name>}
  <content>
\end{xlrbox*}
```

The main difference of these commands as compared to their standard LaTeX counterparts without the leading '`x`' is the way of naming boxes. The label `<xsbox name>` is an identifier that may be composed of arbitrary non-active characters, including spaces and numbers. A command for declaring a box register `<xsbox name>` does not exist.

The `[<width>]` and `[<position>]` options have the same meaning as with `\savebox` and `\makebox`. As usual, the additional length commands

```
\width
\height
\depth
\totalheight
```

are defined for use in the `[<width>]` option and refer to the original dimensions of `<content>`. The value of `<position>` may assume one of '`l`', '`r`', '`c`' or '`s`'. The default is '`c`' for text centred in the box.

`<content>` is typeset in LR-mode. Longer text to be typeset in paragraph mode must be put into a `\parbox` or '`minipage`'.

The starred ('`*`') versions of the commands allow for later colour injection into the boxes at the place of their referencing. The colour which is active at the time of building the box is not saved with the content. This feature only works with pdfLaTeX and LuaLaTeX.

Note that

```
\xsbox{image for frequent use}{\includegraphics{example}}
```

is useful only in the LaTeX → `dvips` → `ps2pdf` work-flow, as all other engines and back-ends already take care of preventing multiple graphics file inclusion.

Verbatim content can only be saved using the '`xlrbox[*]`' environment.

With LaTeX in DVI mode and XeLaTeX, box *saving* commands should not be placed on a line of their own with empty lines above and below. For technical reasons this will produce an empty paragraph. Always place them at the beginning or at the end of a paragraph in the input file. Also, box saving commands cannot be placed in the document preamble with LaTeX (DVI) and XeLaTeX.

Referencing saved content

Previously saved content can be inserted with

`\xusebox{<xsbox name>}`

or

`\the<xsbox name>`

The second, shorthand form can be used if `<xsbox name>` is composed exclusively of letters ('`a`'–'`z`', '`A`'–'`Z`'). For example, a box named '`MyFirstExample`' could be referenced as

```
\theMyFirstExample
```

but a box named '`My 1st Example ;-)`' would require

```
\xusebox{My 1st Example ;-)}
```

The referencing commands `\xusebox{<xsbox name>}` and `\the<xsbox name>` can again be placed inside the `<content>` body of box saving commands. There is no upper limit of nesting levels.

`\xusebox{<xsbox name>}` and `\the<xsbox name>` behave exacly like common TeX boxes. Therefore, they can be scaled, rotated and resized using the corresponding commands from the '`graphicx`' package.

3 Example

An example with colour injection (pdfLaTeX/LuaLaTeX-only) follows:

Here is a silly boxed paragraph that no one will ever use for anything.

The same silly boxed paragraph that no one will ever use was inserted again but with a different colour and rotated by 90 degree.

```
\usepackage{xsavebox}
\usepackage{color}
\usepackage{graphicx}
...
\begin{xlrbox*}{SavedPar}% `*' --> no colour at the time of saving
  \begin{minipage}[b]{1in}
  silly boxed paragraph that no one will ever use
  \end{minipage}
\end{xlrbox*}
%colours injected into \theSavedPar
Here is a \fbox{\color{blue}\theSavedPar} for anything.

The same \fbox{\color{green}\rotatebox{90}{\theSavedPar}} was
inserted again but with a different colour and rotated by 90 degree.
```