The trimclip Package

Part of the adjustbox bundle

Martin Scharrer
martin@scharrer-online.de

CTAN: http://www.ctan.org/pkg/adjustbox

Version v1.0 – 2012/05/16

Abstract

This package extends the standard `graphicx` package by providing the missing `\trimbox` and `\clipbox` macros to trim and clip arbitrary TEX material. The macros allow for verbatim content. Equivalent environments are also provided. The package comes with own clipping drivers for all common output formats as well as a `pgf` fall-back driver.

1 Introduction

The standard LATEX package `graphicx`.allows to scale, resize and rotate either images or text (i.e. any TEX content). For text the macros `\scalebox`, `\resizebox` and `\rotatebox` can be used, while equivalent keys exist for the `\includegraphics` macro. However, while it is possible to trim and clip images using the `trim`, `viewport` and `clip` keys, no equivalent macros are provided. This package closes this gap by defining the macros `\trimbox` and `\clipbox`. As an extra the macro `\marginbox` is also provided. It can be seen as an inverted `\trimbox`, expanding the official size of the content instead of reducing it. Originally these macros were included in the `adjustbox` package together with the general `\adjustbox` macro. However, the fundamental clip and trim macros and their driver files are now packed into this minimalistic package, so that other packages can reuse its functionality without the need to load the ever-growing `adjustbox` package.

The macros provided by this package differ in three aspects from the macros defined by `graphicx`. The content argument is actually read directly as a horizontal box and not as a macro argument, even when the syntax looks the same. This allows for arbitrary content including special things like verbatim material. It is allowed to replace the '`{ }`' around the content with `\bgroup` and `\egroup`. Furthermore, for every macro there is an equivalent environment with the same name. Special care is taken to allow the same name for both, which is normally not allowed. Finally, the lengths arguments of the macros can contain algebraic expressions to calculate the used length. This is only possible with the `graphicx` macros if the `calc` package is loaded. However, the `trimclip` macros use the `adjcalc` wrapper package which either uses ϵ-TEX primitives, `calc` or `pgfmath` to provide this feature.

2 Dependencies

This package uses the author's other packages `collectbox` (to collect the content as a real box) and `adjcalc` (to allow for math expressions for lengths). The latter is part of the same `adjustbox` bundle and should have be installed together with `trimclip`.

3 Drivers

The clip operation can not be implemented using general TEX commands, but is rather output format specific. The clipped material is actually included unclipped and the output file (i.e. PDF or PS file) contains format specific instructions, so that the document viewer will clip the content when the document is displayed. Depending on the used compilation work-flow (like `pdflatex`, `latex`+`dvips` or `latex`+`dvipdfm`, etc.) this clipping instructions must be passed in a different way. In order to support all of these, dedicated driver files are provided which hold the specific low-level instructions. This requirement should also be known to most users from the `graphics/x`, `(x)color` or `hyperref` packages which also require output format specific low-level instructions to implement their features.

A set of driver files for the most common used LATEX compilers is provided with this package (see section 4 for a list). If no suitable driver file is found, the `pgf` package is used instead to implement the clip operation. This (large) package comes with its own set of driver files and should cover any other LATEX compilers. The `trimclip` drivers were inspired by the `graphic/x` and `pgf` driver code and were written by Joseph Wright of the LATEX3 project and Martin Scharrer (the author of this package).

4 Package Options

Normally the package should be loaded without any options. A suitable driver will then automatically be selected. However, the package accepts the following options to select the used driver manually. Any other option is passed to the `graphicx` package and the driver selected by it is used. However, this does not work if `graphicx` or `graphics` was already loaded before. In this case any unknown option is taken as driver and a file '`tc-`⟨*option*⟩`.def` ' is loaded if it exits. If not, the default PGF fall-back driver is used. PGF comes with a own set of drivers but is large and can be considered a significant overhead if used only for rectangular clipping.

pdftex Use the `pdftex` driver. This driver is automatically selected for `pdflatex` and `lualatex` and should not be used for any other LATEX compilers.

dvips Use the `dvips` driver. This driver is automatically selected for `latex`.

xetex Use the `xetex` driver. This driver is automatically selected for `xelatex`.

dvipdfm Use the `xetex` driver which is also compatible with `dvipdfm`.

dvipdfmx Use the `xetex` driver which is also compatible with `dvipdfmx`.

pgf Use the fall-back PGF driver explicitly. This makes sense if issue with another driver are encountered.

It should be noted that choosing an incorrect driver will lead to clip operation not being applied (they act like trim operations) and may lead to a broken output file.

5 Argument Values

All macros of this package and their matching environments require four length values which are used to change the left, bottom, right and top side of the content. Because of the used `adjcalc` package complicated algebraic expressions can be used to calculate these amounts. The used math engine can be changed by loading `adjcalc` with the appropriate option before loading `trimclip`. Please see the `adjcalc` manual for more details on this. Like with the `trim` or `viewport` keys of `\includegraphics` the length values must be separated by spaces. Note that if a previous length expression ends in a macro any trailing spaces will be removed by TEX. Therefore it is required to wrap this *complete* length expression in braces. Several examples of this are shown in the *Usage* section. It is also possible to only provide a single length which is used for all four sides or only two lengths which are taken for the left/right as well as bottom/top side. This simplifies symmetric operations and got inspired by Cascading Style Sheets (CSS) used to style websites.

If a length value is a simple number without a unit, a default unit is substituted (usually '`bp`', *big points*, the standard PostScript and PDF unit). This default unit can be changed using `\adjcalc{defaultunit=`⟨*unit*⟩`}` or completely disabled (`defaultunit=none`). See the `adjcalc` manual for more details.

The length values can contain the following macros to refer to the original size of the content:

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

These LATEX lengths hold the original dimensions of the content and can be used to make relative changes. Like any other length registers they can be used with a factor, e.g. `.5\width` to refer to half the natural width of the content.

6 Usage

6.1 Trimming

```
\trimbox{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨content⟩}
\trimbox{⟨all sites⟩}{⟨content⟩}
\trimbox{⟨left/right⟩ ⟨top/bottom⟩}{⟨content⟩}
\trimbox*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨content⟩}
```

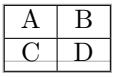
The macro `\trimbox` trims the given amount from the lower left (ll) and the upper right (ur) corner of the box. This means that the amount ⟨*llx*⟩ is trimmed from the left side, ⟨*lly*⟩ from the bottom and ⟨*urx*⟩ and ⟨*ury*⟩ from the right and top of the box, respectively. If only one value is given it will be used for all four

sites. If only two values are given the first one will be used for the left and right side (llx, urx) and the second for the bottom and top side (lly, ury).

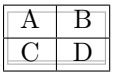
If the starred version is used the four coordinates are taken as the **viewport** instead, i.e. the box is trimmed to the rectangle described by the coordinates. In this case all four values must be specified explicitly.

Examples:

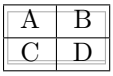
```
\examplecontent
```



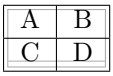

```
\trimbox{2pt 3pt 2pt 3pt}{\examplecontent}
```



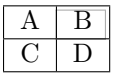

```
\trimbox{2pt 3pt}{\examplecontent}
```




```
\trimbox{2pt}{\examplecontent}
```

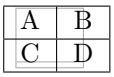



```
\trimbox{{.5\width} {.5\totalheight} 2pt 2pt}
    {\examplecontent}
```

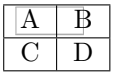



```
\trimbox*{5pt 0pt 3em 2em}{\examplecontent}
```

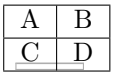

```
\trimbox*{5pt -2pt 3em 2em}{\examplecontent}
```




```
\trimbox*{5pt 10pt 3em 2em}{\examplecontent}
```




```
\trimbox*{5pt -3pt 3em -1pt}{\examplecontent}
```




```
\begin{trimbox}{⟨1, 2 or 4 trim values⟩}
  ⟨content⟩
\end{trimbox}
```

```
\begin{trimbox*}{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}
  ⟨content⟩
\end{trimbox*}
```

The `trimbox` and `trimbox*` environments do the same as the corresponding macros.

6.2 Clipping

```
\clipbox{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨content⟩}
\clipbox{⟨all sites⟩}{⟨content⟩}
\clipbox{⟨left/right⟩ ⟨top/bottom⟩}{⟨content⟩}
\clipbox*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨content⟩}
```

The **\clipbox** macro works like the **\trimbox** and trims the given amounts from the ⟨*text*⟩. However, in addition the trimmed material is also clipped, i.e. it is not shown in the final document. Note that the material will still be part of the output file but is simply not shown. The full content can still be exported using special tools, so using **\clipbox** (or **\includegraphics**`[clip,trim=...]`) to censor classified information would be a bad idea. The starred version will again use the given coordinates as viewport.

```
\begin{clipbox}{⟨1, 2 or 4 trim values⟩}
  ⟨content⟩
\end{clipbox}
```

```
\begin{clipbox*}{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}
  ⟨content⟩
\end{clipbox*}
```

The environment versions of **\clipbox** and **\clipbox***. The same rules as for the trimming environments apply.

Examples:

```
\examplecontent
```

A	B
C	D

```
\clipbox{2pt 3pt 2pt 3pt}{\examplecontent}
```

A	B
C	D

```
\clipbox{2pt 3pt}{\examplecontent}
```

A	B
C	D

```
\clipbox{2pt}{\examplecontent}
```

A	B
C	D

```
\clipbox{{.5\width} {.5\totalheight} 2pt 2pt}
   {\examplecontent}
```

B

```
\clipbox*{5pt 0pt 3em 2em}{\examplecontent}
```

A	
C	

```
\clipbox*{5pt -2pt 3em 2em}{\examplecontent}
```

A	
C	

```
\clipbox*{5pt 10pt 3em 2em}{\examplecontent}
```

A	

```
\clipbox*{5pt -3pt 3em -1pt}{\examplecontent}
```

6.3 Margin

```
\marginbox{⟨all sites⟩}{⟨content⟩}
\marginbox{⟨left/right⟩ ⟨top/bottom⟩}{⟨content⟩}
\marginbox{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨content⟩}
```

```
\begin{marginbox*}{⟨1, 2 or 4 margin values⟩}
  ⟨content⟩
\end{marginbox*}
```

This macro and environment can be used to add a margin (white space) around the content. It can be seen as the opposite of **\trimbox**. The original baseline of the content is preserved because ⟨*lly*⟩ is added to the depth.

Example:

```
Before \fbox{\marginbox{1ex 2ex 3ex 4ex}{Text}} After
```

Before Text After

```
\marginbox*{⟨all sites⟩}{⟨content⟩}
\marginbox*{⟨left/right⟩ ⟨top/bottom⟩}{⟨content⟩}
\marginbox*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨content⟩}
```

```
\begin{marginbox}{⟨1, 2 or 4 margin values⟩}
  ⟨content⟩
\end{marginbox}
```

This starred version is almost identical to the normal **\marginbox**, but also raises the content by the ⟨*lly*⟩ amount, so that the original depth is preserved instead of the original baseline. Note that while **\marginbox** is basically the opposite of **\trimbox**, **\marginbox*** is not the opposite of **\trimbox***.

Example:

```
Before \fbox{\marginbox*{1ex 2ex 3ex 4ex}{Text}} After
```

Before Text After

7 Diagrams

The box dimensions, trim values and change of the baseline for different scenarios are visualized by the following diagrams.

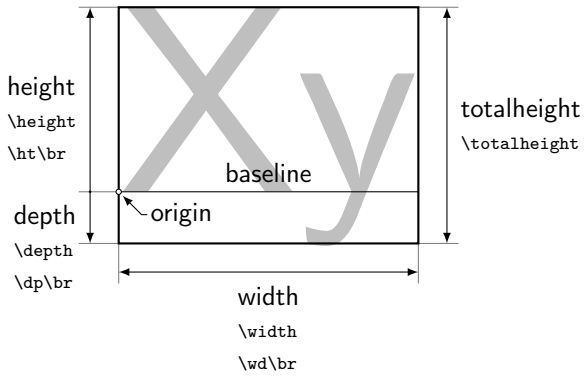


Figure 1: Box dimensions. Shown are also the LaTeX macros and the TeX primitives. Here `\br` stands for a box register. Note that the depth is a positive values on its own downwards pointed axes.

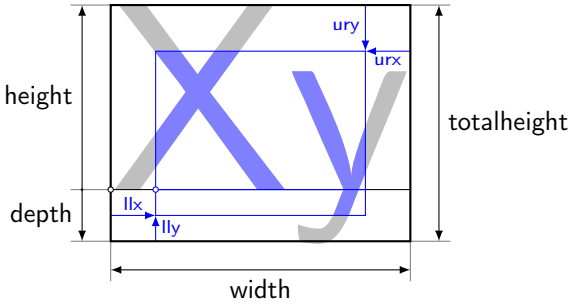


Figure 2: Trimming. The four values are removed from each side.

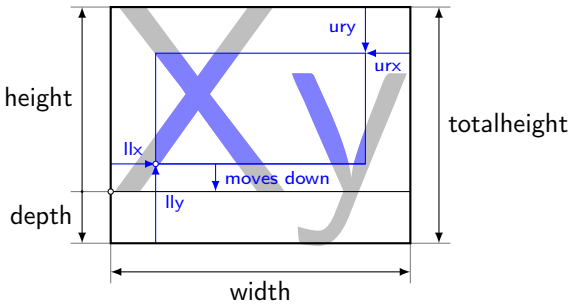


Figure 3: Trimming with `lly`>depth. In this case the resulting box moves up to the original baseline.

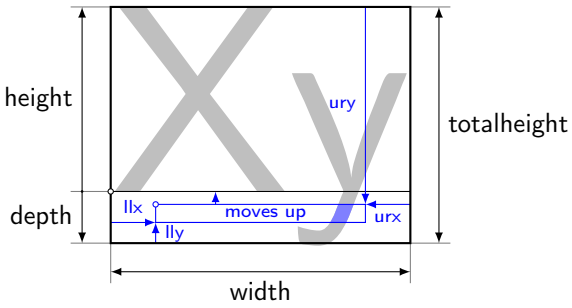


Figure 4: Trimming with `ury`>height. In this case the resulting box falls down to the original baseline

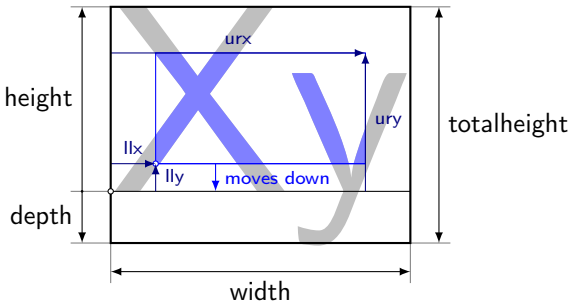


Figure 5: Viewport with `lly`>0pt. The `ll` and `ur` values are taken from the origin. The baseline is the vertical zero reference.

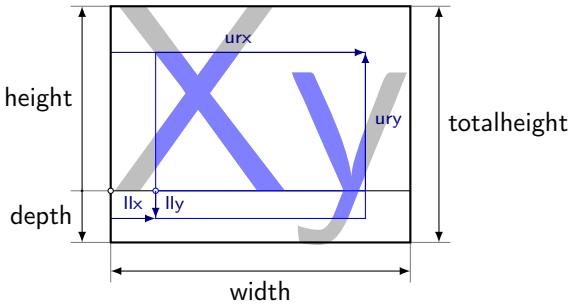


Figure 6: Viewport with `lly`<0pt. In this case the viewport ranges into the depth of the original box.

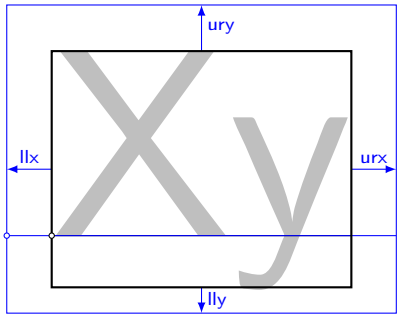


Figure 7: Marginbox. The `llx` and `urx` are added to the left and right and increase the width. The `ury` is added to the height and `lly` to the depth of the box. This keeps the baseline at the original position.

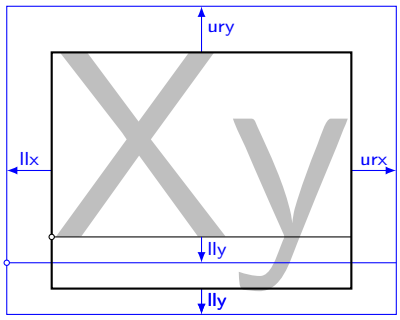


Figure 8: Marginbox*. In addition to the normal margin the content is also raised by `lly`, so that the original depth is preserved. This effectively moves the baseline down.