Writing, running and including the output of external documents from within a main LaTeX document –v. 0.19

Herbert Voß*

24. April 2022

Inhaltsverzeichnis

*herbert@dante.de

1 Syntax

```
\usepackage{hvextern}
```

This package allows to write external METAPOST, TEX, ConTEXt, LATEX, LuaTEX, LuaLATEX, XETEX, XELATEX, Lua, Perl and/or Python source code, which will then be run via `shell escape` to create a PDF oder text output to include it into the main LATEX document.

There is only one environment:

```
\begin{externalDocument}[⟨options⟩]{⟨external filename without extension⟩}
...
source code
...
\end{externalDocument}
```

The main document *must* be run with the `–shell–escape` option, otherwise it won't work, e.g.:

```
laulatex ––shell–escape ⟨file⟩
```

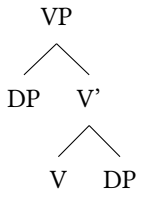
The purpose for this package is to show the output of documents which have to be compiled with a different preamble or a different engine or a complete different system, but integrating the output automatically in the main document..

2 First examples

2.1 Without showing the code

This document was run with LuaLATEX. Suppose you want to insert the output of a document which needs for several reasons a pdfLATEX run. Instead of created and running a document outside of the main document and then to insert the output we can do this from within this LuaLATEX document itself. The external document is compiled with pdfLATEX and the output is insert as pdf image. The source code itself is not shown by the environment `externalDocument`.

```
\begin{externalDocument}[
  compiler=pdflatex,force,cleanup]{Roemer1}
\documentclass{standalone}
%StartVisiblePreamble
\usepackage{fontenc}
\usepackage{libertinus}
\usepackage[linguistics]{forest}
\forestapplylibrarydefaults{linguistics,
    edges}
%StopVisiblePreamble
\begin{document}
\begin{forest}
[VP
  [DP]
  ['V
   [V]
  [DP]
  ]
]
\end{forest}
\end{document}
\end{externalDocument}
```



2.2 Showing code and output of a Python example

The png image is created on the fly with the following arguments of `externalDocument`:

```
\begin{externalDocument}[
  compiler=python3,
  showFilename,
  code,
  ext=py,
  docType=py,
  usefancyvrb,
  grfOptions={width=\linewidth}]{python}
... Python code ...
\end{externalDocument}
```

The code which is declared as header and main can be marked by:

```
\extern@ExampleType{py}
  {\NumChar StartVisibleMain}
  {\NumChar StopVisibleMain}
  {\NumChar StartVisiblePreamble}
  {\NumChar StopVisiblePreamble}
```

`\NumChar` is the default Python comment character # and needs to be saved with a different catagory, which is done internally by the package. The complete definition of the code is:

```
\begin{externalDocument}[
  compiler=python3,
  showFilename,
```

```
  code,
  ext=py,
  force,
  docType=py,
  usefancyvrb,
  grfOptions={width=\linewidth}]{python}
import os
#StartVisiblePreamble
from PIL import Image
import subprocess
# drawing area (xa < xb and ya < yb)
xa = -0.1716
xb = -0.1433
ya = 1.022
yb = 1.044
maxIt = 1024  # iterations
imgx = 1000   # image size
imgy = 750
image = Image.new("RGB", (imgx, imgy))
#StopVisiblePreamble

#StartVisibleMain
for y in range(imgy):
    cy = y * (yb - ya) / (imgy - 1)  + ya
    for x in range(imgx):
        cx = x * (xb - xa) / (imgx - 1) + xa
        c = complex(cx, cy)
        z = 0
        for i in range(maxIt):
            if abs(z) > 2.0: break
            z = z * z + c
        r = i % 4 * 6
        g = i % 8 * 32
        b = i % 16 * 16
        image.putpixel((x, y), b * 65536 + g * 256 + r)
#StopVisibleMain
# now get the filename created by the latex document
imageName = os.path.basename(os.path.splitext(__file__)[0])+".png" # get filename
image.save(imageName, "PNG")
\end{externalDocument}
```

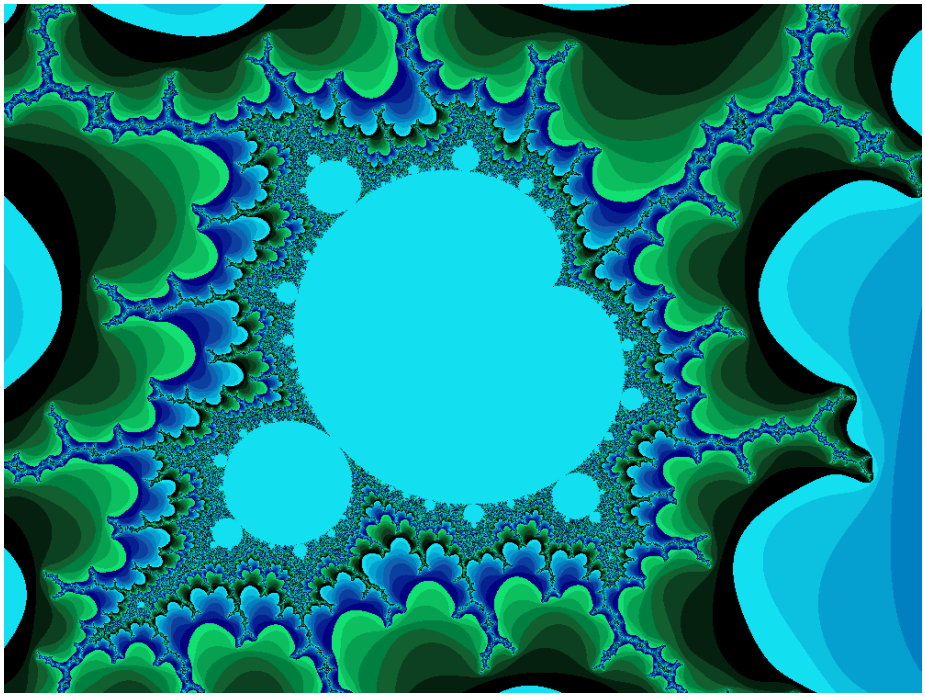
And with using this code we get the image as png inserted. The given filename of the external document is internally extended by a consecutive number which isn't known to the Python code. However, it is no problem in any programming language to get the name of a running file. The forlast line in the above code shows how it can be done with Python.

```
from PIL import Image
import subprocess
# drawing area (xa < xb and ya < yb)
xa = -0.1716
```

```
xb = -0.1433
ya = 1.022
yb = 1.044
maxIt = 1024  # iterations
imgx = 1000   # image size
imgy = 750
image = Image.new("RGB", (imgx, imgy))
```

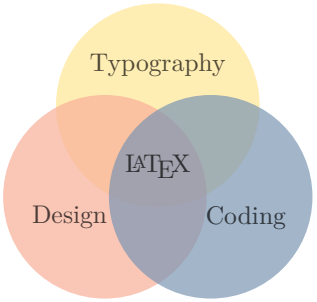
```
for y in range(imgy):
    cy = y * (yb - ya) / (imgy - 1)  + ya
    for x in range(imgx):
        cx = x * (xb - xa) / (imgx - 1) + xa
        c = complex(cx, cy)
        z = 0
        for i in range(maxIt):
            if abs(z) > 2.0: break
            z = z * z + c
        r = i % 4 * 6
        g = i % 8 * 32
        b = i % 16 * 16
        image.putpixel((x, y), b * 65536 + g * 256 + r)
```

python-2.py



```
\usepackage{tikz}
\usepackage[hks,pantone,xcolor]{xespotcolor}

\SetPageColorSpace{HKS}
\definecolor{HYellow}{spotcolor}{HKS05N,0.5}
\definecolor{HRed}{spotcolor}{HKS14N,0.5}
\definecolor{HBlue}{spotcolor}{HKS38N,0.5}
\begin{tikzpicture}[scale=0.7,fill opacity
    =0.7]
 \fill[HYellow]( 90:1.2) circle (2);
 \fill[HRed]   (210:1.2) circle (2);
 \fill[HBlue]  (330:1.2) circle (2);
 \node at ( 90:2)  {Typography};
 \node at ( 210:2) {Design};
 \node at ( 330:2) {Coding};
 \node {\LaTeX};
\end{tikzpicture}
```



3 Setting marker in the source

The marker for the code ranges which should be listed depend to the used programming language:

```
[...]
%StartVisiblePreamble
[... listed preamble code ]
%StopVisiblePreamble
[...]
\begin{document}
[...]
\end{document}
```

everything between `%StartVisiblePreamble` and `%StopVisiblePreamble` will be listed as preamble and in case of a LaTeX source everything between `\begin{document}` and `\end{document}` as body. The marker must be defined in an own config file, e.g. `hv-extern-py.cfg`:

```
\extern@ExampleType{py}
  {\NumChar StartVisibleMain}
  {\NumChar StopVisibleMain}
  {\NumChar StartVisiblePreamble}
  {\NumChar StopVisiblePreamble}
```

`\NumChar` is the comment character #, which needs a special handling. This version of `hvextern` supports the following programming languages (option `compiler`): mpost, tex, latex, luatex, python3, perl, lua, xetex, pdflatex, lualatex, xelatex, and context. The default is pdflatex. The option `docType` selects the config file, which must be one of context, lua, pl, tex, latex, mp, and py. For Lua it is

```
\extern@ExampleType{lua}
  {--StartVisibleMain}
  {--StopVisibleMain}
  {--StartVisiblePreamble}
  {--StopVisiblePreamble}
```

It defines the marker strings for the listed code sequences. In some cases you have to use multiple times the same value for different optional arguments, e.g.

```
ext=lua, compiler=lua, docType=lua, ...
```

4 Optional arguments

The deafault setting is always shown in brackets.

4.1 Programs and runs

The `progpath` should only in some rare cases needed. In general all used compilers will be found by the system. A given `progpath` must be end with a slash, e.g. `./bin/` The option `runsequence` is currently not used.

```
\define@key{hv}{progpath}[]{\def\hv@extern@progpath{#1}}
\define@key{hv}{compiler}[pdflatex]{\def\hv@extern@compiler{#1}}
\define@key{hv}{runsequence}[]{\def\hv@extern@runsequence{#1}}
\define@key{hv}{runs}[1]{\setcounter{hv@extern@runs}{#1}}
```

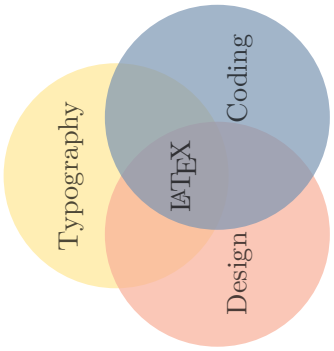
4.2 Grafik options

```
\define@key{hv}{grfOptions}[]{\def\hv@extern@grfOptions{#1}}
```

The option is passed to `\includegraphics`, e.g. `angle=90,width=\linewidth` for the follwing example.

```
\usepackage{tikz}
\usepackage[hks,pantone,xcolor]{xespotcolor}
```

```
\SetPageColorSpace{HKS}
\definecolor{HYellow}{spotcolor}{HKS05N,0.5}
\definecolor{HRed}{spotcolor}{HKS14N,0.5}
\definecolor{HBlue}{spotcolor}{HKS38N,0.5}
\begin{tikzpicture}[scale=0.7,fill opacity
    =0.7]
 \fill[HYellow]   ( 90:1.2) circle (2);
 \fill[HRed] (210:1.2) circle (2);
 \fill[HBlue]  (330:1.2) circle (2);
 \node at ( 90:2) {Typography};
 \node at ( 210:2) {Design};
 \node at ( 330:2) {Coding};
 \node {\LaTeX};
\end{tikzpicture}
```



4.3 Listings options

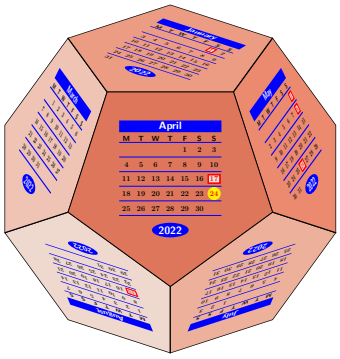
`\define@key{hv}{lstOptions}[]{\def\hv@extern@lstOptions{#1}}`

The option is passed either to `\lstinputlisting`, or, if `usefancyvrb` is active, to `\VerbatimInput`. The following example uses

```
lstOptions={basicstyle=\sffamily\itshape\scriptsize},
```

```
\usepackage{pst-calendar}
```

```
\psscalebox{0.3}{%
  \psCalDodecaeder[Year=2022,style=april]%
}
```



4.4 Background color

There are different colors for the preamble and body listing: the backgrund and frame color.

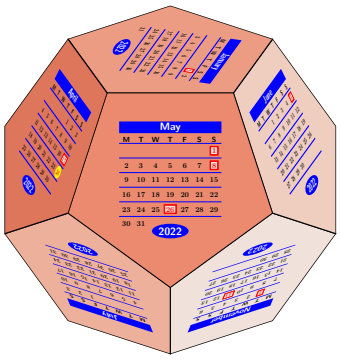
```
\define@key{hv}{BGpreamble}[black12]{\def\hv@extern@BGpreamble{#1}}
\define@key{hv}{BGbody}[black8]{\def\hv@extern@BGbody{#1}}
\define@key{hv}{BOpreamble}[black12]{\def\hv@extern@BOpreamble{#1}}
\define@key{hv}{BObody}[black8]{\def\hv@extern@BObody{#1}}
```

The options are passed to `tcolorbox` and preset to `black12` and `black!8`. The color of the frame is set to the same values, hence not seen. The following example uses

```
BGpreamble=red!10, BOpreamble=red,
BGbody=blue!8, BObody=blue,
```

```
\usepackage{pst-calendar}
```

```
\psscalebox{0.3}{%
  \psCalDodecaeder[Year=2022,style=may]%
}
```



4.5 Type of the source code

The current version of hvextern supports code written as METAPOST, plain TEX, LATEX, ConTEXt, and Python. Every type has its own config file which defines the keywords for the linerange which should be printed for the preamble and the body. For example the latex config file hvextern-latex.cfg:

```
\extern@ExampleType{latex}%                    for _all_LaTeX engines
  {\string\begin\string{document\string}}%
  {\string\end\string{document\string}}%
  {\perCent StartVisiblePreamble}%
  {\perCent StopVisiblePreamble}%

% only for the sequence latex->dvips->ps2pdf
\def\hv@extern@runLATEX#1#2#3#4{% path compiler file extension
  \ifhv@extern@verbose \typeout{>>>> running #1#2 #3#4}\fi
  \ShellEscape{#1#2\space #3#4}%
  \ifhv@extern@verbose \typeout{>>>> running #1dvips #3}\fi
  \ShellEscape{#1dvips\space #3.dvi}%
  \ifhv@extern@verbose \typeout{>>>> running ps2pdf #3.ps}\fi
  \ShellEscape{#1ps2pdf\space -dAutoRotatePages=/None\space -dALLOWPSTRANSPARENCY\space  #3.ps}%
}
```

If a source needs more than running the defined compiler, it must be defined by a macro

```
\def\hv@extern@run<NAME>#1#2#3#4{% path compiler file extension
...}
```

The type of the source code can be different to the compiler, e.g. source latex, but compiler lualatex.

4.6 Output as floating object with caption and label

By default the images are not inserted as a float. This can be changed by the keyword float, a caption and a label are optional. The float type is always figure.

```
\define@boolkey{hv}[hv@extern@]{float}[true]{}
\define@key{hv}{caption}[]{\def\hv@extern@caption{#1}}
\define@key{hv}{label}[]{\def\hv@extern@label{#1}}
```

The image Figure 1 shows an example for a floting object.

```
\usepackage{pst-coxeterp}
```

```
\begin{pspicture}(-1,-1)(1,1)\Simplex[dimension=2]\end{pspicture}
%
\begin{pspicture}(-1,-1)(1,1)\Simplex[dimension=3]\end{pspicture}
%
\begin{pspicture}(-1,-1)(1,1)\Simplex[dimension=5]\end{pspicture}
%
\begin{pspicture}(-1,-1)(1,1)\Simplex[dimension=7]\end{pspicture}
%
% ------------------------------------
```

```
% Only some text to show the pagebreak
% ----------------------------------
```

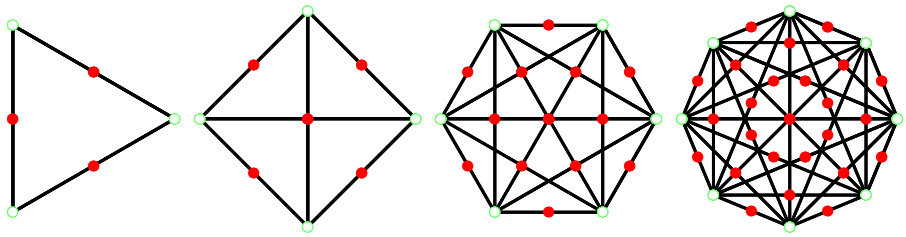


Abbildung 1: An example for Coxeter images

4.7 Output more than one page

The pages which should be printed can be defined by

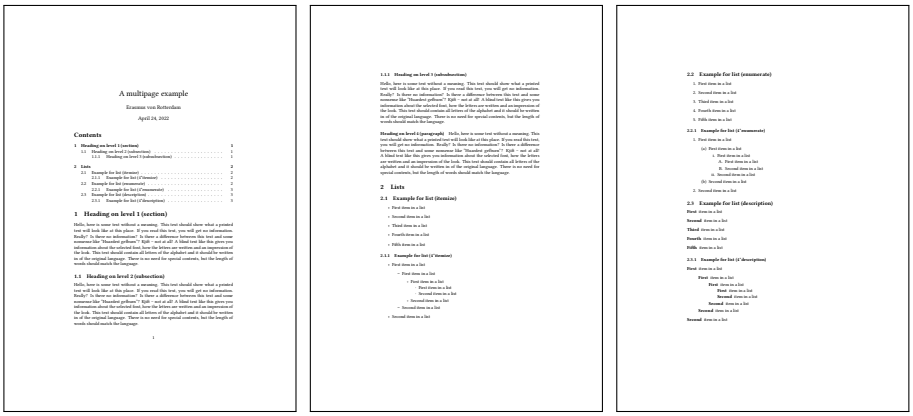
```
\define@key{hv}{pages}[1]{\def\hv@extern@pages{#1}}
\define@boolkey{hv}[hv@extern@]{frame}[true]{}
```

With `frame` the pages can be framed (internally by `\fbox`). It is leaved to the user to choose the correct image width for the pages. The following example uses:

```
pages={1,2,3},
grfOptions={width=0.3\linewidth},
compiler=lualatex, runs=2,  % for the TOC
frame,
```

```
\usepackage[american]{babel}
\usepackage{libertinus}
\usepackage{blindtext}
```

```
\title{A multipage example}
\author{Erasmus von Rotterdam}
\maketitle
\tableofcontents
\blinddocument
```



4.8 Cropping the PDF

Instead of using the documentclass `standalone`, which already crops the created PDF, one can use the optional argument `crop`.

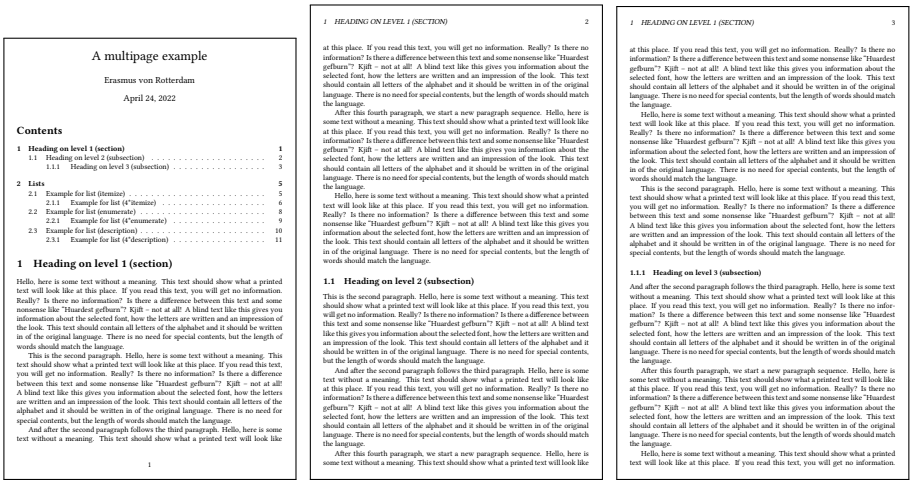
```
\define@boolkey{hv}[hv@extern@]{crop}[true]{}
\define@key{hv}{cropmargin}[2]{\def\hv@extern@cropmargin{#1 }}% length in pt
```

It is also possible to crop a document with more than one page. In this case the beginning and end of the pages should be on the same height. Otherwise the pages will have different heights after cropping (see next image). The follwoing example was created with

```
pages={1,2,3},
grfOptions={width=0.3\linewidth},
compiler=lualatex, runs=2,  % for the TOC
frame,
crop, cropmargin=5,%    5pt margin
```

```
\usepackage[american]{babel}
\usepackage{libertinus}
\usepackage{blindtext}
\pagestyle{headings}
```

```
\title{A multipage example}
\author{Erasmus von Rotterdam}
\maketitle
\tableofcontents
\Blinddocument
```



4.9 Code and output side by side

By default the code and the output is on top of each other. With setting the width of a minipage with `mpwidth` greater than 0 pt the output will be side by side.

```
\define@key{hv}{mpwidth}[0pt]{\setlength\hv@extern@mpwidth{#1}}
```

`mpwidth` is the width of the code. The rest of the line, minus 1em for the space between the minipages, will be the possible width for the output and will be calculated automatically. The two minipages are aligned by its top.

4.10 Align of the output

4.10.1 the default

```
align=\centering,
```

```
\rule{0.5\linewidth}{5mm}
```

4.10.2 Left aligned

```
align=\raggedright,
```

```
\rule{0.5\linewidth}{5mm}
```

4.10.3 Right aligned

```
align=\raggedleft,
```

```
\rule{0.5\linewidth}{5mm}
```

4.10.4 the default – side by side

```
align=\centering,  mpwidth=0.5\linewidth,
```

```
\rule{0.25\linewidth}{5mm}
```

4.10.5 Left aligned – side by side

```
align=\raggedright,  mpwidth=0.5\linewidth,
```

```
\rule{0.25\linewidth}{5mm}
```

4.10.6 Right aligned – side by side

```
align=\raggedleft,  mpwidth=0.5\linewidth,
```

```
\rule{0.25\linewidth}{5mm}
```

4.11 Inline images

By default code and image are own paragraphs. With the optional argument `inline` the created image can be part of the current line. This may make sense, if you need characters which are not part of your current font.

```
\define@boolkey{hv}[hv@extern@]{inline}[true]{\hv@extern@codefalse}
```

With `inline=true` the optional argument `code` is automatically set to false. The next Chinese characters %美好的一天are inserted as inline image without showing the code. The complete code looks like:

```
With \texttt{inline=true} the optional argument \texttt{code} is automatically set to false. The
next  Chinese characters
\begin{externalDocument}[
  compiler=xelatex, inline, runs=2, grfOptions={height=8pt},
  crop, cropmargin=0, cleanup, force, docType=latex]{voss}
\documentclass{ctexart}
\pagestyle{empty}
\begin{document}
□□□□□
\end{document}
\end{externalDocument}
are inserted as inline image without showing the code. The complete code looks like:
```

4.12 Input text instead of an image

By default the created pdf which can be, of course, only test, wil be insert by `\incudegraphics`. If you have only text as output and dont want to create a pdf you can insert this kind of output as verbatim text by setting `includegraphic=false`.

```
\define@boolkey{hv}[hv@extern@]{includegraphic}[true]{}
```

The textfile must have the same main filename with the extension `.txt`. As already mentioned, in every programming language you can get the current used filename from within the code itself. The following Perl example which calculates the Kaprekar constants uses

```
my $filename = $0;             # the current filename
$filename =~ s/\.pl//;         # without extension .pl
$filename = "${filename}.txt"; # for the output
```

Only for some completeness: a Kaprekar constant is a number A with $\max(A) - \min(A) = A$. max and min are the sorted digits of the number A: $495 = 954 - 459$.

```
my $zahl = 1;
my $anfang = 1;
my $ende = 9;
```

```
print $fh "Finding Kaprekarconstants ...\n";
while ($zahl < 8) {
  print $fh "${zahl}-stellig: ";
  foreach ($anfang...$ende) { # for every
      row $_
    @Zeichen = split(//,$_);
    $Min = join("",sort(@Zeichen));
    $Max = reverse($Min);
```

```
    $Dif=$Max-$Min;
    if($_ eq $Dif) {
      $found = 1;
      print $fh $_,",  ";
    }
  }
  if (!$found) { print $fh "---\n"; }
  else         { print $fh "\n";    }
```

```
  $found = false;
  $zahl = $zahl+1;
  $anfang = $anfang*10;
  $ende = $ende*10;
}
```

```
Finding Kaprekarconstants ...
1-stellig: ---
2-stellig:
3-stellig: 495,
4-stellig: 6174,
5-stellig:
6-stellig: 549945,  631764,
7-stellig:
```

Another example with running Lua to calculate and print the Pascal's triangle. The internal filename is available with

```
local filename = arg[0]
local shortFN = str:match("(.+)%..+")   -- delete extension
outFile = io.open(shortFN..".txt","w+") -- open external file
```

```
function nextrow(t)
  local ret = {}
  t[0], t[#t+1] = 0, 0
  for i = 1, #t do ret[i] = t[i-1] + t[i] end
  return ret
end

function triangle(n)
  t = {1}
  for i = 1, n do
    m = (n - i)
    for j = 1,m do outFile:write("    ") end
    for k = 1,i do outFile:write(string.format("%8s",t[k])) end
    outFile:write("\n")
    t = nextrow(t)
  end
end
```

```
triangle(10)
```

```
                                    1
                                1       1
                            1       2       1
                        1       3       3       1
                    1       4       6       4       1
                1       5      10      10       5       1
            1       6      15      20      15       6       1
        1       7      21      35      35      21       7       1
    1       8      28      56      70      56      28       8       1
1       9      36      84     126     126      84      36       9       1
```

4.13 Running additional external programs

For a LaTeX additional programs for bibliography, index, a.s.o. maybe needed.

```
\define@boolkey{hv}[hv@extern@]{biber}[true]{}
\define@boolkey{hv}[hv@extern@]{xindex}[true]{}
\define@key{hv}{xindexOptions}[]{\def\hv@extern@xindexOptions{#1}}
```

The `biber` run needs no additional options, but for `xindex` it maybe useful. The following examples uses

```
\begin{externalDocument}[
  compiler=lualatex, runs=2,
  pages=2,
  crop,
  xindex,
  xindexOptions={-l DE --config AU},
  mpwidth=0.6\linewidth,
  usefancyvrb=false,
  docType=latex,
  ...
]{voss}
```

```
\usepackage{makeidx}\makeindex
\usepackage{hvindex}
```

```
Sort with xindex \verb|-l DE --config AU|
\Index{Österreich} \Index{Öresund}
\Index{Ostern} \Index{Ober} \Index{Oberin}
\Index{Österreich} \Index{Öresund}
\Index{Ödem} \Index{Oligarch} \Index{Oder}
\Index{Ostern} \Index{Ober} \Index{Oberin}
\Index{Obstler} \Index{Öl} \Index{ölen}
\Index{Oder|seealso{Fluss}} \Index{Göbel}
\Index{oder} \index{Fluss!Oder}
\Index{Goethe} \Index{Göthe} \Index{Götz}
\Index{Goldmann}
\printindex
```

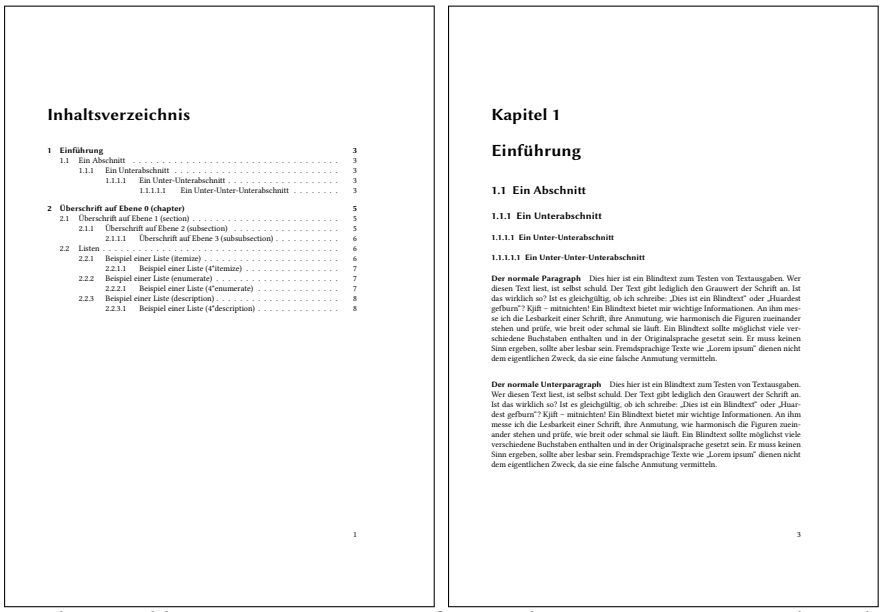
Index

4.14 Using `listings`

The default is using `\lstinputlisting` for the printed code sequences.

```
\documentclass[chapterprefix=on,parskip=half-,DIV=12,fontsize=12pt]{scrbook
    }
\DeclareNewSectionCommand[
  style=section,
  level=4,
  beforeskip=-3.25ex plus -1ex minus -.2ex,
  afterskip=1.5ex plus .2ex,
  font=\normalsize,
  indent=0pt,
  counterwithin=subsubsection
]{subsubsubsection}
\RedeclareSectionCommand[
  level=5,
  toclevel=5,
  tocindent=13em,
  tocnumwidth=5.9em,
  counterwithin=subsubsubsection
]{paragraph}
\RedeclareSectionCommand[
  level=6,
  toclevel=6,
  tocindent=15em,
  tocnumwidth=6.8em
]{subparagraph}
\setcounter{secnumdepth}{\subsubsubsectionnumdepth}
\setcounter{tocdepth}{\subsubsubsectiontocdepth}
```

```
\tableofcontents
\chapter{Einführung}
\section{Ein Abschnitt}
\subsection{Ein Unterabschnitt}
\subsubsection{Ein Unter-Unterabschnitt}
\subsubsubsection{Ein Unter-Unter-Unterabschnitt}
\paragraph{Der normale Paragraph}
\blindtext
\subparagraph{Der normale Unterparagraph}
\blindtext
\blinddocument
```



Inhaltsverzeichnis

1

Kapitel 1

Einführung

1.1 Ein Abschnitt

1.1.1 Ein Unterabschnitt

1.1.1.1 Ein Unter-Unterabschnitt

1.1.1.1.1 Ein Unter-Unter-Unterabschnitt

Der normale Paragraph Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Der normale Unterparagraph Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren miteinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3

It also possible to use `\VerbatimInput` from package `fancyvrb`. In general it makes no difference using the optional argument `usefancyvrb` or not.

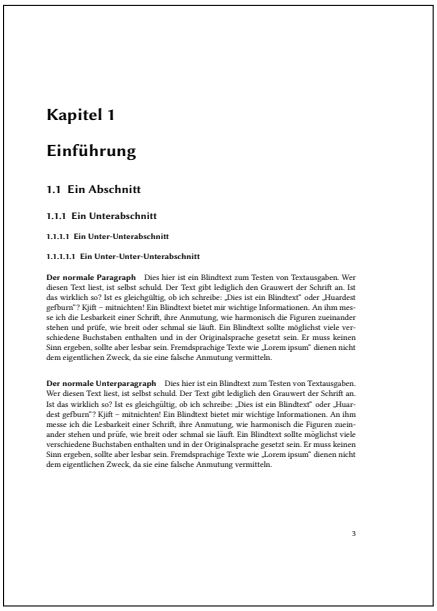
```
\documentclass[chapterprefix=on,parskip=half-,DIV=12,fontsize=12pt]{scrbook}
\DeclareNewSectionCommand[
  style=section,
  level=4,
  beforeskip=-3.25ex plus -1ex minus -.2ex,
  afterskip=1.5ex plus .2ex,
  font=\normalsize,
  indent=0pt,
  counterwithin=subsubsection
]{subsubsubsection}
\RedeclareSectionCommand[
  level=5,
  toclevel=5,
  tocindent=13em,
  tocnumwidth=5.9em,
  counterwithin=subsubsubsection
]{paragraph}
\RedeclareSectionCommand[
  level=6,
  toclevel=6,
  tocindent=15em,
  tocnumwidth=6.8em
]{subparagraph}
\setcounter{secnumdepth}{\subsubsubsectionnumdepth}
\setcounter{tocdepth}{\subsubsubsectiontocdepth}
```

```
\tableofcontents
\chapter{Einführung}
\section{Ein Abschnitt}
```

```
\subsection{Ein Unterabschnitt}
\subsubsection{Ein Unter-Unterabschnitt}
\subsubsubsection{Ein Unter-Unter-Unterabschnitt}
\paragraph{Der normale Paragraph}
\blindtext
\subparagraph{Der normale Unterparagraph}
\blindtext
\blinddocument
```

Inhaltsverzeichnis

1



Kapitel 1

Einführung

1.1 Ein Abschnitt

1.1.1 Ein Unterabschnitt

1.1.1.1 Ein Unter-Unterabschnitt

1.1.1.1.1 Ein Unter-Unter-Unterabschnitt

Der normale Paragraph Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext" oder „Huardest gefburnt"? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum" dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Der normale Unterparagraph Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext" oder „Huardest gefburnt"? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum" dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3

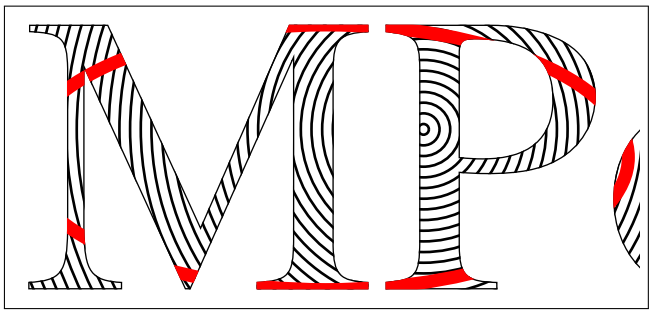
5 Supported engines

5.1 METAPOST example

```
defaultfont:="ptmr8r";
warningcheck:=0;
```

```
draw fullcircle shifted (0.5,0.6) xscaled 8cm yscaled 3.5cm
  withpen pencircle scaled 5bp withcolor red;
special( " /Times-Roman findfont 150 scalefont setfont " &
  " 0 10 moveto (MPost) false charpath clip stroke gsave 150 70 translate "
       &
  " 2 4 600 {dup 0 moveto 0 exch 0 exch 0 360 arc stroke} for grestore ");
```

voss-22.mp



5.2 plainTEX example

```
\footline={\footsc the electronic journal of combinatorics
   {\footbf 16} (2009), \#R00\hfil\footrm\folio}
```

```
\font\bigrm=cmr12 at 14pt
\centerline{\bigrm An elementary proof of the reconstruction conjecture}

\bigskip\bigskip
\centerline{D. Remifa\footnote*{Thanks to the editors of this journal!}}
\smallskip
\centerline{Department of Inconsequential Studies}
\centerline{Solatido College, North Kentucky, USA}
\centerline{\tt remifa@dis.solatido.edu}
\bigskip
\centerline{\footrm
Submitted: Jan 1, 2009; Accepted: Jan 2, 2009; Published: Jan 3, 2009}
\centerline{\footrm Mathematics Subject Classifications: 05C88, 05C89}
\bigskip\bigskip
\centerline{\bf Abstract}
\smallskip
{\narrower\noindent
The reconstruction conjecture states that the multiset of unlabeled
vertex-deleted subgraphs of a graph determines the graph, provided it
has at least 3 vertices.  A version of the problem was first stated
by Stanis\l aw Ulam.  In this paper, we show that the conjecture can
be proved by elementary methods.  It is only necessary to integrate
the Lenkle potential of the Broglington manifold over the quantum
supervacillatory measure in order to reduce the set of possible
counterexamples to a small number (less than a trillion).  A simple
computer program that implements Pipletti's classification theorem
for torsion-free Aramaic groups with simplectic socles can then
finish the remaining cases.}

\bigskip
\beginsection 1. Introduction.

This is the start of the introduction.
```

An elementary proof of the reconstruction conjecture

D. Remifa*
Department of Inconsequential Studies
Solatido College, North Kentucky, USA
remifa@dis.solatido.edu

Submitted: Jan 1, 2009; Accepted: Jan 2, 2009; Published: Jan 3, 2009
Mathematics Subject Classifications: 05C88, 05C89

Abstract

The reconstruction conjecture states that the multiset of unlabeled vertex-deleted subgraphs of a graph determines the graph, provided it has at least 3 vertices. A version of the problem was first stated by Stanislaw Ulam. In this paper, we show that the conjecture can be proved by elementary methods. It is only necessary to integrate the Lenkle potential of the Broglington manifold over the quantum supervacillatory measure in order to reduce the set of possible counterexamples to a small number (less than a trillion). A simple computer program that implements Pipletti's classification theorem for torsion-free Aramaic groups with simplectic socles can then finish the remaining cases.

1. Introduction.

This is the start of the introduction.

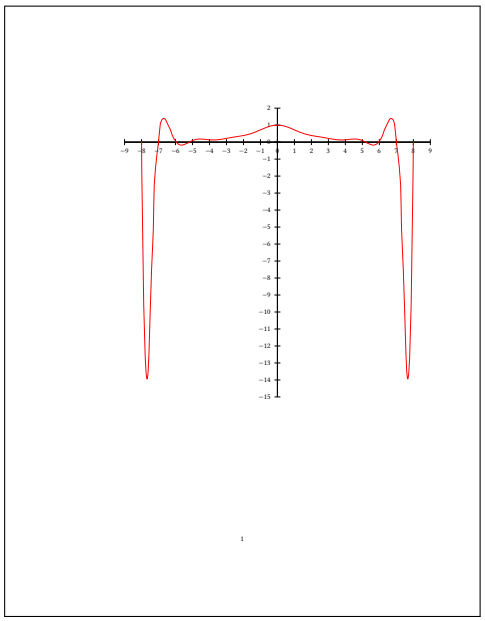
* Thanks to the editors of this journal!

5.3 LaTeX example

```
\usepackage{fontenc}\usepackage{libertinus}
\usepackage{pst-all}
```

```
\psset{unit=0.8cm}
\begin{pspicture}(-9,-15)(9,2)
\psaxes(0,0)(-9,-15)(9,2)
\psplot[algebraic,plotstyle=curve,curvature=1 1 0,
  linewidth=1pt,linecolor=red]{-8}{8}{
  1 - 3876218985722260225*x^2/10892114744073986176
    + 1497597479327145O625*x^4/174273835905183778816
    - 317095420958296875*x^6/26811359370028273664
    + 194412970920703125*x^8/214490874960226189312
    - 2090988251953125*x^10/53622718740056547328
    + 99480224609375*x^12/107245437480113094656
    - 7879638671875*x^14/697095343620735115264
    + 152587890625*x^16/2788381374482940461056}
\end{pspicture}
```

voss-24.tex

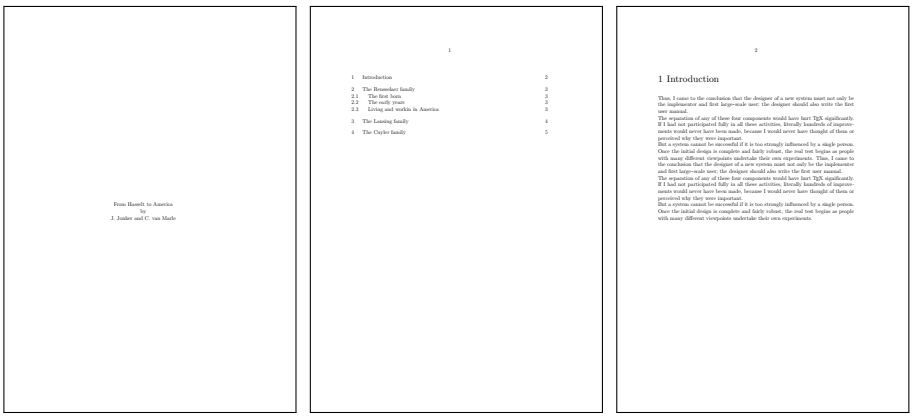


5.4 ConTEXt example

```
\definehead
  [myhead]
  [section]
\setuphead
  [myhead]
  [numberstyle=bold,
   textstyle=bold,
   before=\hairline\blank,
   after=\nowhitespace\hairline]
```

```
\startstandardmakeup
\midaligned{From Hasselt to America}
\midaligned{by}
\midaligned{J. Jonker and C. van Marle}
\stopstandardmakeup
\placecombinedlist[content]
\chapter{Introduction}
\input knuth \input knuth
\chapter[rensselaer]{The Rensselaer family}
\section{The first born}
\input knuth
\section{The early years}
... in those days Hasselt was ...
\section{Living and workin in America}
\input knuth
\chapter[lansing]{The Lansing family}
... the Lansing family was also ...
\chapter[cuyler]{The Cuyler family}
... much later Tydeman Cuyler ...
```

```
\myhead[headlines]{And the end}
foo
```



6 Other options

force=false can speed up the comiling time for the document. If a created image/-output already exists, there is no need to create it with the next run again and again.

cleanup the auxiliary files of a LaTeX-run are deleted, preset to `.aux`, `.log`. It must be a comma seperated list of the extensions of the main file, s.g. `cleanup={aux,log}`.

copyToExampleDir name of a directory for the examples, must first be created by the user himself

ExamplesDir move all examples into a directory

eps create an eps from the pdf (historical)