CONTEXT

Visualization

group: CONTEXT Support Macros

version: 1996.10.21

date: 1997 July 25

author: Hans Hagen

Although an integral part of CONTEXT, this module is one of the support modules. Its stand alone character permits use in PLAIN TEX or TEX based macropackages.

This module is still in development. Depending on my personal need and those of whoever uses it, the macros will be improved in terms of visualization, efficiency and compatibility.

```
\ifx \undefined \writestatus \input supp-mis.tex \fi
```

One of the strong points of TEX is abstraction of textual input. When macros are defined well and do what we want them to do, we will seldom need the tools present in What You See Is What You Get systems. For instance, when entering text we don't need rulers, because no manual shifting and/or alignment of text is needed. On the other hand, when we are designing macros or specifying layout elements, some insight in TEX's advanced spacing, kerning, filling, boxing and punishment abilities will be handy. That's why we've implemented a mechanism that shows some of the inner secrets of TEX.

```
\writestatus{loading}{Context Support Macros / Visualization}
```

In this module we are going to redefine some TEX primitives and PLAIN macro's. Their original meaning is saved in macros with corresponding names, preceded by `normal`. These original macros are (1) used to temporary restore the old values when needed and (2) used to prevent recursive calls in the macros that replace them.

```
\unprotect
```

`\normalhbox` `\normalvbox` `\normalvtop`

There are three types of boxes, one horizontal and two vertical in nature. As we will see later on, all three types are to be handled according to their orientation and baseline behavior. Especially `\vtop`'s need our special attention.

```
\let\normalhbox     = \hbox
\let\normalvbox     = \vbox
\let\normalvtop     = \vtop
\let\normalvcenter  = \vcenter
```

`\normalhskip` `\normalvskip`

Next come the flexible skips, which come in two flavors too. Like boxes these are handled with TEX primitives.

```
\let\normalhskip    = \hskip
\let\normalvskip    = \vskip
```

`\normalpenalty` `\normalkern`

Both penalties and kerns are taken care of by mode sensitive primitives. This means that when making them visible, we have to take the current mode into account.

```
\let\normalpenalty  = \penalty
\let\normalkern     = \kern
```

`\normalhglue` `\normalvglue`

Glues on the other hand are macro's defined in PLAIN TEX. As we will see, their definitions make the implementation of their visible counterparts a bit more TEXnical.

```
\let\normalhglue    = \hglue
\let\normalvglue    = \vglue
```

\normalmkern \normalmskip

Math mode has its own spacing primitives, preceded by `m`. Due to the relation with the current font and the way math is typeset, their unit `mu` is not compatible with other dimensions. As a result, the visual appearance of these primitives is kept primitive too.

```
\let\normalmkern    = \mkern
\let\normalmskip    = \mskip
```

\hfilneg \vfilneg

Fills can be made visible quite easy. We only need some additional negation macros. Because PLAIN TEX only offers `\hfilneg` and `\vfilneg`, we define our own alternative double `ll`'ed ones.

```
\def\hfillneg%
  {\normalhskip\!!zeropoint \!!plus-1fill\relax}

\def\vfillneg%
  {\normalvskip\!!zeropoint \!!plus-1fill\relax}
```

\normalhss \normalhfil \normalhfill \normalvss \normalvfil \normalvfill

The positive stretch primitives are used independant and in combination with `\leaders`.

```
\let\normalhss      = \hss
\let\normalhfil     = \hfil
\let\normalhfill    = \hfill
\let\normalvss      = \vss
\let\normalvfil     = \vfil
\let\normalvfill    = \vfill
```

\normalhfilneg \normalhfill.. \normalvfilneg \normalvfill..

Keep in mind that both `\hfillneg` and `\vfillneg` are not part of PLAIN TEX and therefore not documented in standard TEX documentation. They can nevertheless be used at will.

```
\let\normalhfilneg  = \hfilneg
\let\normalhfillneg = \hfillneg
\let\normalvfilneg  = \vfilneg
\let\normalvfillneg = \vfillneg
```

Visualization is not always wanted. Instead of turning this option off in those (unpredictable) situations, we just redefine a few PLAIN macros.

```
\def\rlap#1{\normalhbox to \!!zeropoint{#1\normalhss}}
\def\llap#1{\normalhbox to \!!zeropoint{\normalhss#1}}
```

```
\def~{\normalpenalty\!!tenthousand\ }
```

\makeruledbox

Ruled boxes can be typeset is many ways. Here we present just one alternative. This implementation may be a little complicated, but it supports all three kind of boxes. The next command expects a ⟨*box*⟩ specification, like:

```
\makeruledbox0
```

\baselinerule \baselinefill \baselinesmash

We can make the baseline of a box visible, both dashed and as a rule. Normally the line is drawn on top of the baseline, but a smashed alternative is offered too. If we want them all, we just say:

```
\baselineruletrue
\baselinefilltrue
\baselinesmashtrue
```

At the cost of some overhead these alternatives are implemented using `\if`'s:

```
\newif\ifbaselinerule  \baselineruletrue
\newif\ifbaselinefill  \baselinefillfalse
\newif\ifbaselinesmash \baselinesmashfalse
```

\iftoprule
\ifbottomrule
\ifleftrule
\ifrightrule

Rules can be turned on and off, but by default we have:

```
\topruletrue
\bottomruletrue
\leftruletrue
\rightruletrue
```

As we see below:

16

```
\newif\iftoprule       \topruletrue
\newif\ifbottomrule    \bottomruletrue
\newif\ifleftrule      \leftruletrue
\newif\ifrightrule     \rightruletrue
```

\boxrulewidth

The width in the surrounding rules can be specified by assigning an apropriate value to the dimension used. This module defaults the width to:

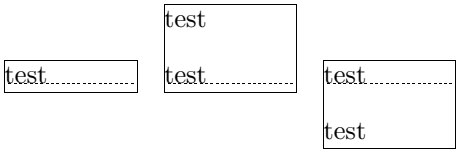
```
\boxrulewidth=.2pt
```

Although we are already low on ⟨*dimensions*⟩ it's best to spend one here, mainly because it enables easy manipulation, like multiplication by a given factor.

17

```
\newdimen\boxrulewidth \boxrulewidth=.2pt
```

The core macro `\makeruledbox` looks a bit hefty. The manipulation at the end is needed because we want to preserve both the mode and the baseline. This means that `\vtop`'s and `\vbox`'es behave the way we expect them to do.



The `\cleaders` part of the macro is responsible for the visual baseline. The `\normalhfill` belongs to this primitive too. By storing and restoring the height and depth of box `#1`, we preserve the mode.

18

```
\def\makeruledbox#1%
  {\edef\ruledheight {\the\ht#1}%
   \edef\ruleddepth  {\the\dp#1}%
   \edef\ruledwidth  {\the\wd#1}%
   \setbox\scratchbox=\normalvbox
     {\dontcomplain
      \offinterlineskip
      \hrule
        \!!height\boxrulewidth
        \iftoprule\else\!!width\!!zeropoint\fi
      \normalvskip-\boxrulewidth
      \normalhbox to \ruledwidth
        {\vrule
           \!!height\ruledheight
           \!!depth\ruleddepth
           \!!width\ifleftrule\else0\fi\boxrulewidth
         \ifdim\ruledheight>\!!zeropoint \else \baselinerulefalse \fi
         \ifdim\ruleddepth>\!!zeropoint \else \baselinerulefalse \fi
         \ifbaselinerule
```

```
            \ifdim\ruledwidth<20\boxrulewidth
              \baselinefilltrue
            \fi
            \cleaders
              \ifbaselinefill
                \hrule
                  \ifbaselinesmash
                    \!!height\boxrulewidth
                  \else
                    \!!height.5\boxrulewidth
                    \!!depth.5\boxrulewidth
                  \fi
              \else
                \normalhbox
                  {\normalhskip2.5\boxrulewidth
                   \vrule
                   \ifbaselinesmash
                     \!!height\boxrulewidth
                   \else
                     \!!height.5\boxrulewidth
                     \!!depth.5\boxrulewidth
                   \fi
                   \!!width5\boxrulewidth
                   \normalhskip2.5\boxrulewidth}%
              \fi
          \fi
          \normalhfill
          \vrule
            \!!width\ifrightrule\else0\fi\boxrulewidth}%
        \normalvskip-\boxrulewidth
        \hrule
          \!!height\boxrulewidth
          \ifbottomrule\else\!!width\!!zeropoint\fi}%
    \wd#1=\!!zeropoint
    \setbox#1=\ifhbox#1\normalhbox\else\normalvbox\fi
      {\normalhbox{\box#1\lower\ruleddepth\box\scratchbox}}%
    \ht#1=\ruledheight
    \wd#1=\ruledwidth
    \dp#1=\ruleddepth}
```

Just in case one didn't notice: the rules are in fact layed over the box. This way the contents of a box cannot visually interfere with the rules around (upon) it. A more advanced version of ruled boxes can be found in one of the core modules of CONTEXT. There we take offsets, color, rounded corners, backgrounds and alignment into account too.

`\ruledhbox`
`\ruledvbox`
`\ruledvtop`
`\ruledvcenter`

These macro's can be used instead of `\hbox`, `\vbox`, `\vtop` and, when in math mode, `\vcenter`. They just do what their names state. Using an auxiliary macro would save us a few words of memory, but it would make their appearance even more obscure.

one two three four five

```
\hbox
  {\strut
   one
   two
   \hbox{three}
   four
   five}
```

19
```
\def\ruledhbox%
  {\normalhbox\bgroup
   \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
   \normalhbox}
```

first line
second line
third line
fourth line
fifth line

```
\vbox
  {\strut
   first line  \par
   second line \par
   third line  \par
   fourth line \par
   fifth line
   \strut }
```

20
```
\def\ruledvbox%
  {\normalvbox\bgroup
   \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
   \normalvbox}
```

first line
second line
third line
fourth line
fifth line

```
\vtop
  {\strut
   first line  \par
   second line \par
   third line  \par
   fourth line \par
   fifth line
   \strut }
```

21
```
\def\ruledvtop%
  {\normalvtop\bgroup
   \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
   \normalvtop}
```

alfa
beta

alfa
beta gamma

alfa
beta

```
\hbox
  {$\vcenter{\hsize.2\hsize
      alfa \par beta}$
   $\vcenter to 3cm{\hsize.2\hsize
      alfa \par beta \par gamma}$
   $\vcenter{\hsize.2\hsize
      alfa \par beta}$}
```

22
```
\def\ruledvcenter%
  {\normalvbox\bgroup
   \dontinterfere
   \dowithnextbox
```

```
    {\scratchdimen=.5\ht\nextbox
     \advance\scratchdimen by .5\dp\nextbox
     \ht\nextbox=\scratchdimen
     \dp\nextbox=\scratchdimen
     \ruledhbox{\box\nextbox}%
     \egroup}%
  \normalvbox}
```

\ruledbox
\setruledbox

Of the next two macros the first can be used to precede a box of ones own choice. One can for instance prefix boxes with \ruledbox and afterwards — when the macro satisfy the needs — let it to \relax.

```
\ruledbox\hbox{What rules do you mean?}
```

The macro \setruledbox can be used to directly rule a box.

```
\setruledbox12=\hbox{Who's talking about rules here?}
```

At the cost of some extra macros we can implement a variant that does not need the =, but we stick to:

23
```
\def\ruledbox%
  {\dowithnextbox{\makeruledbox\nextbox\box\nextbox}}
```

24
```
\def\setruledbox#1=%
  {\dowithnextbox{\makeruledbox\nextbox\setbox#1=\nextbox}}
```

\investigate..
\investigate..
\investigate..

Before we meet the visualizing macro's, we first implement ourselves some handy utility ones. Just for the sake of efficiency and readability, we introduce some status variables, that tell us a bit more about the registers we use:

```
\ifflexible
\ifzero
\ifnegative
\ifpositive
```

These status variables are set when we call for one of the investigation macros, e.g.

```
\investigateskip\scratchskip
```

We use some dirty trick to check stretchability of ⟨*skips*⟩. Users of these macros are invited to study their exact behavior first. The positive and negative states both include zero and are in fact non-negative (≥ 0) and non-positive (≤ 0) .

25
```
\newif\ifflexible
\newif\ifzero
\newif\ifnegative
\newif\ifpositive
```

26
```
\def\investigateskip#1%
  {\relax
   \scratchdimen=#1\relax
   \edef\!!stringa{\the\scratchdimen}%
   \edef\!!stringb{\the#1}%
   \ifx\!!stringa\!!stringb \flexiblefalse \else \flexibletrue \fi
   \ifdim#1=\!!zeropoint\relax
     \zerotrue      \else
     \zerofalse     \fi
```

```
    \ifdim#1<\!!zeropoint\relax
      \positivefalse \else
      \positivetrue  \fi
    \ifdim#1>\!!zeropoint\relax
      \negativefalse \else
      \negativetrue  \fi}
```

27

```
\def\investigatecount#1%
  {\relax
   \flexiblefalse
   \ifnum#1=0
     \zerotrue      \else
     \zerofalse     \fi
   \ifnum#1<0
     \positivefalse \else
     \positivetrue  \fi
   \ifnum#1>0
     \negativefalse \else
     \negativetrue  \fi}
```

28

```
\def\investigatemuskip#1%
  {\relax
   \edef\!!stringa{\the\scratchmuskip}%
   \edef\!!stringb{0mu}%
   \def\!!stringc##1##2\\{##1}%
   \expandafter\edef\expandafter\!!stringc\expandafter
     {\expandafter\!!stringc\!!stringa\\}%
   \edef\!!stringd{-}%
   \flexiblefalse
   \ifx\!!stringa\!!stringb
     \zerotrue
     \negativefalse
     \positivefalse
   \else
     \zerofalse
     \ifx\!!stringc\!!stringd
       \positivefalse
       \negativetrue
     \else
       \positivetrue
       \negativefalse
     \fi
   \fi}
```

`\dontinterfere`

Indentation, left and/or right skips, redefinition of `\par` and assignments to `\everypar` can lead to unwanted results. We can therefore turn all those things off with `\dontinterfere`.

29

```
\def\dontinterfere%
  {\everypar  = {}%
   \let\par   = \endgraf
   \parindent = \!!zeropoint
   \parskip   = \!!zeropoint
   \leftskip  = \!!zeropoint
   \rightskip = \!!zeropoint
```

```
\relax}
```

\dontcomplain

In this module we do a lot of box manipulations. Because we don't want to be confronted with to many over- and underfull messages we introduce \dontcomplain.

30
```
\def\dontcomplain%
  {\hbadness  = \!!tenthousand
   \hfuzz     = \maxdimen
   \vbadness  = \!!tenthousand
   \vfuzz     = \maxdimen}
```

Now the neccessary utility macros are defined, we can make a start with the visualizing ones. The implementation of these macros is a compromise between readability, efficiency of coding and processing speed. Sometimes we do in steps what could have been done in combination, sometimes we use a few boxes more or less then actually needed, and more than once one can find the same piece of rule drawing code twice.

\ifcenteredv..
\normalvcue

Depending on the context, one can force visual vertical cues being centered along \hsize or being put at the current position. Although centering often looks better, we've chosen the second alternative as default. The main reason for doing so is that often when we don't set the \hsize ourselves, TEX takes the value of the surrounding box. As a result the visual cues can migrate outside the current context.

This behavior is accomplished by a small but effective auxiliary macro, which behavior can be influenced by the boolean \centeredvcue. By saying

```
\centeredvcuetrue
```

one turns centering on. As said, we turn it off.

31
```
\newif\ifcenteredvcue  \centeredvcuefalse
```

32
```
\def\normalvcue#1%
  {\normalhbox \ifcenteredvcue to \hsize \fi {\normalhss#1\normalhss}}
```

We could have used the more robust version

```
\def\normalvcue%
  {\normalhbox \ifcenteredvcue to \hsize \fi
   \bgroup\bgroup\normalhss
   \aftergroup\normalhss\aftergroup\egroup
   \let\next=}
```

or the probably best one:

```
\def\normalvcue%
  {\hbox \ifcenteredvcue to \hsize
     \bgroup\bgroup\normalhss
     \aftergroup\normalhss\aftergroup\egroup
   \else
     \bgroup
   \fi
   \let\next=}
```

Because we don't have to preserve ⟨*catcodes*⟩ and only use small arguments, we stick to the first alternative.

\testrulewidth

We build our visual cues out of rules. At the cost of a much bigger DVI file, this is to be prefered over using characters (1) because we cannot be sure of their availability and (2) because their dimensions are fixed.

As with ruled boxes, we use a ⟨*dimension*⟩ to specify the width of the ruled elements. This dimension defaults to:

```
\testrulewidth=\boxrulewidth
```

Because we prefer whole numbers for specifying the dimensions, we often use even multiples of `\testrulewidth`.

\visiblestre..

A second variable is introduced because of the stretch components of ⟨*skips*⟩. At the cost of some accuracy we can make this stretch visible.

```
\visiblestretchtrue
```

33

```
\newdimen\testrulewidth  \testrulewidth=\boxrulewidth
\newif\ifvisiblestretch  \visiblestretchfalse
```

\ruledhss
\ruledhfil
\ruledhfilneg
\ruledhfill
\ruledhfillneg

We start with the easiest part, the fills. The scheme we follow is *visual filling – going back – normal filling*. Visualizing is implemented using `\cleaders`. Because the ⟨*box*⟩ that follows this command is constructed only once, the `\copy` is not really a prerequisite. We prefer using a `\normalhbox` here instead of a `\hbox`.

34

```
\def\setvisiblehfilbox#1\to#2#3#4%
  {\setbox#1=\normalhbox
     {\vrule
        \!!width#2\testrulewidth
        \!!height#3\testrulewidth
        \!!depth#4\testrulewidth}%
   \smashbox#1}
```

35

```
\def\doruledhfiller#1#2#3#4%
  {#1#2%
   \bgroup
     \dontinterfere
     \dontcomplain
     \setvisiblehfilbox0\to{4}{#3}{#4}%
     \setvisiblehfilbox2\to422%
     \copy0\copy2
     \bgroup
       \setvisiblehfilbox0\to422%
       \cleaders
         \normalhbox to 12\testrulewidth
           {\normalhss\copy0\normalhss}%
         #1%
     \egroup
     \setbox0=\normalhbox
       {\normalhskip-4\testrulewidth\copy0\copy2}%
     \smashbox0
     \box0
   \egroup}
```

The horizontal fillers differ in their boundary visualization. Watch the small dots. Fillers can be combined within reasonable margins.

\hss test

\hfil test

\hfill test

\hfil\hfil . test \hfil

The negative counterparts are visualizes, but seldom become visible, apart from their boundaries.

\hfilneg test

\hfillneg test

Although leaders are used for visualizing, they are visualized themselves correctly as the next example shows.

All five substitutions use the same auxiliary macro. Watch the positive first – negative next approach.

```
\def\ruledhss%
  {\doruledhfiller\normalhss\normalhfilneg{0}{0}}

\def\ruledhfil%
  {\doruledhfiller\normalhfil\normalhfilneg{10}{-6}}

\def\ruledhfill%
  {\doruledhfiller\normalhfill\normalhfillneg{18}{-14}}

\def\ruledhfilneg%
  {\doruledhfiller\normalhfilneg\normalhfil{-6}{10}}

\def\ruledhfillneg%
  {\doruledhfiller\normalhfillneg\normalhfill{-14}{18}}
```

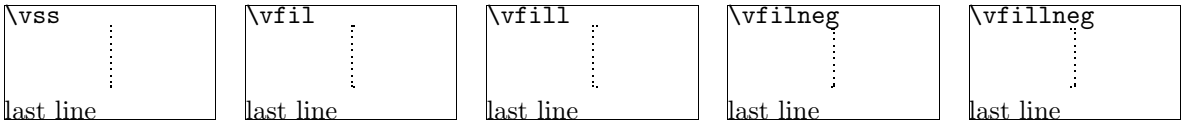
\ruledvss \ruledvfil \ruledvfilneg \ruledvfill \ruledvfillneg

The vertical mode commands adopt the same visualization scheme, but are implemented in a slightly different way.

```
\def\setvisiblevfilbox#1\to#2#3#4%
  {\setbox#1=\normalhbox
     {\vrule
        \!!width#2\testrulewidth
        \!!height#3\testrulewidth
        \!!depth#4\testrulewidth}%
   \smashbox#1}%

\def\doruledvfiller#1#2#3%
  {#1#2%
   \bgroup
     \dontinterfere
     \dontcomplain
     \offinterlineskip
     \setvisiblevfilbox0\to422%
     \setbox2=\normalvcue
```

```
    {\normalhskip -#3\testrulewidth\copy0}%
  \smashbox2
  \copy2
  \bgroup
    \setbox2=\normalvcue
      {\normalhskip -2\testrulewidth\copy0}%
    \smashbox2
    \copy2
    \cleaders
      \normalvbox to 12\testrulewidth
        {\normalvss\copy2\normalvss}%
      #1%
    \setbox2=\normalvbox
      {\normalvskip-2\testrulewidth\copy2}%
    \smashbox2
    \box2
  \egroup
  \box2
\egroup}
```

Because they act the same as their horizontal counterparts we only show a few examples.



Keep in mind that \vfillneg is not part of PLAIN TEX, but are mimmicked by a macro.

43
```
\def\ruledvss%
  {\doruledvfiller\normalvss\normalvfilneg{2}}
```

44
```
\def\ruledvfil%
  {\doruledvfiller\normalvfil\normalvfilneg{-4}}
```

45
```
\def\ruledvfill%
  {\doruledvfiller\normalvfill\normalvfillneg{-12}}
```

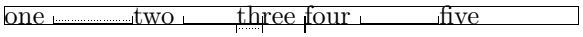
46
```
\def\ruledvfilneg%
  {\doruledvfiller\normalvfilneg\normalvfil{8}}
```

47
```
\def\ruledvfillneg%
  {\doruledvfiller\normalvfillneg\normalvfill{16}}
```

\ruledhskip Skips differ from kerns in two important aspects:

- line and pagebreaks are allowed at a skip
- skips can have a positive and/or negative stretchcomponent

Stated a bit different: kerns are fixed skips at which no line or pagebreak can occur. Because skips have a more open character, they are visualized in a open way.




```
one
\hskip +30pt plus 5pt
two
\hskip +30pt
\hskip -10pt plus 5pt
three
\hskip   0pt
four
\hskip +30pt
five
```

When skips have a stretch component, this is visualized by means of a dashed line. Positive skips are on top of the baseline, negative ones are below it. This way we can show the combined results. An alternative visualization of stretch could be drawing the mid line over a length of the stretch, in positive or negative direction.

48
```
\def\doruledhskip%
  {\relax
   \dontinterfere
   \dontcomplain
   \investigateskip\scratchskip
   \ifzero
     \setbox0=\normalhbox
       {\normalhskip-\testrulewidth
        \vrule
          \!!width4\testrulewidth
          \!!height16\testrulewidth
          \!!depth16\testrulewidth}%
   \else
     \setbox0=\normalhbox to \ifnegative-\fi\scratchskip
       {\vrule
          \!!width2\testrulewidth
          \ifnegative\!!depth\else\!!height\fi16\testrulewidth
        \cleaders
          \hrule
            \ifnegative
              \!!depth2\testrulewidth
              \!!height\!!zeropoint
            \else
              \!!height2\testrulewidth
              \!!depth\!!zeropoint
            \fi
          \normalhfill
        \ifflexible
          \normalhskip\ifnegative\else-\fi\scratchskip
          \normalhskip2\testrulewidth
          \cleaders
            \normalhbox
              {\normalhskip 2\testrulewidth
               \vrule
                 \!!width2\testrulewidth
                 \!!height\ifnegative-7\else9\fi\testrulewidth
                 \!!depth\ifnegative9\else-7\fi\testrulewidth
```

```
                \normalhskip 2\testrulewidth}%
              \normalhfill
          \fi
          \vrule
            \!!width2\testrulewidth
            \ifnegative\!!depth\else\!!height\fi16\testrulewidth}%
        \setbox0=\normalhbox
          {\ifnegative\else\normalhskip-\scratchskip\fi
           \box0}%
      \fi
      \smashbox0%
      \ifvisiblestretch \else
        \flexiblefalse
      \fi
      \ifflexible
        % breaks ok but small displacements can occur
        \skip2=\scratchskip
        \advance\skip2 by -1\scratchskip
        \divide\skip2 by 2
        \advance\scratchskip by -\skip2
        \normalhskip\scratchskip
        \normalpenalty\!!tenthousand
        \box0
        \normalhskip\skip2
      \else
        \normalhskip\scratchskip
        \box0
      \fi
      \egroup}
```

49

```
\def\ruledhskip%
  {\bgroup
   \afterassignment\doruledhskip
   \scratchskip=}
```

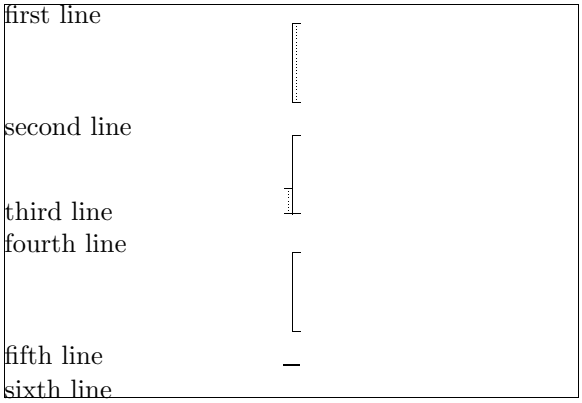
The visual skip is located at a feasible point. Normally this does not interfere with the normaltypesetting process. The next examples show (1) the default behavior, (2) the (not entirely correct) distributed stretch and (3) the way the text is typeset without cues.

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test test

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test test

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test test

\ruledvskip We are less fortunate when implementing the vertical skips. This is a direct result of interference between the boxes that visualize the skip and skip removal at a pagebreak. Normally skips disappear at the top of a page, but not of course when visualized in a \vbox. A quite perfect simulation could have been built if we would have had available two more primitives: \hnop and \vnop. These new primitives could stand for boxes that are visible but are not taken into account in any way. They are there for us, but not for TEX.




```
first line
\vskip +30pt plus 5pt
second line
\vskip +30pt
\vskip -10pt plus 5pt
third line
\par
fourth line
\vskip +30pt
fifth line
\vskip   0pt
sixth line
```

We have to postpone \prevdepth. Although this precaution probably is not completely waterproof, it works quite well.

```
\def\dodoruledvskip%
  {\nextdepth=\prevdepth
   \dontinterfere
   \dontcomplain
   \offinterlineskip
   \investigateskip\scratchskip
   \ifzero
     \setbox0=\normalvcue
       {\vrule
          \!!width32\testrulewidth
          \!!height2\testrulewidth
          \!!depth2\testrulewidth}%
   \else
     \setbox0=\normalvbox to \ifnegative-\fi\scratchskip
       {\hrule
          \!!width16\testrulewidth
          \!!height2\testrulewidth
        \ifflexible
          \cleaders
            \normalhbox to 16\testrulewidth
              {\normalhss
               \normalvbox
                 {\normalvskip 2\testrulewidth
                  \hrule
                  \!!width2\testrulewidth
                  \!!height2\testrulewidth
                  \normalvskip 2\testrulewidth}%
               \normalhss}%
            \normalvfill
        \else
```

```
        \normalvfill
      \fi
      \hrule
        \!!width16\testrulewidth
        \!!height2\testrulewidth}%
    \setbox2=\normalvbox to \ht0
      {\hrule
        \!!width2\testrulewidth
        \!!height\ht0}%
    \ifnegative
      \ht0=\!!zeropoint
      \setbox0=\normalhbox
        {\normalhskip2\testrulewidth % will be improved
         \normalhskip-\wd0\box0}%
    \fi
    \smashbox0%
    \smashbox2%
    \setbox0=\normalvcue
      {\box2\box0}%
    \setbox0=\normalvbox
      {\ifnegative\normalvskip\scratchskip\fi\box0}%
    \smashbox0%
  \fi
  \ifvisiblestretch
    \ifflexible
      \skip2=\scratchskip
      \advance\skip2 by -1\scratchskip
      \divide\skip2 by 2
      \advance\scratchskip by -\skip2
      \normalvskip\skip2
    \fi
  \fi
  \normalpenalty\!!tenthousand
  \box0
  \prevdepth=\nextdepth % not \dp0=\nextdepth
  \normalvskip\scratchskip}
```

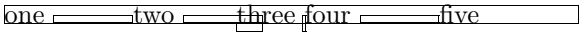
We try to avoid interfering at the top of a page. Of course we only do so when we are in the main vertical list.

51
```
\def\doruledvskip%
  {\endgraf % \par
   \ifdim\pagegoal=\maxdimen
     \ifinner
       \dodoruledvskip
     \fi
   \else
     \dodoruledvskip
   \fi
   \egroup}
```

52
```
\def\ruledvskip%
  {\bgroup
   \afterassignment\doruledvskip
```

```
\scratchskip=}
```

\ruledkern The macros that implement the kerns are a bit more complicated than needed, because they also serve the visualization of glue, our PLAIN defined kerns with stretch or shrink. We've implemented both horizontal and vertical kerns as ruled boxes.




```
one
\kern +30pt
two
\kern +30pt
\kern -10pt
three
\kern   0pt
four
\kern +30pt
five
```

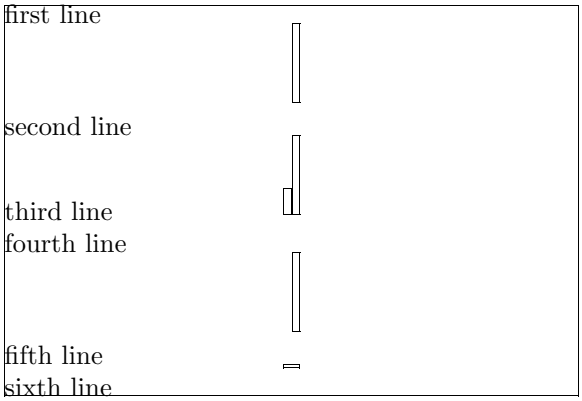
Positive and negative kerns are placed on top or below the baseline, so we are able to track their added result. We didn't mention spacings of 0 pt yet. Zero values are visualized a bit different, because we want to see them anyhow.

53
```
\def\doruledhkern%
  {\dontinterfere
   \dontcomplain
   \baselinerulefalse
   \investigateskip\scratchskip
   \boxrulewidth=2\testrulewidth
   \ifzero
     \setbox0=\ruledhbox to 8\testrulewidth
       {\vrule
          \!!width\!!zeropoint
          \!!height16\testrulewidth
          \!!depth16\testrulewidth}%
     \setbox0=\normalhbox
       {\normalhskip-4\testrulewidth\box0}%
   \else
     \setbox0=\ruledhbox to \ifnegative-\fi\scratchskip
       {\vrule
          \!!width\!!zeropoint
          \ifnegative\!!depth\else\!!height\fi16\testrulewidth
        \ifflexible
          \normalhskip2\testrulewidth
          \cleaders
            \normalhbox
              {\normalhskip 2\testrulewidth
               \vrule
                 \!!width2\testrulewidth
                 \!!height\ifnegative-7\else9\fi\testrulewidth
                 \!!depth\ifnegative9\else-7\fi\testrulewidth
               \normalhskip 2\testrulewidth}%
          \normalhfill
        \else
          \normalhfill
        \fi}%
```

```
    \testrulewidth=2\testrulewidth
    \setbox0=\ruledhbox{\box0}%  \make...
  \fi
  \smashbox0%
  \normalpenalty\!!tenthousand
  \normalhbox to \!!zeropoint
    {\ifnegative\normalhskip1\scratchskip\fi
     \box0}%
  \afterwards\scratchskip
  \egroup}
```

54
```
\def\ruledhkern#1%
  {\bgroup
   \let\afterwards=#1\relax
   \afterassignment\doruledhkern
   \scratchskip=}
```

After having seen the horizontal ones, the vertical kerns will not surprise us. In this example we use `\par` to switch to vertical mode.




```
first line
\par \kern +30pt
second line
\par \kern +30pt
\par \kern -10pt
third line
\par
fourth line
\par \kern +30pt
fifth line
\par \kern   0pt
sixth line
```

Like before, we have to postpone `\prevdepth`. If we leave out this trick, we got ourselves some wrong spacing.

55
```
\def\dodoruledvkern%
  {\nextdepth=\prevdepth
   \dontinterfere
   \dontcomplain
   \baselinerulefalse
   \offinterlineskip
   \investigateskip\scratchskip
   \boxrulewidth=2\testrulewidth
   \ifzero
     \setbox0=\ruledhbox to 32\testrulewidth
       {\vrule
          \!!width\!!zeropoint
          \!!height4\testrulewidth
          \!!depth4\testrulewidth}%
   \else
     \setbox0=\ruledvbox to \ifnegative-\fi\scratchskip
       {\hsize16\testrulewidth
        \ifflexible
```

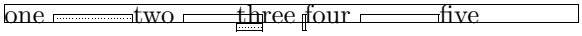
```
          \cleaders
            \normalhbox to 16\testrulewidth
              {\normalhss
               \normalvbox
                 {\normalvskip 2\testrulewidth
                  \hrule
                    \!!width2\testrulewidth
                    \!!height2\testrulewidth
                  \normalvskip 2\testrulewidth}%
               \normalhss}%
            \normalvfill
        \else
          \vrule
            \!!width\!!zeropoint
            \!!height\ifnegative-\fi\scratchskip
          \normalhfill
        \fi}
    \fi
    \testrulewidth=2\testrulewidth
    \setbox0=\ruledvbox{\box0}%  \make...
    \smashbox0%
    \setbox0=\normalvbox
      {\ifnegative\normalvskip\scratchskip\fi
       \normalvcue
        {\ifnegative\normalhskip-16\testrulewidth\fi\box0}}%
    \smashbox0%
    \normalpenalty\!!tenthousand
    \box0
    \prevdepth=\nextdepth} % not \dp0=\nextdepth

56  \def\doruledvkern%
      {\ifdim\pagegoal=\maxdimen
         \ifinner
           \dodoruledvkern
         \fi
       \else
         \dodoruledvkern
       \fi
       \afterwards\scratchskip
       \egroup}

57  \def\ruledvkern#1%
      {\bgroup
       \let\afterwards=#1\relax
       \afterassignment\doruledvkern
       \scratchskip=}

58  \def\ruledkern%
      {\ifvmode
         \let\next=\ruledvkern
       \else
         \let\next=\ruledhkern
       \fi
       \next\normalkern}
```

A a bit more TEXnic solution is:

```
\def\ruledkern%
  {\csname ruled\ifvmode v\else h\fi kern\endcsname\normalkern}
```

\ruledhglue
\ruledvglue

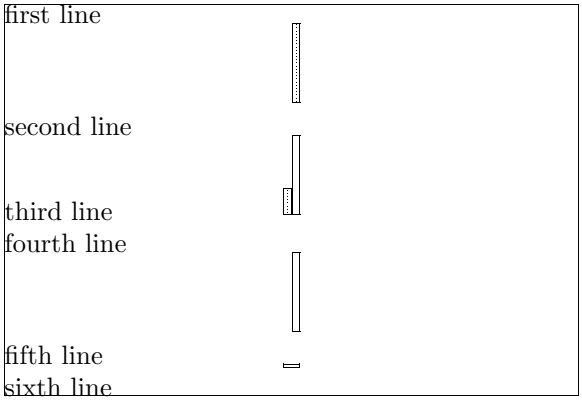
The non-primitive glue commands are treated as kerns with stretch. This stretch is presented as a dashed line. I have to admit that until now, I've never used these glue commands.




```
one
\hglue +30pt plus 5pt
two
\hglue +30pt
\hglue -10pt plus 5pt
three
\hglue   0pt
four
\hglue +30pt
five
```

59
```
\def\doruledhglue%
  {\leavevmode
   \scratchcounter=\spacefactor
   \vrule\!!width\!!zeropoint
   \normalpenalty\!!tenthousand
   \ruledhkern\normalhskip\scratchskip
   \spacefactor=\scratchcounter
   \egroup}
```

60
```
\def\ruledhglue%
  {\bgroup
   \afterassignment\doruledhglue\scratchskip=}
```




```
first line
\vglue +30pt plus 5pt
second line
\vglue +30pt
\vglue -10pt plus 5pt
third line
\par
fourth line
\vglue +30pt
fifth line
\vglue   0pt
sixth line
```

61
```
\def\doruledvglue%
  {\endgraf % \par
   \nextdepth=\prevdepth
   \hrule\!!height\!!zeropoint
   \normalpenalty\!!tenthousand
   \ruledvkern\normalvskip\scratchskip
   \prevdepth=\nextdepth
   \egroup}
```

```
\def\ruledvglue%
  {\bgroup
   \afterassignment\doruledvglue\scratchskip=}
```

\ruledmkern
\ruledmskip

Mathematical kerns and skips are specified in mu. This font related unit is incompatible with those of ⟨*dimensions*⟩ and ⟨*skips*⟩. Because in math mode spacing is often a very subtle matter, we've used a very simple, not overloaded way to show them.

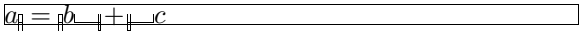
```
\def\dodoruledmkern#1%
  {\dontinterfere
   \dontcomplain
   \setbox0=\normalhbox
     {$\normalmkern\ifnegative-\fi\scratchmuskip$}%
   \setbox0=\normalhbox to \wd0
     {\vrule
        \!!height16\testrulewidth
        \!!depth16\testrulewidth
        \!!width\testrulewidth
      \leaders
        \hrule
          \!!height\ifpositive16\else-14\fi\testrulewidth
          \!!depth\ifpositive-14\else16\fi\testrulewidth
        \normalhfill
      \ifflexible
        \normalhskip-\wd0
        \leaders
          \hrule
            \!!height\testrulewidth
            \!!depth\testrulewidth
          \normalhfill
      \fi
      \vrule
        \!!height16\testrulewidth
        \!!depth16\testrulewidth
        \!!width\testrulewidth}%
   \smashbox0%
   \ifnegative
      #1\scratchmuskip
     \box0
   \else
     \box0
      #1\scratchmuskip
   \fi
   \egroup}
```

$a \mkern3mu = \mkern3mu b \quad \mkern-2mu + \mkern-2mu \quad c$

```
$a \mkern3mu = \mkern3mu
 b \quad
 \mkern-2mu + \mkern-2mu
 \quad c$
```

```
\def\doruledmkern%
  {\investigatemuskip\scratchmuskip
   \flexiblefalse
```

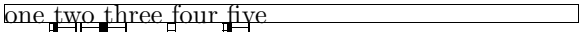
```
  \dodoruledmkern\normalmkern}

\def\ruledmkern%
  {\bgroup
   \afterassignment\doruledmkern\scratchmuskip=}
```



```
$a \mskip3mu = \mskip3mu
 b \quad
 \mskip-2mu + \mskip-2mu
 \quad c$
```

```
\def\doruledmskip%
  {\investigatemuskip\scratchmuskip
   \flexibletrue
   \dodoruledmkern\normalmskip}

\def\ruledmskip%
  {\bgroup
   \afterassignment\doruledmskip\scratchmuskip=}
```

`\penalty` After presenting fills, skip, kerns and glue we've come to see penalties. In the first implementation — most of the time needed to develop this set of macros went into testing different types of visualization — penalties were mere small blocks with one black half, depending on the sign. This most recent version also gives an indication of the amount of penalty. Penalties can go from less than −10000 to over +10000, and their behavior is somewhat non-lineair, with some values having special meanings. We therefore decided not to use its value for a lineair indicator.



```
one
\penalty +100
two
\penalty +100
\penalty -100
three
\penalty    0
four
\penalty +100
five
```

The small sticks at the side of the penalty indicate it size. The next example shows the positive and negative penalties of 0, 1, 10, 100, 1000 and 10000.

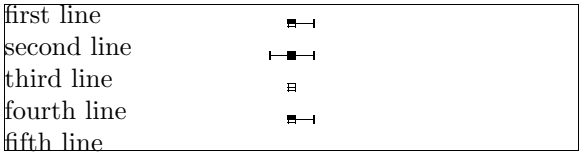
This way stacked penalties of different severance can be shown in combination.

```
\def\setruledpenaltybox#1#2#3#4#5#6%
  {\setbox#1=\normalhbox
     {\ifnum#2=0 \else
        \ifnum#2>0
          \def\sign{+}%
        \else
```

```
          \def\sign{-}%
        \fi
        \dimen0=\ifnum\sign#2>9999
                  28\else
                \ifnum\sign#2>999
                  22\else
                \ifnum\sign#2>99
                  16\else
                \ifnum\sign#2>9
                  10\else
                  4
                \fi\fi\fi\fi \testrulewidth
        \ifnum#2<0
          \normalhskip-\dimen0
          \normalhskip-2\testrulewidth
          \vrule
            \!!width2\testrulewidth
            \!!height#3\testrulewidth
            \!!depth#4\testrulewidth
        \fi
        \vrule
          \!!width\dimen0
          \!!height#5\testrulewidth
          \!!depth#6\testrulewidth
        \ifnum#2>0
          \vrule
            \!!width2\testrulewidth
            \!!height#3\testrulewidth
            \!!depth#4\testrulewidth
         \fi
      \fi}%
    \smashbox#1}

69  \def\doruledhpenalty%
      {\dontinterfere
       \dontcomplain
       \investigatecount\scratchcounter
       \testrulewidth=2\testrulewidth
       \boxrulewidth=\testrulewidth
       \setbox0=\ruledhbox to 8\testrulewidth
         {\ifnegative\else\normalhss\fi
          \vrule
            \!!depth8\testrulewidth
            \!!width\ifzero0\else4\fi\testrulewidth
          \ifpositive\else\normalhss\fi}%
       \setruledpenaltybox{2}{\scratchcounter}{0}{8}{-3.5}{4.5}%
       \normalpenalty\!!tenthousand
       \setbox0=\normalhbox
         {\normalhskip-4\testrulewidth
          \ifnegative
            \box2\box0
          \else
            \box0\box2
```

```
      \fi}%
    \smashbox0%
    \box0
    \normalpenalty\scratchcounter
    \egroup}

70  \def\ruledhpenalty%
      {\bgroup
       \afterassignment\doruledhpenalty
       \scratchcounter=}
```

The size of a vertical penalty is also shown on the horizontal axis. This way there is less interference with the often preceding or following skips and kerns.




```
first line
\par \penalty +100
second line
\par \penalty +100
\par \penalty -100
third line
\par \penalty    0
fourth line
\par \penalty +100
fifth line
```

```
71  \def\doruledvpenalty%
      {\ifdim\pagegoal=\maxdimen
       \else
         \nextdepth=\prevdepth
         \dontinterfere
         \dontcomplain
         \investigatecount\scratchcounter
         \testrulewidth=2\testrulewidth
         \boxrulewidth=\testrulewidth
         \setbox0=\ruledhbox
           {\vrule
              \!!height4\testrulewidth
              \!!depth4\testrulewidth
              \!!width\!!zeropoint
            \vrule
              \!!height\ifnegative.5\else4\fi\testrulewidth
              \!!depth\ifpositive.5\else4\fi\testrulewidth
              \!!width8\testrulewidth}%
         \setruledpenaltybox{2}{\scratchcounter}{4}{4}{.5}{.5}%
         \setbox0=\normalhbox
           {\normalhskip-4\testrulewidth
            \ifnegative
              \box2\box0
            \else
              \box0\box2
            \fi
            \normalhss}%
         \smashbox0%
```

```
      \normalpenalty\!!tenthousand
      \nointerlineskip
      \dp0=\nextdepth  % not \prevdepth=\nextdepth
      \normalvbox
        {\normalvcue{\box0}}%
    \fi
    \normalpenalty\scratchcounter
    \egroup}
```

72
```
\def\ruledvpenalty%
  {\bgroup
   \afterassignment\doruledvpenalty
   \scratchcounter=}
```

73
```
\def\ruledpenalty%
  {\ifvmode
     \let\next=\ruledvpenalty
   \else
     \let\next=\ruledhpenalty
   \fi
   \next}
```

At the cost of some more tokens, a bit more clever implementation would be:

```
\def\ruledpenalty%
  {\csname ruled\ifvmode v\else h\fi penalty\endcsname}
```

\showfils
\dontshowfils
\showboxes
\dontshowboxes
\showskips
\dontshowskips
\showpenalties
\dontshowpen..

For those who want to manipulate the visual cues in detail, we have grouped them.

```
\def\showfils%
  {\let\hss      = \ruledhss
   \let\hfil     = \ruledhfil
   \let\hfill    = \ruledhfill
   \let\hfilneg  = \ruledhfilneg
```

74
```
   \let\hfillneg = \ruledhfillneg
   \let\vss      = \ruledvss
   \let\vfil     = \ruledvfil
   \let\vfill    = \ruledvfill
   \let\vfilneg  = \ruledvfilneg
   \let\vfillneg = \ruledvfillneg}
```

75
```
\def\dontshowfils%
  {\let\hss      = \normalhss
   \let\hfil     = \normalhfil
   \let\hfill    = \normalhfill
   \let\hfilneg  = \normalhfilneg
   \let\hfillneg = \normalhfillneg
   \let\vss      = \normalvss
   \let\vfil     = \normalvfil
   \let\vfill    = \normalvfill
   \let\vfilneg  = \normalvfilneg
   \let\vfillneg = \normalvfillneg}
```

76
```
\def\showboxes%
  {\baselineruletrue
```

```
    \let\hbox     = \ruledhbox
    \let\vbox     = \ruledvbox
    \let\vtop     = \ruledvtop
    \let\vcenter  = \ruledvcenter}
```

77
```
\def\dontshowboxes%
  {\let\hbox     = \normalhbox
   \let\vbox     = \normalvbox
   \let\vtop     = \normalvtop
   \let\vcenter  = \normalvcenter}
```

78
```
\def\showskips%
  {\let\hskip    = \ruledhskip
   \let\vskip    = \ruledvskip
   \let\kern     = \ruledkern
   \let\mskip    = \ruledmskip
   \let\mkern    = \ruledmkern
   \let\hglue    = \ruledhglue
   \let\vglue    = \ruledvglue}
```

79
```
\def\dontshowskips%
  {\let\hskip    = \normalhskip
   \let\vskip    = \normalvskip
   \let\kern     = \normalkern
   \let\mskip    = \normalmskip
   \let\mkern    = \normalmkern
   \let\hglue    = \normalhglue
   \let\vglue    = \normalvglue}
```

80
```
\def\showpenalties%
  {\let\penalty  = \ruledpenalty}
```

81
```
\def\dontshowpenalties%
  {\let\penalty  = \normalpenalty}
```

All these nice options come together in two macros. The first one turns the options on, the second turnes them off. Both macros only do their job when we are actually showing the composition.

\showingcomp..
\showcomposi..
\dontshowcom..
\

```
\showingcompositiontrue
\showcomposition
```

Because the output routine can do tricky things, like multiple column typesetting and manipulation of the pagebody, shifting things around and so on, the macro `\dontshowcomposition` best can be called when we enter this routine. Too much visual cues just don't make sense. In CONTEXT this has been taken care of.

82
```
\newif\ifshowingcomposition
```

83
```
\def\showcomposition%
  {\ifshowingcomposition
     \showfils
     \showboxes
     \showskips
     \showpenalties
   \fi}
```

```
\def\dontshowcomposition%
  {\ifshowingcomposition
     \dontshowfils
     \dontshowboxes
     \dontshowskips
     \dontshowpenalties
   \fi}
```

\showmakeup
\defaulttest..

Just to make things even more easy, we have defined:

```
\showmakeup
```

For the sake of those who don't (yet) use CONTEXT we preset `\defaulttestrulewidth` to the already set value. Otherwise we default to a corps related value.

```
\def\defaulttestrulewidth{.2pt}
```

Beware, it's a macro not a ⟨*dimension*⟩.

```
\ifx\korpsgrootte\undefined
  \edef\defaulttestrulewidth{\the\testrulewidth}
\else
  \def\defaulttestrulewidth{.02\korpsgrootte}  % still dutch
\fi
```

```
\def\showmakeup%
  {\testrulewidth=\defaulttestrulewidth
   \showingcompositiontrue
   \showcomposition}
```

```
\protect
```

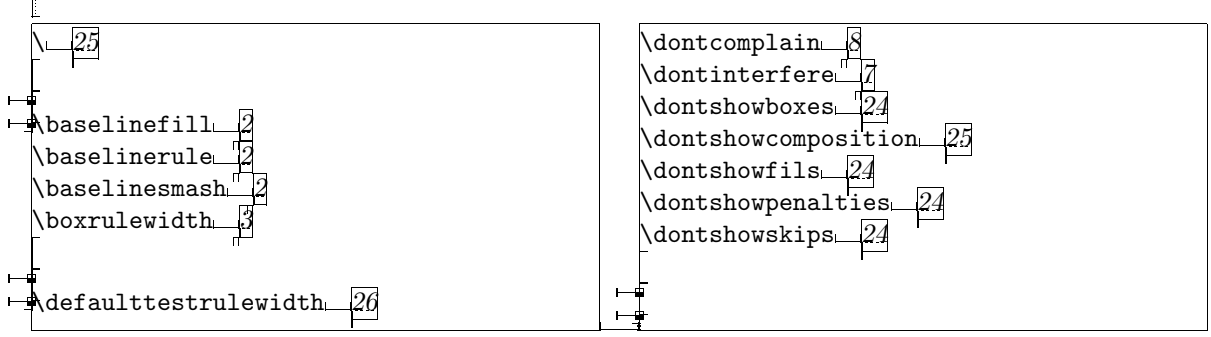
Lets end with some more advanced examples. Definitions and enumerations come in many flavors. The next one for instance is defined as:

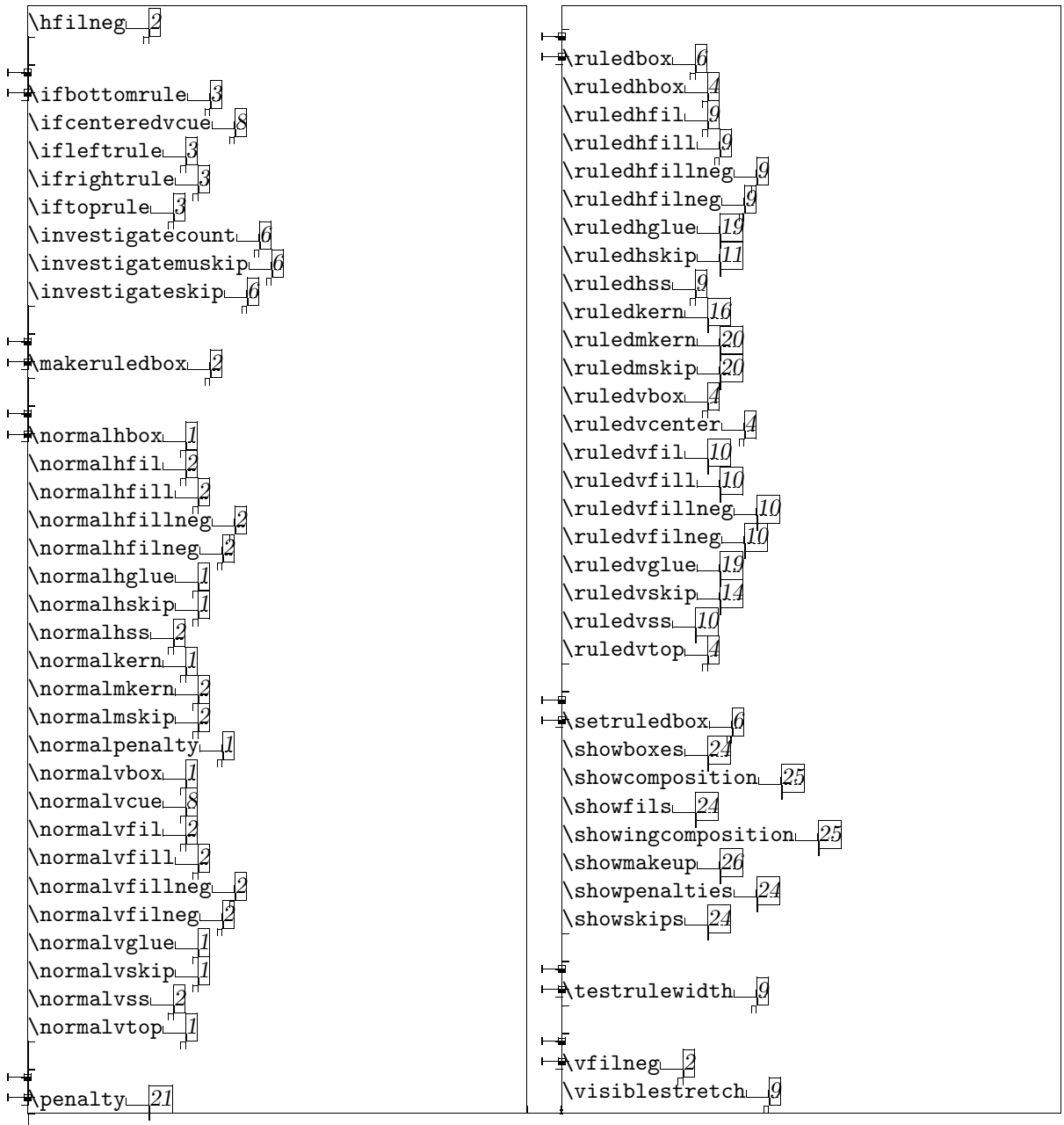
```
\definedescription[test][place=left,hang=3,width=6em]
```

When applied to some text, this would look like:

visual debugger I would be very pleased if TEX had two more primitives: `\vnop` and `\hnop`. Both should act and show up as normal boxes, but stay invisible for TEX when it's doing calculations. The `\vnop` for instance should not interact with the internal mechanism responsible for the disappearing skips, kerns and penalties at a pagebreak. As long as we don't have these two boxtypes, visual debugging will never be perfect.

The index to this section looks like:





Although not impressive examples or typesetting, both show us how and where things happen. When somehow the last lines in this two column index don't allign, then this is due to some still unknown interference.