Cheats (or not): When \prevdepth = -1000pt

Hans Hagen

There are numerous quantities that a user can set in TEX, for example \parskip and \parindent. These are internal registers, and being registers they can be manipulated with for instance \advance. The more curious \prevdepth and \prevgraf are not registers but are properties of the current list. They are set by the engine but users can also set them in order to control or even fool the machinery. Here we focus on \prevdepth.

The depth of a box is normally positive but rules can have a negative depth in order to get a rule above the baseline. When TEX was written the assumption was that a negative depth of more than 1000 points made no sense at all. The last depth on a vertical list is registered in the \prevdepth variable. This is essentially a reference into the current list.

In order to illustrate some interesting side effects of setting this \prevdepth, and especially when we set it to −1000pt, this special sentinel value can be changed in LuaMetaTEX. However, as dealing with the property is somewhat special in the engine, you should not set it unless you know that the macro package is aware of it.

```
line 1\par line 2\par\nointerlineskip line 3\par
```

Assuming that we haven't set any inter-paragraph spacing this gives:

line 1

line 2
line 3

Here \nointerlineskip is (normally) defined as:

```
\def\nointerlineskip{\prevdepth-1000pt }
```

TEX also internally sets \prevdepth to −1000pt at the beginning of a vertical list, or just after a rule, to automatically suppress the next interline glue at those places.

In LuaMetaTEX we made all these hard-coded numbers configurable and this −1000pt is one of them. One reason for this is that it makes it easier to explain some of the side effects. When I tried to explain this to a curious user it was no fun to show pages spanning thousands of points. The variable that we can set is named \ignoredepthcriterion; in LMTX, that variable is used when we need to check for the magical value of \prevdepth.

We are now ready to give a more extensive example (\ruledhbox is a ConTEXt command):

```
\ruledhbox \bgroup
    \PrevTest{-10.0pt}\quad
    \PrevTest{-20.0pt}\quad
    \PrevTest{-49.9pt}\quad
    \PrevTest{-50.0pt}\quad
    \PrevTest{-50.1pt}\quad
    \PrevTest{-60.0pt}\quad
    \PrevTest{-80.0pt}%
\egroup
```

The \PrevTest helper, to construct a box of a given depth (negative in our cases here), is defined as (first example):

```
\def\PrevTest#1%
  {\setbox0\ruledhbox{\strut$\tf#1$}%
   \dp0=#1
   \vbox\bgroup\hsize4em
     FIRST\par
     \unhbox0\par
     LAST\par
   \egroup}
```

or (second example)

```
\def\PrevTest#1%
  {\setbox0\ruledhbox{\strut$\tf#1$}%
   \dp0=#1
   \vbox\bgroup
     \ruledhbox{FIRST}\par
     \box0\par
     \ruledhbox{LAST}\par
   \egroup}
```

In this example we set \ignoredepthcriterion to −50.0pt instead of the normal −1000pt. The result is shown in figures 1 and 2. The first case is what we normally have in text; we haven't set prevdepth explicitly between lines, so TEX just looks at the depth of the lines. In the second case, where we typeset boxes instead of their contents, the depth is ignored when it is less than the criterion value which is why, when we set the depth of the box to a negative value, we get somewhat interesting skips.

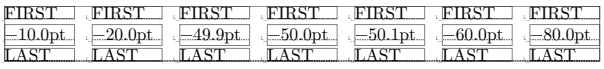

Figure 1: Showing explicitly-set depths of lines.

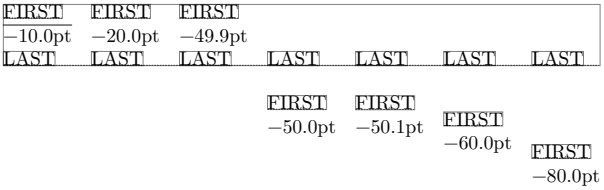

Figure 2: Depths above and below the magic \prevdepth value.

doi.org/10.47397/tb/44-2/tb137hagen-prevdepth

I'm sure one can use this effect in ways other than intended, but I doubt is any user is interested in doing so. Still, the fact that we can lower the criterion makes for nice experiments. For the record, in figure 3 you see what we get with positive values:

```
\ruledhbox \bgroup
    \PrevTest{10.0pt}\quad
    \PrevTest{20.0pt}\quad
    \PrevTest{49.9pt}\quad
    \PrevTest{50.0pt}\quad
    \PrevTest{50.1pt}\quad
    \PrevTest{60.0pt}\quad
    \PrevTest{80.0pt}%
\egroup
```

Watch the interline skip kick in when we make the depth larger than `\ignoredepthcriterion`, set here to (positive) 50pt. The (extremely) small additional space at 50.1pt is generated from `\lineskip`, while the full `\baselineskip` shows up at 60pt.

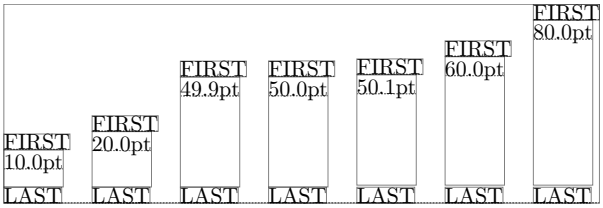

Figure 3: Positive depths.

So why don't we run into these side effects in regular documents? Simply because no one uses these excessive depths in box production, and also because hardly any user will work with such large paper sizes so for most users 1000pt (let alone −1000pt) is not something that they will naturally encounter.

This is true for many more mechanisms: sane usage is expected, so extreme cases can be ignored deep down in the engine. For instance, this is also why (due to old-times' performance reasons) wrap around of integers (and dimensions are just integers) is not that harmful and sometimes even useful, for instance when collecting content in boxes, where only when a user manipulates a related dimension some checking for overflow happens. Let's call them handy side effects.

Appendix: Implementation

For the record, here is the relevant snippet from original TEX:

```
@d ignore_depth==-65536000
       {|prev_depth| value that is ignored}
...
if prev_depth > ignore_depth then begin
  d := width(baseline_skip) - prev_depth
       - height(b);
  if d < line_skip_limit then
    p := new_param_glue(line_skip_code)
  else begin
    p := new_skip_param(baseline_skip_code);
    width(temp_ptr) := d;
  end;
  link(tail) := p;
  tail := p;
end;
link(tail) := b;
tail := b;
prev_depth := depth(b);
```

What we did was to make the `ignore_depth` constant into something like `int_par(ignore_depth_criterion_code)`.

If you look at the LuaMetaTEX code you will find something similar to the above, but there we split into functions. We also interface to a bit more granular paragraph (property) management, as well as adding callbacks.

◇ Hans Hagen
Pragma ADE