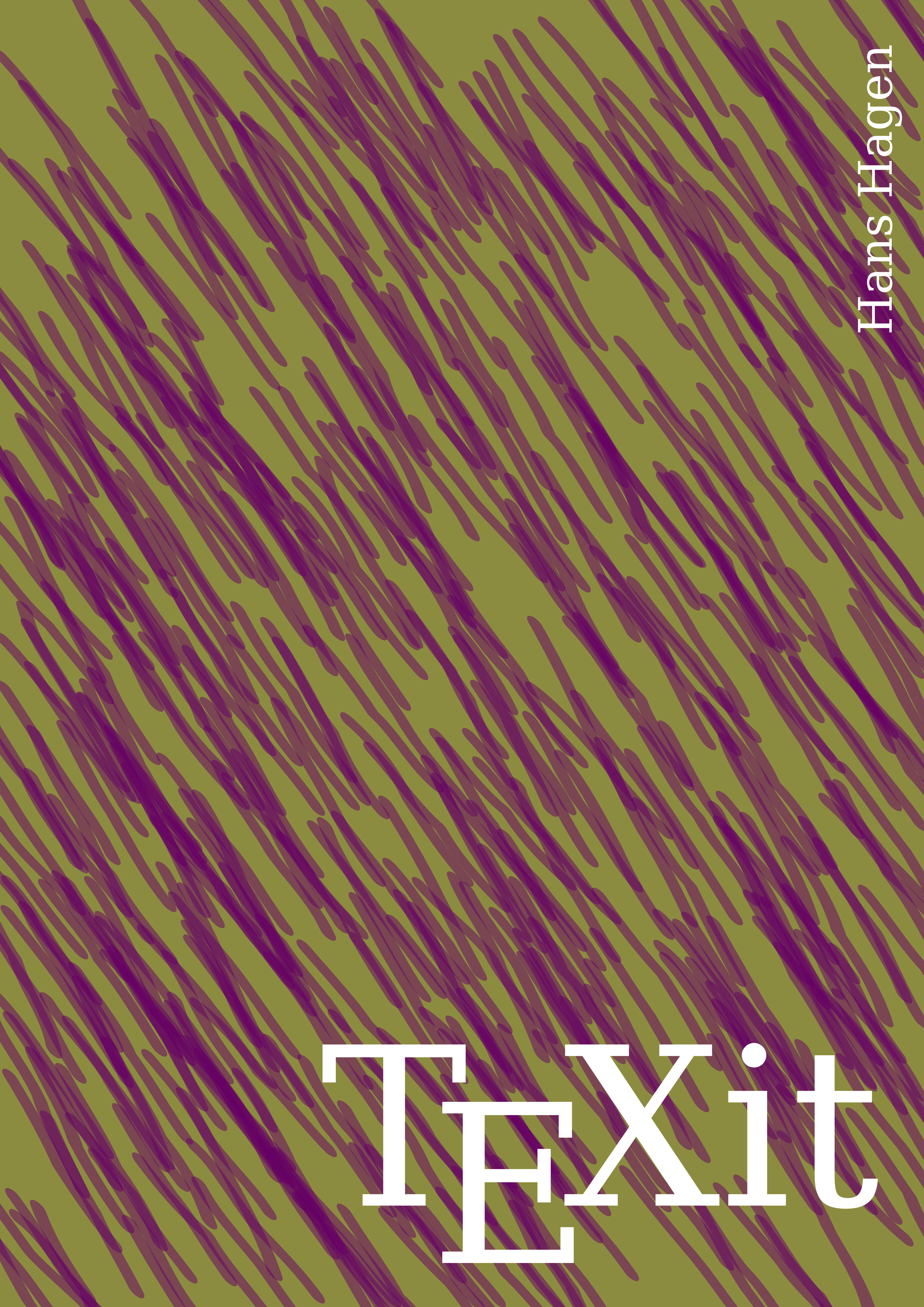


Hans Hagen

TeXit

Contents

Introduction

I needed a place to collect examples of TEX coding and this is it. The examples presented here are an unorganized bunch. Some originate in questions asked on the mailing list. Others are byproducts of tests made when playing with some (new) functionality. When you plan to use TEX for a long time, it doesn't hurt to see a bit of TEX coding but when possible I will also show the ConTEXt way.

I hope that this document is useful. You can of course always try to challenge me for more examples. Hopefully I will not forget about this document and extend it occasionaly.

Hans Hagen
Hasselt NL

1 Lookahead

When you look at the TeX source of a macro package, your can often see constructs like this:

```
\def\foo#1%
  {We do something with  "#1".}
```

or maybe:

```
\def\foo#1{%
    We do something with  "#1".%
}
```

Normally the percentage symbol is used to indicate a comment, but here are no comments. In these cases, it makes the definition effectively

```
\def\foo#1{do something with "#1"!}
```

which is different from when we would not have that percent sign there:

```
\def\foo#1 {We do something with "#1"!}
```

That variant is valid TeX code but expects a space as delimiter of the argument to `\foo`. This means that you can say:

```
\foo{1} \foo 2 \foo {34} and \foo 56 .
```

while this can trigger an error message (when no space is seen at some point) or at least give unexpected results.

```
\foo{1}\foo 2\foo {34}and\foo 56.
```

A different use of the percent is seen in cases like this:

```
\def\foo#1%
  {We do something %
   with "#1".}
```

This time we want to preserve the space after `something` because an end-of-line would either or not collapse it with `with` depending on how the endofline character is set up. Normally

```
\def\foo#1%
  {We do something
   with "#1".}
```

will also add a space after something but when TeX is set up to ignore lines you get a collapse. So the explicit space is a robust way out. Both cases of using or omitting the comment symbol are easy to spot as they trigger an error or result in weird typeset results.

```
\def\fooA#1%
  {\ifnum#1>100
     yes\else nop%
   \fi}
```

```
\def\fooB#1{\ifnum#1>100 yes\else nop \fi}
```

```
\def\fooC#1%
  {\ifnum#1>100%
     yes\else nop%
   \fi}
```

We test this with:

```
\fooA{100} \fooB{100} \fooC{100}
\fooA{101} \fooB{101} \fooC{101}
```

And the result is probably what you expect:

```
nop nop  nop
yes yes yes
```

However, when we have the following macro body:

```
\def\fooA#1%
  {\ifnum#1>100
     1\else 0%
   \fi}
```

```
\def\fooB#1{\ifnum#1>100 1\else 0\fi}
```

```
\def\fooC#1%
  {\ifnum#1>100%
     1\else 0%
   \fi}
```

We get this output. Do you see the issue?

```
0 0 0
1 1 0
```

A preferred way to catch this is the following as a `\relax` ends scanning for a number:

```
\def\foo#1%
  {\ifnum#1>100\relax
     1\else 0%
   \fi}
```

However, watch what happens here:

```
\edef\result{\foo{123}}
```

The `\result` macro has the following body:

```
macro:->\relax 1
```

A neat trick out of this is the following:

```
\def\foo#1%
  {\ifnum#1>\numexpr100\relax
     1\else 0%
   \fi}
```

Now the body of `\result` looks like this:

```
macro:->1
```

Of course this also works:

```
\def\foo#1%
  {\ifnum#1>100 %
     1\else 0%
   \fi}
```

as a space also delimits scanning the number. But that method can actually introduce that space in the output. Think of this definition:

```
\def\foo#1#2%
  {\ifnum#1>#2 %
     1\else 0%
   \fi}
```

What if #2 has a trailing space? What if it is a verbose number? What if it is a counter variable?

```
\scratchcounter=100
    [\foo{101}{100}] [\foo{101}{100 }] [\foo{101}\scratchcounter]
\scratchcounter=101
    [\foo{100}{101}] [\foo{100}{101 }] [\foo{100}\scratchcounter]
```

[1] [1] [1]
[0] [0] [0]

If you really want to introduce an unpredictable situation, use a coding style like this:

```
\def\foo#1#2#3#4{\if#1=#2#3\else#4\fi}
```

This is not that imaginary as you often see users play safe and do things like this:

```
\ifnum\scratchcounterone=\scratchcountertwo%
```

```
    ...
\else
    ...
\fi
```

Here the percent sign is useless as the number scanner already got the number, just try:

```
\scratchcounterone=1
\scratchcountertwo=1

\ifnum\scratchcounterone=\scratchcountertwo
    yes
\else
    nop
\fi
```

A previous one liner formatted like this really is not better!

```
\def\foo#1#2#3#4%
  {\ifnum#1=#2%
      #3%
   \else
      #4%
   \fi}
```

When you define macros more often than not you don’t want unexpected spaces (aka spurious spaces) which is why in ConTEXt for instance setups ignores lines:

```
\startsetups foo
    here
    we ignore
    spaces at the end
    of a line
\stopsetups
```

so we get: “herewe ignorespaces at the endof a line” which means that the normally few times that we *do* want spaces we need to be explicit:

```
\startsetups foo
    here\space
    we ignore\space
    spaces at the end\space
    of a line\space
\stopsetups
```

Now we’re okay: “here we ignore spaces at the end of a line”. The same is true for:

```
\starttexdefinition foo
```

```
    here\space
    we ignore\space
    spaces at the end\space
    of a line\space
\stoptexdefinition
```

There are more cases where TEX will look further. Take for example skip (glue) scanning. A glue specification can have `plus` and `minus` fields.

```
\scratchdimenone=10pt
\scratchskipone =10pt plus 10pt minus 10pt
\scratchskiptwo =0pt
```

Now take the following test:

```
{1 \scratchskiptwo  10pt             plus 10pt \relax\the\scratchskiptwo}
{2 \scratchskiptwo  \scratchdimenone plus 10pt \relax\the\scratchskiptwo}
{3 \scratchskiptwo 1\scratchdimenone plus 10pt \relax\the\scratchskiptwo}
{4 \scratchskiptwo  \scratchskipone  plus 10pt \relax\the\scratchskiptwo}
{5 \scratchskiptwo 1\scratchskipone  plus 10pt \relax\the\scratchskiptwo}
```

```
1 10.0pt plus 10.0pt
2 10.0pt plus 10.0pt
3 10.0pt plus 10.0pt
4 plus 10pt 10.0pt plus 10.0pt minus 10.0pt
5 10.0pt plus 10.0pt
```

If you wonder what the second `\relax` does, here is a variant:

```
1 0.0pt
2 0.0pt
3 0.0pt
4 plus 10pt 0.0pt
5 0.0pt
```

```
{1 \scratchskiptwo  10pt             plus 10pt \relax\the\scratchskiptwo}
{2 \scratchskiptwo  \scratchdimenone plus 10pt \relax\the\scratchskiptwo}
{3 \scratchskiptwo 1\scratchdimenone plus 10pt \relax\the\scratchskiptwo}
{4 \scratchskiptwo  \scratchskipone  plus 10pt \relax\the\scratchskiptwo}
{5 \scratchskiptwo 1\scratchskipone  plus 10pt \relax\the\scratchskiptwo}
```

```
1 10.0pt plus 10.0pt
2 10.0pt plus 10.0pt
3 10.0pt plus 10.0pt
4 plus 10pt 10.0pt plus 10.0pt minus 10.0pt
5 10.0pt plus 10.0pt
```

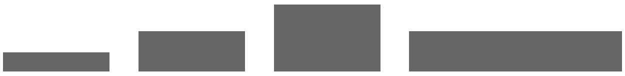
In this second variant TEX happily keep looking for a glue specification when it sees the `\the` so it serializes `\scratchskiptwo`. But as it sees `0pt` then, it stops scanning the glue spec.

What we get typeset is the old value, not the new one! If you want to prevent this you need to `\relax`.

Another case where TEX keeps scanning is the following:

```
\vrule width 40pt height 2pt depth 5pt \quad
\vrule width 40pt height 20pt depth 5pt height 10pt \quad
\vrule width 40pt height 10pt height 20pt \quad
\vrule width 40pt height 20pt depth 5pt height 10pt width 80pt
```

This gives the rules:



So you can overload dimensions. The space before the `quad` is gobbled as part of the look ahead for more keywords.

Often rules (just like glue assignments) are wrapped in macro definitions where the macro writer used `\relax` to look ahead. That way you prevent an error message in cases like:

```
\def\foo{\vrule width 40pt height 2pt}
```

```
The \foo depth of this thought is amazing.
```

because `of` definitely is not a valid dimension. Even more subtle is:

```
\def\foo{\hskip 10pt plus 1fil}
```

```
The \foo fine points of typesetting can actually become a nightmare.
```

As TEX will now see the `f` of `fine` as further specification and think that you want `1fill`.

So, the most important lesson of this chapter is that you need to be aware of the way TEX scans for quantities and specifications. In most cases the users can safely use a `\relax` to prevent a lookahead. And try to avoid adding percent signs all over the place.

2 Conditions

In case you wonder why we have modes in ConTEXt, here is an example that might convince you. The TEX language has conditionals and they are in fact quite efficient, take for instance:

```
\ifnum\scratchcounter>10
    \ifdim\scratchdimen>10pt
        one
    \else
        two
    \fi
\else
    three
\fi
```

When the first test fails, TEX will do a fast scan over the following tokens and expand the `three` branch. In order to do such a fast scan, the nested condition needs to be properly balanced: the `\else` is optional but the nested `\fi` definitely isn't. Now imagine that you use a few pseudo booleans, like:

```
\newif\ifalpha \alphatrue
\newif\ifbeta  \betatrue
```

And you need it in:

```
\ifalpha
    \ifbeta
        YES
    \else
        NOP
    \fi
\else
    NOP
\fi
```

This happens occasionally in real applications and one can either repeat the `NOP` or wrap it in a macro in order to save tokens. However, way more natural would be something like this:

```
\ifalphaorbeta
    YES
\else
    NOP
\fi
```

This basically would introduce a new kind concept: an expandable macro flagged as `\if` kind of token. I actually experimented with that in LuaTEX but rejected it eventually. Instead

`\ifcondition` was introduced. This is basically equivalent to `\iffalse` when TEX is in fast `\if*` skipping mode, but when a real test happens the next argument is expanded. That macro is expected to end up as something equivalent to `\iftrue` or `\iffalse` so that other the nexct branch or the `\else` is entered. Here is an example:

```
\ifcondition\alphaorbeta
    YES
\else
    NOP
\fi
```

There are several ways to define `\alphaorbeta` now and we show a few here. It's up to you to figure out which ons is the most efficient.

```
\def\alphaorbeta{\ifcase0\ifalpha \else\ifbeta \else1\fi\fi\relax}
\def\alphaorbeta{\ifcase \ifalpha0\else\ifbeta0\else1\fi\fi\relax}
\def\alphaorbeta{\ifnum1=\ifalpha1\else\ifbeta1\else0\fi\fi\relax}
\def\alphaorbeta{\ifnum 0\ifalpha1\fi  \ifbeta1\fi       >1\relax}
```

Now, do we expect users to come up with such constructs? Of course not. Even in ConTEXt we don't really need them, although there are a few places where they can be used. In ConTEXt you would just do this:

```
\enablemode[alpha]
\enablemode[beta]

\doifelsemode {alpha,beta} {
    YES
} {
    NOP
}
```

Of course such a verbose macro is less efficient but even if you use this test 10.000 times in a run it will not take more than 0.06 seconds on a decent 2013 laptop.

3 Leaders

The following example comes from a question on the ConTeXt list. It exhibits a few low level tricks. For the purpose of this example we use `\ruledhbox` instead of `\hbox`. We start with a simple command that puts something at the end of a line:

```
\starttexdefinition MyFill #1
    \removeunwantedspaces
    \hfill
    \ruledhbox{#1}
\stoptexdefinition
```

We use this in:

```
\startitemize[packed,joinedup][rightmargin=3em]
    \startitem
        \samplefile{ward}\MyFill{DW}
    \stopitem
\stopitemize
```

and get:

- The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. DW

But, the requirement was that we move the number towards the right margin, so instead we need something:

```
\starttexdefinition MyFill #1
    \removeunwantedspaces
    \hfill
    \rlap{\ruledhbox to \rightskip{\hss#1}}
\stoptexdefinition
```

This already looks more like it:

- The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. DW

But also part of the requirements was that there should be dots between the end of the last sentence and the number. In low level TeX speak that means using leaders: repeated boxed content where the repitition is driven by a glue specification. Let's naively use leaders now:

```
\starttexdefinition MyFill #1
    \leaders
        \ruledhbox to 1em{\hss.\hss}
        \hfill
    \ruledhbox{#1}
\stoptexdefinition
```

Let’s see what we get:

- The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. DW

Again we need to move the number to the right. This time we need a different solution because we need to fill the space in between. When TEX ends a paragraph it adds `\parfillskip` so we will now manipulate that parameter.

```
\starttexdefinition MyFill #1
    \parfillskip-1\rightskip plus 1fil\relax
    \leaders
        \ruledhbox to 1em{\hss.\hss}
        \hfill
    \ruledhbox{#1}
\stoptexdefinition
```

Does it look better?

- The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. DW

Indeed it does, but watch this:

```
\startitemize[packed,joinedup][rightmargin=8.5em]
    \startitem
        \samplefile{ward}\MyFill{DW}\par
        \samplefile{ward}\par
        \samplefile{ward}\MyFill{DW}
    \stopitem
\stopitemize
```

The first `\MyFill` will set the `\parfillskip` to a value that will also be used later on.

- The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient

who smokes many packs of cigarettes per day—and we humans are the cigarettes. .... DW
The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes.
The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. .... DW

The way out is the following

```
\starttexdefinition MyFill #1
    \begingroup
    \parfillskip-1\rightskip plus 1fil\relax
    \leaders
        \ruledhbox to 1em{\hss.\hss}
        \hfill
    \ruledhbox{#1}
    \par
    \endgroup
\stoptexdefinition
```

This looks more or less okay. The `\par` keeps the adaption local but for it to work well, the `\par` must be inside the group.

- The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. .... DW
  The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes.
  The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes. .... DW

Now it’s time to go for perfection! First of all, we get rid of any leading spacing. If we need some we should inject it after a cleanup. We also use a different leader command. Instead of `to` we use a `spread` so that we get half the emwidth and not something slightly less due to the width of the period.

```
\starttexdefinition MyFill #1
    \removeunwantedspaces
    \begingroup
    \parfillskip-1\rightskip plus 1fil\relax
    \cleaders
        \ruledhbox spread 1em{\hss.\hss}
        \hfill
    \ruledhbox{#1}
    \par
    \endgroup
\stoptexdefinition
```

So, we end up here:

- Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world. . . . . RS

For which we used this:

```
\startitemize[packed,joinedup][rightmargin=5em]
    \startitem
        \samplefile{sapolsky}\MyFill{RS}\par
    \stopitem
\stopitemize
```

Finally we get rid of the tracing:

```
\starttexdefinition unexpanded MyFill #1
    \begingroup
    \parfillskip-1\rightskip plus 1fil\relax
    \leaders
        \hbox to \emwidth{\hss.\hss}
        \hfill
    \hbox{#1}
    \par
    \endgroup
\stoptexdefinition
```

Watch a few more details. It brings us to:

- Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world. . . . . RS

```
\definefiller
  [MyFiller]
  [offset=.25\emwidth,
   method=middle]

\starttexdefinition unexpanded MyFill #1
    \begingroup
    \parfillskip-1\rightskip plus 1fil\relax
    \filler[MyFiller]%
    \hbox{#1}
    \par
    \endgroup
\stoptexdefinition
```

- Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world. . . . . RS

When writing these examples I realized that it's rather trivial to add this option to the already existing filler mechanism. The definition of such a filler looks like this:

```
\definefiller
  [MyFiller]
  [offset=.25\emwidth,
   rightmargindistance=-\rightskip,
   method=middle]
```

The sample code now becomes:

```
\startitemize[packed,joinedup][rightmargin=5em]
    \startitem
        \samplefile{sapolsky}\fillupto[MyFiller]{RS}
    \stopitem
\stopitemize
```

Ans as expected renders as:

- Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world. . . . . RS