The \texttt{ltpara.dtx} code*

Frank Mittelbach

June 3, 2021

Abstract

This code defines four special kernel hooks to support paragraph tagging as well as four public hooks which can be occasionally useful.

1 Introduction

The building of paragraphs in the \TeX engine(s) has a number of peculiarities that makes it on one hand fairly flexible but on the other hand somewhat awkward to control or reliably to extend. Thus to better understand the code below we start with a brief introduction of the mechanism; for more details refer to the \TeXbook [? chap. 14] (for the full truth you may even have to study the program code).

1.1 The default processing done by the engine

\TeX automatically starts building a paragraph when it is currently in vertical mode and encounters anything that can only live in horizontal mode. Most often this is a character, but there are also many commands that can be used only in horizontal mode. If any of them is encountered, \TeX will immediately back up (i.e., the character or command is read later again), adds a $\backslash$\texttt{parskip} glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special “start of paragraph processing” and only then rereads the character or command that caused the mode change.$^1$

This “start of paragraph processing” first adds an empty box at the start of the horizontal list of width $\backslash$\texttt{parindent} (which represents the paragraph indentation) unless the paragraph was started with $\backslash$\texttt{noindent} in which case no such box is added$^2$. It then reads and processes all tokens stored in the special engine token register $\backslash$\texttt{everypar}. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, \TeX offers the possibility to put some special code into $\backslash$\texttt{everypar} to gain control at (more or less) the start of the paragraph. For example, in LaTeX and a number of packages, special code like the following is sometimes used:

\begin{verbatim}
$\backslash$\texttt{everypar}{$\{\texttt{setbox}\z@\texttt{lastbox}\}$\texttt{everypar}{}} ...
\end{verbatim}

\footnote{This file has version v1.0g dated 2021/05/27, \copyright LA\TeX Project.}

$^1$Already not quite true: the command $\backslash$\texttt{noindent} starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

$^2$That’s a bit different from placing a zero-sized box!
This removes the paragraph indentation box again (that was already placed by \TeX), then resets \everypar so that it doesn’t do anything on the next paragraph start and then does whatever it wants to do, e.g., in an \item of a list it will typeset the label in front of the paragraph text. However, there is only one such \everypar token register and if different packages and/or the kernel all attempt to add their own code here, coordination is very difficult if not impossible.

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive \par is called while \TeX is in unrestricted horizontal mode, i.e., is building a paragraph. At other times this primitive does nothing or generates as an error depending on the mode \TeX is in, e.g., the \par in \hbox{a\par b} is ignored, but \$a\par b\$ would complain.

If this primitive ends the paragraph it does some special “end of horizontal list” processing, then calls \TeX paragraph builder that breaks the horizontal list into lines then these lines are added as boxes to the enclosing vertical list and \TeX returns to vertical mode.

This \par command can be given explicitly, but there are also situations in which \TeX is generating it on the fly. Most often this happens when \TeX encounters a blank line which is automatically changed to a \par command which is then executed. The other possibility is that \TeX encounters a command which is incompatible with horizontal processing, e.g., \vskip (a request for adding vertical space). In such case it silently backs up, and inserts a \par in the hope that this gets it out of horizontal mode and makes the offending command acceptable.

The important point to note here is that \TeX really inserts the command \par which can be redefined. Thus, it may not have its original “primitive” meaning and therefore may not end the horizontal list and call the paragraph builder. This approach offers some flexibility but also allows you to easily produce a \TeX document that loops forever, for example, the simple line

```
A \let\par\relax \vskip
```

will start a horizontal list at A, redefines \par, then sees \vskip and inserts \par to end the paragraph. But this now only runs \relax so nothing changes and \vskip is read again, issues a \par which \ldots In short, it takes a plain \TeX document with five tokens to run forever (as not even memory is consumed and therefore eventually exhausted).

There are no other ways than changing \par to gain control at the end of a paragraph, i.e., there is no token list like \everypar that is inserted, i.e., the only way to change the default behavior is to modify the action that \par executes with similar issues as outlined before: different processes need to ensure that they do not overwrite their modifications or worse, think that the \par in front of them is the engine primitive while in fact it has already been changed by other code.

To make matters slightly worse there are a few places where \TeX handles the situation differently (most likely for speed reasons back when computers were much slower). If \TeX finds itself in unrestricted horizontal mode at the end of building a vertical box (or an \insert, \vadjust or at the end of executing the output routine code), it will finish the horizontal list not by issuing a \par command (which would be consistent with all other places, but by simply executing the primitive version of \par regardless of the definition that \par has at the time.

Thus, if you have carefully crafted a redefined \par to execute some special actions at the end of a paragraph and you write something like

```
\vbox{Some paragraph \ldots text.}
```
you will find that your code has never run for the last paragraph in that box. \LaTeX avoids this problem, by making sure that all its boxes (such as \texttt{parbox} or the \texttt{minipage} environment, etc.) all internally add an explicit \texttt{par} at the end so that such code is run and \TeX finds itself in vertical mode already without the need to start up the paragraph builder internally. But, of course, this only works for boxes under direct control of the \TeX kernel, if some package uses low-level \texttt{vbox}es without adding this precaution the \TeX optimization kicks in and no special \texttt{par} code is executed.

And there is another optimization that is painful: if a paragraph is interrupted by a mathematical display, e.g., \[...\] in \LaTeX or $$...$$ in plain \TeX, then \TeX will resume horizontal mode afterward, i.e., build a new horizontal list (without inserting an indentation box or \texttt{everypar} at that point). However, if that list immediately ends with an explicit or implicit \texttt{par} then \TeX will simply throw away this “null” paragraph and not do its usual “end of horizontal list” processing, so this special case need to be accounted for when introducing some extended processing.

## 2 The new mechanism implemented for \LaTeX

To improve the situation (and also to support automatic tagging of PDF documents) we now offer public as well as private hooks at the start and end of the paragraph processing. The public hooks can be used by packages (or by the user in the preamble or within the document) and using the hook mechanisms it is possible to reorder or arrange code from different packages in a way that it can safely coexist.

To make that happen we have to make use of the basic functionality that is offered by \TeX, e.g., we install special code inside \texttt{everypar} to provide hooks at the beginning and we redefine \texttt{par} to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of \texttt{everypar} is not overwriting our code. This is done through a trick: we basically hide the real \texttt{everypar} from the packages and offer them a new token register (with the same name). So if they install their own code it doesn’t overwrite ours. Our code then inserts the new \texttt{everypar} at the right place inside the process so that it looks as if it was the primitive \texttt{everypar}.

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that \TeX inserts the token \texttt{par} (that doesn’t have a defined meaning) we can’t hide “the real thing” and offer the package an indistinguishable alternate.

Fortunately, \LaTeX has already redefined \texttt{par} for its own purposes. As a result there aren’t many packages that attempt to change \texttt{par}, because without a lot of extra care that would fail miserably. But bottom line, if you load a package that alters \texttt{par} then the end of paragraph hooks are most likely not executing while that redefinition is active.\footnote{Ideally, \texttt{everypar} wouldn’t be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longer term goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in \texttt{everypar}), this will not happen overnight. For that reason support for this legacy method is retained.}

\footnote{Similarly to the \texttt{everypar} situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.}
2.1 The provided hooks

<table>
<thead>
<tr>
<th>钩名</th>
<th>描述</th>
</tr>
</thead>
<tbody>
<tr>
<td>para/before</td>
<td>该钩在内核钩@kernel@before@para@before（下面讨论）后，在垂直模式下执行，当LaTeX贡献\parskip到垂直列表后，且在水平模式开始前。该钩应该不产生任何排版材料或只增加垂直材料。如果它开始了一个段落，会生成错误。原因是我们正在开始处理一个段落，并且这样会导致无尽的递归。</td>
</tr>
</tbody>
</table>
|para/begin | 该钩在内核钩\@kernel@before@para@begin（下面讨论）后，在水平模式下执行，当没有\noindent时，即段落没有以任何方式开始，插入缩进盒。如果需要阻止自动放置（在钩执行后），可以使用\OmitIndent。更精确地说，\OmitIndent会清除缩进盒。

该钩通过某种等效于将\box\IndentBox跟在任何可用的内核或包中（执行一些遗留代码）。

该钩应该小心不要在其内嵌段落（例如，通过添加一个\parbox或一个marginpar）中添加任何代码，因为它会导致段落的递归，直到耗尽可用内存。这只能通过确保不执行该钩的内嵌段落（或至少不是永远递归）来实现。

|para/end   | 该钩在段落结束时，当LaTeX返回到垂直模式后，以及任何从水平列表迁移到垂直列表（例如，从一个\vadjust）的材料处理后执行。该钩可以用于添加进一步的水平材料，但不应改变模式（甚至暂时退出水平模式会搞出混乱—任何尝试都会导致错误消息）！该钩在段落结束时，当LaTeX返回到垂直模式后。

该钩作为公共钩提供，但由于需要保持在水平模式内，一个新的段落需要小心，因为什么被放入了这个钩。

|para/after | 该钩直接在LaTeX返回到水平模式后，以及任何从水平列表中迁移到垂直列表的材料（例如，从一个\vadjust）处理后执行。该钩作为反向钩提供。

---

5One could allow it but only if the newly started paragraph is processed without any hooks. Furthermore correct spacing would be a bit of a nightmare so for now this is forbidden.

6Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.
This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn’t prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook \@kernel@after@para@after is executed as the final action of the paragraph processing.

\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

\@kernel@before@para@before For future extensions, not currently used by the kernel.
\@kernel@after@para@after For future extensions, not currently used by the kernel.
\@kernel@before@para@begin Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after \TeX has switched to horizontal mode but before any indentation box got added or any \everypar was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn’t be touched this isn’t checked.

\@kernel@after@para@end Used by the kernel to implement tagging. It is executed directly after the public \para/end hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

2.2 Altered and newly provided commands

\par \endgraf \para_end:

An explicit request for ending a paragraph is known in plain \TeX under the name \endgraf where it simply calls the paragraph primitive (regardless of what \par may have as its current definition). In \LaTeX \endgraf with that behavior was also made available.

With the new paragraph handling in \LaTeX, ending a paragraph means a bit more than just calling the engine’s paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus \endgraf was changed to provide this additional functionality (and so by extension \par subject to its current meaning).

The expl3 name for the functionality is \para_end:

Note: The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!
Inside the `\begin{para}` hook one can use this command to suppress the indentation box at the start of the paragraph. (Technically it is possible to use this command outside the hook as well, but this should not be relied upon.) The box itself remains available for use.

The `\Expl3` name for the function is `\para omit indent:`.

The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. It remains available even if the `\OmitIndent` command was used; in that case it will just not be automatically placed.

The `\Expl3` name for the box register is `\g para indent_box`.

The commands `\RawIndent` and `\RawNoindent` are not meant for normal paragraph building (where the result is a textual paragraph in the traditional meaning of the word), but for special cases where `\TeX`'s low-level algorithm is used to achieve special effects, but where the result is not a “paragraph”.

They are called “raw”, because they bypass `\TeX`’s hook mechanism for paragraphs and simply invoke the low-level `\TeX` algorithm. I.e., they are like the original `\TeX` primitives `\indent` and `\noindent` (that is they execute no hooks other than `\everypar`) except that they can only be used in vertical mode and generate an error if found elsewhere.

To avoid issues a paragraph started by them should always be ended by `\RawParEnd` and not by `\par` (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray `\pars`.

The `\Expl3` names for the functions are `\para raw indent:`, `\para raw indent:` and `\para raw end:`.

### 2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

#### 2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

```
PARA: 1-i start (document env. on input line 38)
PARA: 1-i end (document env. on input line 38)
```

---

7Technical note for those who know their `\TeX`book: the `\RawParEnd` command invokes the original `\TeX` engine definition of `\par` that (solely) triggers the paragraph builder in `\TeX` when found inside unrestricted horizontal mode and does nothing in other processing modes.
As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that \LaTeX{} considers more things “a paragraph” than you have probably thought.

This was generated by the following hook code:

\begin{verbatim}
\newcounter{paracnt} % sequence counter
\newcounter{paralevel} % level counter

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using \texttt{expl3} functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of \LaTeX{}\raisebox{1ex}{2}ε thrown in as well. When popping the result gets stored in \texttt{\paracntvalue} and the \texttt{\ERROR} should never happen because it means we have tried to pop from an empty stack.

\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush {\seq_gpush:No \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop {\seq_gpop:NNF \g_para_seq \paracntvalue \ERROR}
\ExplSyntaxOff

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

\AddToHook{para/begin}{%
  \stepcounter{paracnt}\stepcounter{paralevel}%
  \ParaPush
}

To display the sequence number we \texttt{\typeout} the current sequence and level number. The command \texttt{\@currenvir} gives us the current environment and \texttt{\on@line} produces a space and the current input line number.

\AddToHook{para/end}{%
  \ParaPop
  \typeout{PARA: \paracntvalue-\roman{paralevel} end \space \space
          (\@currenvir space env. \on@line)}%
}

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in \texttt{\paracntvalue} the way we have set this up above.

\AddToHook{para/end}{%
  \ParaPop
  \typeout{PARA: \paracntvalue-\roman{paralevel} end \space \space
          (\@currenvir space env. \on@line)}%
\end{verbatim}
We also typeset again a tiny red number with that value, this time sticking out to the right.\footnote{Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn’t take up any space, because it paragraph is no longer empty and thus isn’t dropped!} We also decrement the level counter since our level has finished.

\rlap{\color{red}\tiny \paracntvalue}\
\addtocounter{paralevel}{-1}\
\makeatother

\section*{2.3.2 Mark the first paragraph of each \texttt{itemize}}

The code for this is rather simple. We apply hook code that is executed only once inside a hook that is executed at the begin of each \texttt{itemize}. We explicitly change the color back and forth so that we don’t introduce grouping around the paragraph.

\AddToHook{env/itemize/begin}{%\
  \AddToHookNext{para/begin}{\color{blue}%\n    \AddToHookNext{para/end}{\color{black}}%\n  }%\n}\makeatother

As a result the first paragraph of each \texttt{itemize} will appear in blue.

\section*{2.4 Some technical notes}

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

\subsection*{2.4.1 Glue items between paragraphs (found with \texttt{fancypar})}

In the past \LaTeX{} placed two glue items between two consecutive paragraph, e.g.,

\begin{verbatim}
  text1 \par text2 \par
\end{verbatim}

would show something like

\begin{verbatim}
  \glue(\parskip) 0.0 plus 1.0
  \glue(\baselineskip) 5.16669
\end{verbatim}

but now there is another \texttt{parskip} glue (that is always 0pt):

\begin{verbatim}
  \glue(\parskip) 0.0 plus 1.0
  \glue(\parskip) 0.0
  \glue(\baselineskip) 5.16669
\end{verbatim}

The reason is that we generate a “fake” paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn’t contribute anything vertically but if somebody writes code the unravels a constructed list using \texttt{\lastbox}, \texttt{\unskip} and \texttt{\unpenalty} then the code has to remove one additional glue item or else will fail.
Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

<table>
<thead>
<tr>
<th>B</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>\box</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>5</td>
</tr>
<tr>
<td>\endgraf</td>
<td></td>
</tr>
<tr>
<td>\ERROR</td>
<td>7</td>
</tr>
<tr>
<td>\everypar</td>
<td>1–6</td>
</tr>
<tr>
<td>I</td>
<td>6</td>
</tr>
<tr>
<td>\indent</td>
<td></td>
</tr>
<tr>
<td>\IndentBox</td>
<td>4, 6</td>
</tr>
<tr>
<td>\insert</td>
<td>2</td>
</tr>
<tr>
<td>\item</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>8</td>
</tr>
<tr>
<td>\lastbox</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>\marginpar</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>1, 4, 6</td>
</tr>
<tr>
<td>O</td>
<td>4, 6</td>
</tr>
<tr>
<td>\OmitIndent</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>2, 3, 5, 6</td>
</tr>
<tr>
<td>\par commands:</td>
<td></td>
</tr>
<tr>
<td>\para_end</td>
<td>5</td>
</tr>
<tr>
<td>\g_para_indent_box</td>
<td>6</td>
</tr>
<tr>
<td>\paraomitindent</td>
<td>6</td>
</tr>
<tr>
<td>\pararaw_end</td>
<td>6</td>
</tr>
<tr>
<td>\pararawindent</td>
<td>6</td>
</tr>
</tbody>
</table>

\pararaw_noindent: 6
para/after: 4
para/before: 4
para/begin: 4
para/end: 4
\paracntvalue: 7
\parabox: 3, 4
\parindent: 1
\parskip: 1, 4, 8
\par

\RawIndent: 6
\RawNoindent: 6
\RawParEnd: 6
\relax: 2
\relax: 2
\relax: 2

\TeX and \LaTeX\ε commands:
\@currenvir: 7
\@kernel@after@para@after: 5
\@kernel@after@para@end: 4, 5
\@kernel@before@para@before: 4, 5
\@kernel@before@para@begin: 4, 5
\on@line: 7
\typeout: 7
\typeout: 7
\unpenalty: 8
\unskip: 8
\unpenalty: 8
\unskip: 8
\vadjust: 2
\vbox: 3
\vskip: 2