\TeX{} from \textquote{indent to \par}\footnote{Published in TUGboat 14.1, 60–62 (1993).}

Marek Ryćko \footnote{Wydawnictwo Do, ul. Filtrowa 1, 00-611 Warszawa, Poland} and Bogusław Jackowski \footnote{ul. Tatrzańska 6/1, 80-331 Gdańsk, Poland}

1 Introduction

There are some seemingly simple questions about T\TeX{} which may be difficult to answer without precise knowledge of T\TeX{} mechanisms.

In Section 2 we will ask three such questions, encouraging the reader to answer them without reading the explanation.

Actually, the explanation follows immediately from a detailed specification of T\TeX{}’s action at the beginning and at the end of a paragraph. We believe that if such a specification of all T\TeX{}’s mechanisms existed, answers to most questions concerning behavior of T\TeX{} would be equally simple.

The pivot sections are 5 and 7. Section 3 contains necessary introductory material.

2 Questions

In all questions we assume the normal meaning of tokens of plain T\TeX{}.

Q1. What is the difference between:
\begin{verbatim}
(\*) \everypar{\def\indent{1}}
  \indent 3 is a prime number.
\end{verbatim}
and
\begin{verbatim}
(\**) \everypar{\def\vrule{1}}
  \vrule 3 is a prime number.
\end{verbatim}
What is typeset in both cases and why?

Q2. Assuming that T\TeX{} is in vertical mode, what is the difference between:
\begin{verbatim}
(\*) \parindent=0mm \indent \par
\end{verbatim}
and
\begin{verbatim}
(\**) \noindent \par
\end{verbatim}
What is appended to the main vertical list and why?

Q3. What is the difference between:
\begin{verbatim}
(\*) \par
\end{verbatim}
and
\begin{verbatim}
(\**) {\par}
\end{verbatim}
What is the state of T\TeX{} after executing these commands in both cases and why?

3 From input characters to commands

Let us start with a closer look into T\TeX{}’s way of processing of input data. Three levels of the processing can be distinguished:

L1. Reading characters from the input file and transforming them into tokens (lexical analysis).

L2. Expanding tokens.

L3. Executing commands; at this level T\TeX{} creates internal lists (horizontal, vertical and math lists), transforms them into boxes and writes some boxes to the DVI file (using the \texttt{\shipout} command).

Knuth says about ‘eyes,’ ‘mouth’ and ‘stomach’ of T\TeX{}, etc.; we prefer to speak about ‘levels.’

3.1 Names and meanings of tokens

In order to understand what happens at the beginning and at the end of a paragraph it is essential to be aware of the difference between names and meanings of tokens.

Following Knuth, we will denote by $*\text{\texttt{xyz}}$ the meaning of the command $\texttt{\xyz}$ at the beginning of the T\TeX{} job. By $\texttt{xyz}$ we will denote a token, the name of which consists of the letters ‘xyz’. Such a token is created by T\TeX{} from the sequence of letters ‘xyz’ preceded by a current escape character, usually backslash.

For example, the token $\texttt{\hbox}$, the name of which consists of the letters ‘hbox,’ has initially the meaning $*\texttt{\hbox}$. Saying $\texttt{\let\hbox=\par}$ a user may change the meaning of $\texttt{\hbox}$ to the current meaning of $\texttt{\par}$, most likely to $*\texttt{\par}$. Incidentally, T\TeX{} replaces every empty input line with the token $\texttt{\par}$ regardless of the meaning of this token. The meaning of $\texttt{\par}$ may be $*\texttt{\par}$, but $\texttt{\par}$ may be also, for example, a macro expanding to a sequence of tokens.

3.2 Transforming input characters into tokens

From the point of view of T\TeX{}, the input file is a sequence of characters organized into lines. T\TeX{} reads such characters one by one and transforms them at the level 1 into so-called tokens, according to definite rules. For example, the following sequence of 15 input characters:
is unexpandable, hence no further expansion takes place.

Each token created at this level is associated with its current meaning which can be either a primitive meaning (a meaning that is built in into \TeX) or it can be a macro (a meaning that can be defined by a user in terms of other meanings). Regarding the meaning we can classify all tokens as follows:

(a) with respect to expandability as expandable and unexpandable;
(b) with respect to primitivity as primitive and macros.

The expandable tokens can be primitive, like \if, \the, \noexpand, \csname, or they can be macros defined using \def or a related assignment (\edef, \gdef, \xdef).

All unexpandable tokens are primitive. This group contains, among others: tokens like \hskip, \hbox, etc.; letters and other characters; all tokens defined by the \chardef assignment; some tokens defined by \let or \futurelet.

### 3.3 Expanding tokens

The level 2 of \TeX, i.e., the expansion level, reads tokens from the input token list and expands them. If the first token in the input token list is expandable, the level 2 of \TeX expands it, that is, replaces this token (possibly with some tokens following it) with another sequence of tokens.

If—after the replacement—the first token is still expandable, the expansion is repeated until the list starts from an unexpandable token. Obviously, this process may loop infinitely.

For example, the result of expansion of the first token in the input token list:

```
\enspace Don...
```

is the sequence of tokens:

```
\kern .5em \enspace Don...
```

because the first token \enspace is expandable (it is a plain \TeX macro) and its expansion is \kern .5em. The token \kern is unexpandable, hence no further expansion takes place.

The input token list with an unexpandable token at the beginning is submitted to the level 3 of \TeX.

### 3.4 Commands

By a command we mean an unexpandable (primitive) token at the beginning of the input token list. If a command may or must have arguments, only the first token is a command. For example, in the input token list:

```
\kern .5em \enspace Don...
```

the token \kern is the command and the tokens \enspace Don... are arguments. They are being read as a part of the process of executing the command.

In general, a command can read arguments from an input list either demanding expansion from the level 2 or not.

The level 3 of \TeX—the level that executes commands—is the central level. Every time this level is about to execute the next command it ‘asks’ the level 2 to prepare the input token list such that at the beginning of the list there is a primitive (unexpandable) token. In turn, the level 2 ‘asks’ the level 1 for preparing necessary tokens.

The level 3 executes the command according to its meaning, taking into account the current internal state of \TeX, including the values of various parameters, and, in particular, taking into account current \TeX’s mode.

One of the results of executing commands is creation of various kinds of internal lists. The types of lists include: horizontal, vertical and math lists.

At every moment \TeX is in one of the following six modes determining what type of list it is currently constructing:

(a) vertical mode (v-mode)
(b) internal vertical mode (iv-mode)
(c) horizontal mode (h-mode)
(d) restricted horizontal mode (rh-mode)
(e) math mode
(f) display math mode

At the very beginning of a job \TeX is in v-mode and all the lists are empty. A list is constructed by appending new elements to it. The process of list construction can be briefly summarized as follows: mathematical lists are converted into h-lists; an h-list created in h-mode (a material for a paragraph) is converted into a v-list and appended to a current v-list; a vertical list created in v-mode is converted to boxes by a page builder; eventually, boxes to which a command \shipout is applied are written to a DVI file.

### 4 Summary of paragraph construction

In the process of creating a paragraph by \TeX there are three distinct phases:

**P1.** Switching from v-mode to h-mode (opening a new h-list—see Section 5).

**P2.** Creating the h-list. (We do not discuss this phase in the paper. The notion of h-list is explained in ‘The \TeXbook,’ pp. 94–95. The systematic description of how the commands processed in h-mode influence the state of the h-list contain chapters 24 and 25 of ‘The \TeXbook,’ pp. 267–287).
P3. Switching from h-mode to v-mode (converting the h-list into a v-list and appending this vertical list to the main v-list; this is discussed in Section 7).

We will focus our attention on the moment of switching from v-mode or iv-mode to h-mode and back again. For the sake of simplicity we confine ourselves to the case where display math is not used inside a paragraph.

5 Switching from vertical to horizontal mode

In this section we describe when and how the level 3 of \TeX accomplishes the change of modes from v-mode or iv-mode to h-mode.

First we say 'when', i.e., we list the commands that—if executed in one of v-modes—switch \TeX's state to h-mode.

Then we say 'how', that is, we list the actions that \TeX performs during the mode change.

5.1 Switching from vertical to horizontal mode: when

Some commands will be called here vh-switches, because if encountered in v-mode or in iv-mode they switch \TeX to h-mode. They can be classified into two groups:

(a) explicite vh-switches:
- *\indent;
- *\noindent;
(b) implicit vh-switches (called by Knuth horizontal commands):
- letter: any character token of category 11 (also implicit; for example, control sequence \textbackslash d after executing the assignment `\let\d=A'; the assignment associates the token \textbackslash d with a meaning that is primitive in \TeX);
- other character: any character token of category 12 (also implicit; for example, control sequence \textbackslash one after executing the assignment `\let\one=1');
- \char;
- a `chardef' token, i.e., a control sequence or an active character which has been assigned a meaning by the command \textbackslash chardef (for example, control sequence \textbackslash a after the assignment `\chardef\ae=*\AE'; once again, the assignment associates the token \textbackslash a with a meaning that is primitive in \TeX);
- *\noboundary (a new primitive that appeared in \TeX 3.0);
- *\unhbox, *\unhcopy (independently of the contents of the box being an argument);
- *\valign;
- *\vrule;
- *\hskip;
- *\hfil, *\hfill, *\hss, *\hilneg (these tokens are primitive, not macros, even though the effects they cause could be achieved using *\hskip with appropriate parameters);
- *\accent;
- *\discretionary, *\-;
- *\ (control space *\ is a primitive command and if used in v-mode switches the mode to horizontal; note that normal space \textbackslash , in general any space token, is ignored in v-mode);
- $(also the first$ of the pair $starting the displayed math formula).

It should be stressed that commands *\hbox, *\vbox and *\vtop are not switches. Such commands encountered in v-mode do not change the mode. The box (preceded by proper glue) is appended to the current v-list.

5.2 Switching from vertical to horizontal mode: how

Assume that \TeX is in either v-mode or iv-mode. When the level 3 encounters a vh-switch at the beginning of the input token list it performs in turn the following actions:

(a) Optionally, a vertical glue \textbackslash parskip is appended to the vertical list:
- if \TeX is in iv-mode and the list is empty, the glue is not appended,
- if \TeX is in iv-mode and the list is not empty, the glue is appended,
- if \TeX is in v-mode the glue is always appended to the part called 'recent contributions' of the main v-list.

(b) If \TeX is in v-mode (not iv-mode) the page builder is exercised, that is \TeX runs the algorithm that moves elements of the v-list from the part of 'recent contributions' to the part 'current page'. In particular it may cause page breaking (running the \output routine).

(c) Switching from v-mode or iv-mode to h-mode occurs.

(d) Variables \textbackslash spacefactor and \textbackslash prevgraf are assigned values 1000 and 0, respectively (these assignments are called by Knuth 'global intimate assignments' and work in a rather peculiar way).

(e) A new h-list is initialsed in the following way:
- if the vh-switch that caused the mode change was *\noindent, the newly created h-list is empty;
- if the vh-switch that caused the mode change was anything else (*\indent or any horizontal command), an empty box of width *\parindent is put at the beginning of the h-list.
(f) The following elements are appended to the beginning of the input token list:
- the contents of the token register \everypar (normally this register is empty),
- the vh-switch, provided it is a horizontal command; this is the explicit vh-switches *\indent and *\noindent are not put back into the input token list.
The rest of the input token list remains unchanged.

(g) Execution of the commands from the input token list starts. The commands are supplied by the level 2 of \TeX.

6 Answer to the question Q1
Let us recall the question Q1 of Section 2. We have asked about the difference between
(*) \everypar{\def\indent{1}}
\indent 3 is a prime number.
and
(**) \everypar{\def\vrule{1}}
\vrule 3 is a prime number.

From the point (f) of the list of actions performed by \TeX at the beginning of a paragraph (Section 5.2) we can draw the following conclusions: If a paragraph has stared from the \indent command, the token \indent is not put back to the input token list, therefore after executing the actions (a)–(f) the input token lists differ in both cases.

In the case (*) the list is: \[\text{\def\indent{1}3\text{is a prime number.}}\]; in the case (**) the list contains one more token: \[\text{\def\vrule{1}3\text{is a prime number.}}\].

Since redefining \indent has nothing to do with the remainder of the list, the typesetting result in the case (a) will be ‘3 is a prime number.’

In the case (**) the token \vrule is first defined as a macro expanding to the token 1 and then the newly defined macro \vrule is expanded to 1. Therefore in this case the result will be ‘13 is a prime number.’

This example shows some of consequences of the rule that the explicit vh-switches (\indent and \noindent) are not put back into the input token list after switching to h-mode.

7 Switching from horizontal to vertical mode
When the level 3 of \TeX executes commands in h-mode, some commands cause closing the h-list and performing some actions that lead to switching from h-mode to v-mode.

In Section 7.1 below we say when \TeX switches from h-mode to v-mode, i.e., we list the commands that cause switching. Then we explain how this mode change is performed.

7.1 Switching from horizontal to vertical mode: when
The commands listed below are called hv-switches, because if executed in h-mode they usually cause \TeX to complete the h-mode and switch back to the enclosing v-mode or iv-mode. Similarly to the case of vh-switches, there are two groups of switches:
(a) explicit hv-switches:
- *\par (any token the current meaning of which is the same as the meaning of the token \par when \TeX starts job);
(b) implicit hv-switches (called by Knuth vertical commands):
- *\unvbox;
- *\unvcopy;
- *\halign;
- *\hrule;
- *\vskip;
- *\vfil;
- *\vfill;
- *\vss;
- *\vfilneg;
- *\end;
- *\dump.

7.2 Switching from horizontal to vertical mode: how
The behavior of \TeX when it reads a hv-switch heavily depends on the type of the switch. If the switch is a vertical command (implicit hv-switch), \TeX proceeds as follows:
- it inserts a token \par at the beginning of the input token list (before the hv-switch token), regardless of the meaning of the \par token;
- it starts executing commands from the input list (possibly expanding \par if currently it is a macro).

It should be emphasized that \TeX does not change the mode before reading the token \par and that the expanded meaning of \par may redefine the token that triggered the action (please note the danger of looping).

If the switch is explicit (*\par), \TeX ‘truly’ finishes the paragraph, performing all or some of the actions (a)–(h) listed below.

\TeX’s behavior depends on whether the h-list is empty or not at the moment. If the h-list contains at least one element, all of the actions (a)–(h) are performed. If the h-list is empty, only the actions marked with an asterisk are executed, i.e., (e), (g) and (h).

All possible actions are:
(a) discarding the final element of the h-list, provided it is glue or leaders;
(b) appending to the end of the h-list the following three elements:
8 Answer to the question Q2

The question was:

\( (*) \) \texttt{parindent}=0mm \ indent\par
and
\( (** \) \texttt{noindent}\par

Recall that we start in v-mode. The assignment of \( (*) \) \texttt{parindent}=0mm is just an assignment and does not append anything to the v-list. In both cases the command switching to h-mode (\texttt{indent} or \texttt{noindent}) causes appending the vertical glue of the size \texttt{parskip} to the vertical list.

The command \texttt{par} works differently in both cases (see Section 7.2) because h-lists constructed are different:

\( (*) \) h-list at the moment of executing of the \texttt{par} command contains a box of width 0 mm,
\( (** \) h-list at the moment of executing of the \texttt{par} command is empty (the \texttt{noindent} command does not append anything to the h-list).

So, according to what has been said in Section 7.2, points (a) and (b), in the case \( (*) \text{\TeX} \) ‘breaks into lines’ a list containing:

- the empty box,
- \texttt{parfillskip glue},
- \texttt{parfillskip glue,}