A DISCIPLINE FOR THE PROGRAMMING OF INTERACTIVE I/O IN PASCAL

C. Bron, E.J. Dijkstra
Twente University of Technology
Dept. of Electrical Engineering
P.O. Box 217, 7500 AE Enschede, The Netherlands

In this note it is discussed how a compatible view can be maintained between the sequential file concept of PASCAL and terminal I/O without any alterations of, or new proposals with regard to the set of standard I/O procedures [1] (p. 161-163, 84-87).
This note has been inspired by a recent paper in SIGPLAN-notices [2] and by a discussion which envolved in the course of an implementation of PASCAL for the PDP-11.

Since input from terminal is a sequence of characters, we restrict this discussion to textfiles only.
As noted in [2] the main problems in establishing a compatible view between sequential files and terminal input are: the value of the file buffer (1) and the treatment of line separators (2).

Ad (1): since eof(terminalin) never holds, the value of terminalin↑ must always be defined. This poses a problem with regard to the initialization of terminalin↑. Either the user is forced to type in some character before he is actually willing to supply input, or the system performs some standard initialization with an innocent character like a blank. This decision may cause problems when, subsequently, layout-sensitive input has to be processed, and it may therefore impose a restriction on the way in which terminal input can be supplied.

Ad (2): With the existence of lineseparators the standard procedures eoln and readln are associated. In particular: readln skips over a line separator and should make the filebuffer terminalin↑ equal to the first character of the next line. If terminal input is line oriented, i.e. messages and data are always supplied on a line, and closed off by a line-separator, it may be a serious nuisance that the first character of the next message must be given when the end of the previous message is being processed (by readln). Often, terminal input is line-buffered by a terminal-concentrator, which would imply that in order to define the next input character a whole line must be specified.

The solution we propose is straightforward, but obscured by the fact that traditionally line separators are considered as line terminators.
This tradition is clearly reflected in the name of the PASCAL-function: eoln (end of line). If we take the name line separator literally, we are however equally justified in considering it as the beginning of a line.
Based on this view it would be more natural to rename eoln: lsep, and to leave it up to a systems implementor to decide whether he likes to synchronize with line separators as line starters or line terminators.
As it turns out, viewing line separators as line starters works very natural for terminal input:
After the processing of an input message the file terminalin remains in the eoln-state. When a next message is to be processed, this is prepared by a call on "readln", thus skipping over the line separator and preparing terminalin↑ with the first character of the next message.
Used in this way the name "readln" becomes more natural: a line will

actually be read (in case of line buffering) or prepared for reading.
We reject the disadvantage mentioned in [2] that the processing of data would be unnatural in the form: readln; read(a,b,c,...).
It may reflect the choice of an unpleasant interface for readln in the version with a non-standard parameterlist. At least there should have been a symmetrical version, which first skips the line-separator and then assigns to the variables the values derived from the input file.
However, since we don't like non-standard parameterlists anyway, we are just as happy with the parameterless form of readln, used as in the example above.

Three additional benefits are derived from the view exposed above:
(1): The natural initialization of the file terminalin is now: "at the beginning of a line". It can therefore be initialized in the "lsep"-state which does not require an artificial value in the file-buffer. (PASCAL requires the buffer to be filled with a blank in this case ([1] p.162)). This initialization nicely corresponds with the virgin state of a terminal device when a program is started.
(2): The neutral state of terminalin during a conversation is always the lsep-state.
(3): Considering line separators to belong to the line they precede, may facilitate the processing of linenumbers (in sequential file systems that support linenumbering).
The linenumber may now be considered as an inseparable part of the line-separator. It will be processed by readln and thus defines the number of the line which is bound to be processed next.

There is one additional aspect that has to be taken into account if our view is adopted universally (i.e. not only for terminal input but also for stored sequential files), notably: how to structure a file of lines in such a way that the eof-"mark" appears in a convenient position.
Our preference would be for the eof-state in text-files to imply the lsep-state (this may not be the case for most existing implementations, and is not defined in this way by the PASCAL-report).
Line-oriented processing of a textfile f would then proceed as follows:

```
while not eof(f) do
begin readln(f); readrestofline(f){lsep(f)} end
```

Alternatively we may impose the discipline to terminate the last line by a line-separator (immediately followed by eof, of course).
The corresponding program structure would then be:

```
loop readln(f)
     exit if eof(f);
     readrestofline(f) {lsep(f)}
end {loop}
```

Without the loop-construct this can be rewritten as:

```
readln(f);
while not eof(f) do
begin readrestofline(f); {lsep(f)} readln(f) end
```

And this brings us back to existing programming practice, with the exception of the additional call of readln at the beginning, which nicely corresponds to the proposed initialization of terminalin.

Terminal output and mixed input/output

Terminal output presents no complications in the PASCAL sequential file model. We may conceptually associate each call on put(terminalout) with the physical writing of the character on the terminal.
Conversations, in which input and output are mixed on the same display

may proceed in terms of an alternation of lines, or in the form of a question and reply on the same line (if write is used in stead of writeln). Sometimes, however, it may be convenient to have several questions and answers on the same line. We suggest that his can be accomplished without any form of program control. All that is necessary is to specify an additional control character on the console keyboard (apart from the normal one used to terminate a line) to have the internal effect of a line separator. However, this control character is not echoed as a CRLF, possibly as a blank.
It will be clear that in this way the interactive terminal user can specify to some extent the layout of his conversation.

March 16th, 1979.

[1] Jensen, K. & Wirth, N.
"PASCAL User Manual and Report" (Springer 1975).
[2] Clark, R.G.
"Interactive Input in PASCAL",
SIGPLAN Notices 14 (#2, Feb.1979), 9-13.