PERKINS81
Perkins H.; Lazy I/O is Not the Answer; ACM SIGPLAN Notices 16, 4 (April 1981) pp. 81-88.

In this paper, Perkins discusses the synchronization problems with input from terminal devices in the programming language Pascal. He describes one of the solutions - lazy I/O - and argues why this is not a good solution. In its stead he presents another solution and his arguments as to why it is superior to the lazy I/O approach.

In Pascal, a file is a sequence of items and each file has a buffer variable associated with it, containing the item that will be returned by the next call to the READ procedure. In the case of text files, this one character "look-ahead" implies that after a file is RESET, the file must be read so that the buffer variable will contain the next character to be returned by READ. Thus, in the case of terminal input, the file is RESET at the start of the program which means that the file must be read before any statements can be executed. Thus, it is not possible to prompt the user before he has to enter the initial input.

A second problem occurs with the READLN procedure. After executing READLN, the buffer variable must contain the first character of the next line. However, what is usually meant by READLN(A,B,C) is read the values of A, B, and C and skip to the beginning of the next line. If READLN(A,B,C) is used to read 3 numbers, it will do so and then advance the file position to the beginning of the line beyond the one containing the third number.

One solution to the problem, lazy I/O, delays the actual input operation until the program actually references the buffer variable (calls EOLN, EOF, READ, or READLN). The RESET operation is used to open the file and it sets a flag indicating that the buffer has not had a value read in yet. Thus each time one of the above procedures is called, the flag is checked and the file is read if necessary.

The author then argues that this solution has a number of problems. First, programs that depend on lazy I/O are not portable since not all Pascal implementations use it. Secondly, programs that depend on lazy I/O are non-standard (i.e. do not conform to the ISO Pascal standard). Third, lazy I/O has rather confusing semantics and does not completely hide the programmer from the one character look-ahead inherent in Pascal (i.e. invoking EOF or EOLN can cause the file to be read which is not normally the case). The last argument Perkins presents is that lazy I/O is not efficient because each I/O operation causes the flag variable to be checked, etc and this inefficiency is contrary to one of the design goals of Pascal.

The solution proposed by the author is as follows: Put a slash (/) in the program statement after the name of each file to be used for interactive input. When such a file is RESET, the file is put into a state where it thinks there is a new line separator in the buffer. Thus, to read a new line from the terminal, the programmer should first call the READLN procedure with no arguments (other than the file) and he should never use READLN with any variables in its parameter list.

The author then proceeds to argue that programs which use this solution are much easier to transport to other systems - with or without lazy I/O. If the system does not support lazy I/O then the program will wait for immediately after the standard input file is RESET, in which case the user should enter a carriage return to put the buffer in its line-separator state. The program will then continue with proper synchronization. If the system does support lazy I/O then a READ statement must be added to the beginning of the program and a carriage return entered at the beginning of program execution to get proper synchronization.

This paper points out the problems with input synchronization in Pascal, and exposes the problems with the lazy I/O solution. The author presents a useful solution, but one wonders whether the programs really are portable since they execute differently and in some cases still need to be modified to make them work. Also, the solution would also appear to restrict the usefulness of some of the Pascal procedures, i.e. the READLN procedure