CHARACTER STREAMS

A.D. Birrell and R.M. Needham

Computer Laboratory, University of Cambridge

"Why do intelligent people designing operating systems get into apparently ridiculous messes about character codes?" [1]

When designing an operating system we typically include facilities for handling material which consists basically of characters, to provide, for example, input to compilers, output for messages, for listings, or for terminals. We describe such facilities as stream handling, and the objects handled as streams. A stream is thus a first-in first-out buffer for characters, together with a suitable set of operations. Specifying streams causes the designer to take various decisions about character codes, record structures, and carriage control characters.

There are also purely internal decisions, concerning, for example, format on disc, double buffering etc. We are concerned here with the former class of decisions about what a stream consists of. It appears that whatever decisions are made on these points are unsuitable from some points of view or for some purposes.

There are three major areas of criticism:

1) Some sequences of input cannot be mapped into a system-provided stream without loss of information.

2) The stream may not be able to be readily mapped to or from the I/O model provided by a language system.

3) The stream may not be convenient for producing some desired sequences of physical output.

The origins and desinations of streams in a system may be of considerably varying complexity and the natural formats may also vary. Some are as simple as cards and interactive terminals, others may reach the complexity of plotters and typesetters; many are defined by language systems or text-processing software.

In most present systems, when writing a stream we write it in a format which is specifically acceptable at the destination we have in mind. This is often not a conscious action, since most destinations in a system require the same format. The remainder have to be treated specially when writing the stream. However, if a wide variety of destinations exist, or if we do not know the format required by the destination (as will occur frequently in distributed systems or across networks), this approach becomes unsatisfactory.

Our view of the problem is often complicated by the existence of some form of 'record structure' for the stream. The purpose of this record structure is generally to break the stream up into units which can be handled conveniently for purposes of display or analysis. However, in order to impose a record structure on a stream, the operating system usually decides unilaterally on input what pattern it will interpret as end of record, and this decision is bound inextricably into the character stream. This often leads to carriage return, line feed, or end-of-card being raised to an elevated status and these characters must forever after be treated specially. In our view, structure for the purpose of display or analysis should be imposed by the program displaying or analysing the stream. For a single stream, this may at different times correspond to a word, or a FORTRAN statement, or a line, or a screenful. When this structure is no longer considered a property of the stream, we can consider the remaining problems.

Typically, the input sequence when a stream is created is different from the output sequence desired when that stream is used. The former is dependent on the originating device or system and the latter on the destination. Thus there is inevitably a translation involved. For an operating system with m different source formats and n different destination formats there are potentially $m \times n$ different translation algorithms. What the traditional operating system character stream facility attempts to do is to reduce this potentially vast number of translations by translating all inputs into a single form. There are then only $m+n$ different translation algorithms. Unfortunately, this technique leads to the three problems given above.

It would appear that a solution may be found by not performing on input any translation or reformatting which may involve loss of information. The interposition of a stream between a produce and a consumer should be transparent. Provided that a stream as represented internally contains a type marker which indicates its format (and which may also be an indication of where it came from), the consumer of the stream should be able to render it into any format he requires at the time of consumption. The consumer should not require that the stream be already in a particular format, but merely that the required view of it be readily computable. This is not to say that an excessive number of translation algorithms is required; for the many cases where present techniques work well an analogous approach can be used. The crucial difference is that what used to be an imposed definition becomes at the most a default, and any stream has an unambiguous statement of format accompanying it. Where there used to be a requirement for standardisation of representations of streams, there is now only a requirement for standardisation and proper allocation of values in a type field. With a large enough field this is easy. We are here taking it for granted that the underlying operating system is such that shareable translation algorithms may be provided informally as required without complex procedures for "putting them in the System".

The problems we address here showed some signs of going away with the increased use of devices which support ASCII strings and diminishing use of media such as cards. However the same problems have returned with added virulence through the use of computer networks and the transmission of documents in formats appropriate to advanced editors and document-production systems. For example, it is not satisfactory to have to look at a raw printout of a document in order to guess what would have to be done to make a compilable program. A simple labelling scheme would do away with this problem.

The purpose of this note is to suggest that the traditional translation to a standard internal format which one of us confesses to having advocated in lectures for many years should be abandoned as an oversimplification. Record structures or anything similar are purely an analytic view imposed on data by the user of it for his own benefit; they should constitute a voluntary view and not an enforced one.

[1] Cambridge University Examination papers: Diploma in Computer Science, 1971, Paper III, Question 3.