Published: 8/11/66

Identification.

On the Interpretation of ASCII Character Streams within Multics.
J. H. Saltzer, C. Strachey

Acknowledgement

The concept of a canonical representation of a printed line image which is described here has been used in at least two character oriented systems, in TYPSET on the IBM 7094 (as suggested by Earl Van Horn) and the TITAN operating system on the ATLAS computer.

Discussion

Characters are intended ultimately for human communication, and conventions about a character stream must be made with this in mind. A character stream is a representation of printed lines. In general, there are many possible character streams which represent the same line. In particular, on input a typist may produce the same printed line twice with different sets of key strokes. For example, the line

```
     start     lda  alpha,4   get first result.
```

may have been typed in with either spaces or horizontal tabs separating the fields; one cannot tell by looking at the printed image. Since the human not blessed with a tape recorder memory cannot by reading distinguish between several ways of typing a printed representation, _no program_ should deliberately attempt to distinguish, either.

For example, a program should be able to compare easily two character streams to see if they are the "same" in the sense that they produce the same printed image. It follows that all character input to Multics must be converted into a standard (_canonical_) form. Similarly , all programs producing character output, including editors, must produce the canonical form of output stream.

Effectively, we have said that of all possible strings of ASCII characters, only certain of those strings will ever be found within Multics. All of those strings which produce the same "equivalent" printed effect on a typewriter console are represented within Multics as one string, the canonical form for that printed image.

No restriction has been placed on the human being at his console; he is free to type a non-canonical character stream. This stream will automatically be converted to the canonical form before it reaches his program. (There must be an escape hatch for the user who wants his program to receive the raw input from his typewriter, unprocessed in any way. We assume that such an escape hatch is provided.)

Similarly, a device interface module (DIM) is free to rework a canonical stream on output into a different form if, for example, the different form happens to print more rapidly or reliably on this device.

We assume that every DIM is able to determine unambiguously what precise physical motion of the device corresponds to the actual character stream coming from or going to it. In particular, the DIM must know the location of physical tab settings. This requirement places a constraint on devices with movable tab stops: When the tab stops are moved, the DIM must be informed of the new settings.

<u>The Canonical Form</u>

To describe the canonical form, we give a set of definitions of a canonical message. Each definition is followed by a discussion of its implications. Formal definitions are included for the benefit of readers who find them useful. For the reader who finds them confusing, they can be safely ignored without loss of content. In the formal definitions, capitalized abbreviations stand for defined Multics control characters as given in section BC.2.01 and the vertical bar means "or".

1. The canonical form deals with messages. A message consists of a sequence of print positions, possibly separated by, beginning, or ending with carriage motion.

```
<message> ::= <carriage motion>|<print position>
     |<print position><message>
     |<carriage motion><print position><message>
```

The most important property of the canonical form is that graphics are in the order that they appear on the printed page reading from left to right and top to bottom. Between the graphic characters appear only the carriage motion characters which are necessary to move the carriage from one graphic to the next. Overstruck graphics are stored in a standard form including a backspace character (see below).

2. There are two mutually exclusive types of carriage motion, gross motion and simple motion.

```
⟨carriage motion⟩ ::= ⟨gross motion⟩|⟨simple motion⟩
          |⟨gross motion⟩⟨simple motion⟩
```

Carriage motion generally appears between two graphics; the amount of motion represented depends only on the relative position of the two graphics on the page. Simple motion separates characters within a printed line; it includes positioning, for example, for super- and subscripts. Gross motion separates lines.

3. Gross motion consists of any number of successive New Line (NL) characters.

```
           ⟨gross motion⟩ ::= ⟨NL⟩|⟨gross motion⟩⟨NL⟩
```

The DIM must translate vertical tabs into new line characters on input.

4. Simple motion consists of any number of Space characters (SP) followed by some number (possibly zero) of vertical half line forward (HLF) or reverse (HLR) characters. The number of vertical half line feed characters is exactly the number needed to move the carriage from the lowest character of the preceding print position.

```
⟨simple motion⟩ ::= ⟨SP⟩|⟨SP⟩⟨simple motion⟩
         |⟨simple vertical motion⟩

⟨simple vertical motion⟩ ::= ⟨up feed⟩|⟨down feed⟩

⟨up feed⟩ ::= ⟨HLR⟩|⟨HLR⟩ ⟨up feed⟩

⟨down feed⟩ ::= ⟨HLF⟩|⟨HLF⟩ ⟨down feed⟩
```

The basis for the amount of simple carriage motion represented is always the horizontal and vertical distance between successive graphics that appears on the actual device. In the translation to and from the canonical form, the DIM must of course take into account the actual (possibly variable) horizontal tab stops on the physical device.

5. A print position consists of some non-zero number of character positions, occupying different half line vertical positions in the same horizontal carriage position. All but the last character position of a print position are followed by a backspace character and some number of HLF characters.

   ⟨print position⟩ ::= ⟨character position⟩
        |⟨character position⟩⟨BS⟩⟨down feed⟩⟨print position⟩

6. A character position consists of a sequence of graphic formers separated by backspace characters. The graphic formers are ordered according to the ASCII numeric value of the graphics they contain. (The first graphic former contains the graphic with the smallest code, etc.) Two graphic formers containing the same graphic will never appear in the same character position.

   ⟨character position⟩ ::= ⟨graphic former⟩
            |⟨graphic former⟩⟨BS⟩⟨character position⟩

   Note that all possible uses of a backspace character in a raw input stream have been covered by statements about horizontal carriage movements and overstruck graphics.

7. A graphic former is a possibly zero-length sequence of graphic controls followed by one of the 94 ASCII non-blank graphic characters.

   ⟨graphic former⟩ ::= ⟨graphic⟩|⟨setup sequence⟩⟨graphic⟩

8. A graphic setup sequence is a color shift or a bell (BEL) or a color shift followed by a bell. The color shift only appears when the following graphic is to be a different color from the preceding one in the message.

   ⟨setup sequence⟩ ::= ⟨color shift⟩|⟨BEL⟩
   |⟨color shift⟩⟨BEL⟩

   ⟨color shift⟩ ::= ⟨RRS⟩|⟨BRS⟩

   In the absence of a color shift, the first graphic in a message is printed in black shift.

Other control characters are treated similarly to bell. They appear immediately before the next graphic typed, in the order typed. By virtue of the above definitions, the defined Multics control characters HT, VT, and CR will never appear in a canonical stream.

The apparent complexity of the canonical form is a result of its generality in dealing with all possible combinations of typewriter carriage motions. Viewed in the perspective of present day language input to computer systems, we observe that many of the alternatives are rarely, if ever, encountered. In fact for most input, the following three statements, describing a simplified canonical form, are completely adequate:

1. A message consists of strings of character positions separated by carriage motion.

2. Carriage motions consists of New Line or Space Characters.

3. Character positions consist of a single graphic or an occasional overstruck graphic. A character position representing overstrikes contains the numerically smallest graphic, a backspace character, the next largest graphic, etc.

Thus we may conclude that for the most part, the canonical stream will differ little with the raw input stream from which it was derived.

Examples.

In this section are several illustrations of canonical form. The examples do not attempt to cover every conceivable variation or combination of characters, but rather illustrate the intent and the method. (In the examples, assume that the typist's machine has horizontal tab stops set at 11, 21, 31, etc.)

Example 1:

```
Typist:          This is ordinary text.<NL>
Printed line:    This is ordinary text.
Canonical form:  This is ordinary text.<NL>
```

For the case of simple, straight line input, the canonical form reduces to the original key strokes of the typist. Most input probably falls into this category.

Example 2:

```
Typist:          start<HT>lda<HT>alpha,4<HT>get argument<NL>
Printed line:    start     lda       alpha,4   get argument
Canonical form:  start     lda       alpha,4   get argument<NL>
```

HT is the fixed horizontal tab typed in; the tabs have been converted to blanks in the canonical form.

Example 3:

```
Typist:           Here full<BS><BS><BS><BS>____ means that<NL>
Printed line:     Here full means that
Canonical form:   Here _<BS>f_<BS>u_<BS>l_<BS>l means that<NL>
```

Here is probably the most common example of canonical conversion, to insure that overstruck graphics are stored in a standard pattern.

Example 4:

```
Typist:
We see no solu <BS>tion<CR><HT><BS><BS><BS>__<NL>
Printed line:
We see no solution
Canonical form:
We see _<BS>n_<BS>o solution<NL>
```

(Recall that the carriage return (CR) does not produce a line feed.) The most important property of the canonical form is that meanderings of the typist within a line are irrelevant. Example 4 illustrates that the typist need merely concern himself with the printed image. Instead of a tab and three backspaces, the typist could have typed seven space characters and produced the same printed image and same canonical form.