Published: 8/14/67

Identification

Code Conversion

E. L. Ivie, D. L. Stone

Purpose

The Input/Output System (IOS) provides certain basic capabilities for transforming data coming from and going to I/O devices. The purpose of this section is to summarize what these capabilites are and how they can be utilized. For a more detailed description of their operation and implementation the reader is referred to the sections of BF.10.

General Discussion

Multics has adopted the revised ASCII character set as the standard system character set. This set includes 94 printing graphic symbols and 34 control functions (including the space). The internal codes used in the 6.45 store to represent each of these graphics and functions are given in the table of Section BC.2.01.

The sets of graphic symbols and control functions used by the I/O devices which can be connected to Multics do not in general match the standard ASCII set. There are graphics and control functions on the device which have no counterpart in ASCII, and there are ASCII graphics and control functions which cannot be produced on the device.

Even for those graphics and control functions which are both defined in ASCII and representable on the device, the bit pattern generated or accepted by the device does not in general match the ASCII bit code.

Therefore, the task of communicating with Multics through a given I/O device involves a translation process which converts the 'language' of the device into the 'language' of Multics and vice versa. This translation or transformation process is called code conversion. The transformation from device to Multics is called input code conversion, while the transformation from Multics to device is called output code conversion.

See Section BF.1.02 for a list of the devices for which code conversion is automatically performed.

Code Conversion Modes

Multics has provision for more that one possible code conversion transformation in communicating to and from a given I/O device. These possible transformations are called code conversion modes. On most (currently all) character-oriented devices there are two input code conversion modes and three output code conversion modes.

Each device has a default input and/or output code conversion mode. This is the mode that is used until another mode is specified. Code conversion modes can be changed by the mode argument of an attach or changemode call. See Section BF.1.01 for the details on how this is done.

Input Code Conversion Modes

The two input code conversion modes are raw and canonical. The default mode is canonical.

Raw Mode

In the raw input mode the data is delivered to the user exactly as it was delivered to the IOS by the device. No reordering, transformation, or alteration of any kind is performed by the CC or any other IOS module.

This mode is required for binary input. It also provides the user with an escape mechanism so that he can perform his own code conversion if he wishes or use the device in some other unforseen way.

Canonical Mode

The canonical input mode is provided for the person who is generating data on a character-oriented device such as a keypunch or typewriter. There are six operations performed on a character string when the input code conversion mode is canonical. They are described below in the order in which they are performed.

1. Conversion of the string to 9-bit ASCII:
This step maps each device character bit pattern into its equivalent ASCII code. The result is obvious for those device graphics and control functions which are also defined in ASCII.

Certain non-ASCII graphics on I/O devices have been defined to be 'stylized' versions of ASCII graphics in Section BC.2.04. There is a straight-forward mapping performed for these characters also.

If a device generates a bit pattern which has no equivalent ASCII code assigned, a status bit is set (See BF.1.21) and the bit

pattern is left in the input string unchanged.

2. Elimination of hidden character sequences:
Input code conversion has been defined as a translation from device to Multics language. Included in the device language is the ability to interject certain directives to control certain aspects of the translation process.

One such capability effectively inhibits communication with the system while some local operation is performed on the device. This would be useful, for example, if one reached the right margin of the device before reaching the 'logical' end of the line being typed. By typing the three-character sequence, escape, "c", and new line, the carriage of the typewriter can be returned to the left margin without actually designating an end of line.

Hidden character elimination removes from the character string all three-character sequences which begin with an escape and a "c". See also Section BC.2.04 for a description of this operation.

3. Canonicalization
In addition to having a standard character set (ASCII), Multics requires the character strings be in a fixed, 'canonical' order. The definition of a canonical string is found in Section BC.2.02.

Briefly, canonicalization consists of ordering characters by horizontal position first, by vertical position next, and then by ascending ASCII bit code.

The effect of canonicalization is limited by a special set of characters called 'canonicalization delimiters' (e.g. the new-line character). Each sub-string which occurs between two canonicalization delimiters is separately canonicalized. Delimiters are described more fully in BF.1.09 and BF.10.01.

Canonicalization is done at this point in code conversion so that the typist can have a certain amount of freedom in the way he forms a line.

4. Erase and kill processing:
Since much of the data which passes through input code conversion has been manually generated, there will be mis-typed characters. To correct mistakes, two elementary editing functions are provided. The first function is performed by a special reserved character called the erase character. Each erase character causes all characters in the horizontal position it occupies and in the preceding horizontal position to be deleted from the string.

A kill character erases everything to its left on the line being typed. The new line character limits the effects of the erase and kill characters. It is called an erase-kill delimiter.

Other characters can also be designated as erase-kill delimiters. See Section BF.1.09 and BF.10.01 for a description of delimiters. See Section BC.2.03 for a description of erase and kill.

5. Interpretation of escape sequences:
Many of the consoles that can be connected to Multics are not capable of generating directly the full ASCII character set. For such devices, escape sequences are defined so that the ASCII codes for non-representable characters can be generated by the use of sequences of multiple characters that are available.

Section BC.2.04 lists that escape sequences that are available on each I/O device. This code conversion step scans for defined escape sequences and replaces them with their equivalent ASCII character.

6. Recanonicalization:
Escape processing may destroy the canonical order of the character string. This last operation reorders the string so that the final string produced by input code conversion is canonical.

Output Code Conversion Modes

The output modes are straight, edited, and normal. The normal mode is the default mode.

Straight Mode

The straight output mode implies no conversion at all. It is provided for the same reasons that the raw mode is provided as an input mode.

The Normal and Edited Modes

The operations performed by output code conversion with the mode normal or edited can best be described if one considers the ASCII character set as being divided into six categories.

1. The character is a graphic in the device character set which is to be printed; or the character is a control character whose function is available on the device (i.e., backspace on a 1050 but not half-line feed), and which function is to be performed.

2. The character is to be deleted from the data.

3. The character is to be printed as the shortest escape sequence which defines it (i.e., left parenthesis overstruck by minus sign for a left brace on a 1050 with a 938 ball).

4. The character is to be replaced by a blank in the data.

5. The character is a control character which must be simulated. Such a character is backspace on non-backspaceable devices.

6. The character is precisely the bit pattern which should be sent to the device. Clearly, this category is only open to devices which can cope with the ASCII set, or at least some part thereof. No conversion is necessary.

In the normal code conversion mode, as many characters as possible are placed in categories (1) , (5) and (6) and all others in category (3). In the edited mode all characters normally in category (3) are distributed between categories (2) and (4) depending upon whether they are controls or graphics, respectively. Non-ASCII characters remain in category (3).

Other distributions of the ASCII characters into these six categories can be obtained by creating a new 'output code conversion table'. The way this is done is described in BF.10.03.