TO: MSPM Distribution
FROM: K. J. Martin
DATE: 11/17/67
SUBJ: BY.4.02

Four changes have taken place in BY.4.02.

1)   The procedures read_in and write_out may be called
     with either varying or non-varying strings.

2)   An entry has been added to write_out which tacks a
     new line character on the character string to be
     written out.

3)   The interim versions for 6.36 are no longer discussed
     in this document. The 6.36 user should refer to
     Multics Checkout Bulletin MCB-27.

4)   A discussion of asynchronous I/0 has been added
     referring the user to BY.4.04.

Identification

User I/O Procedures to Read and Write
read_in, write_out
K. J. Martin

Purpose

Two procedures, read_in and write_out, are provided to
read from the Multics standard input stream, user_input,
and write into the Multics standard output stream, user_output.
They are provided to simplify calls for the user and to
interpret I/O status returns using the Multics standard
error handling mechanism described in BY.11.00 - BY.11.04.

Usage

        call read_in (workspace, n, nin);

        call write_out (workspace, n);

        call write_out$nl (workspace, n);

where

workspace   is a character string (either varying or non-varying)
            at least n characters in length.

n is a fixed decimal integer indicating how many
            characters are to be read or written. If n
            is zero or negative, n is taken to be the length
            of workspace.

nin   is the number of characters actually read.

In read_in, input (up to n characters) is read from the
stream user-input into the character string, workspace.
Input is read up to and including the first break character
found in the stream, user_input. That is, read_in reads
the smaller of 1) n characters, or 2) the number of characters
up to and including the break character; then returns
in nin, the number of characters read. In normal command
usage, the only break character is the new line, <NL>.
However, the user may define his break characters using
the I/O call, breaks, described in BF.1.12.

Write_out writes from the character string, workspace,
into the stream, user_output. Write_out writes n characters.
The $nl entry adds a new line character to the character
string contained in workspace before writing.

Implementation

The procedures read_in and write_out are much alike except
for their call to the I/O system and the value returned
by read_in. Hence they will be considered together.

The procedures examine n to determine whether it is
negative. If so, n = length (workspace);

where length is a PL/I built-in function. If write_out$nl was
called, n is incremented by 1 and a new line character is
placed in workspace. Then one of the following calls
(described in BF.1.12) is made:

call read ("user_input", 0, addr (workspace), n, nin,
          status);

call write ("user_output", 0, addr (workspace), n, status);

where addr is a PL/I built-in function which returns a
pointer to the character string, workspace. Declarations
are:

dcl workspace char (*),

    (n,nin) fixed binary (17),

    status bit (144);

The bit string, status, is returned by the I/O system
containing status information about the transaction.
BF.1.21 contains a complete description of status information.
When control returns, read_in and write_out call check_io_status
(see BY.4.03) and if an error is indicated, call seterr
(see BY.11.01) with the error information. They then
signal read_in_err or write_out_err respectively.

Asynchronous I/O

The streams user_input and user_output are asynchronous
by default. This means that at any time, more information
may have been read by the device than the user has yet
requested in read calls. Subsequent read calls collect
that information. On writing, some of the information
which the user thinks he has written may not actually
have appeared yet on the device. Normally the user is
not affected by these asynchronous characteristics.

Occassionally, however, the user may want to start all
I/O over fresh. He can wipe out read-ahead or write-behind
data by calling the procedures reset_user_in and reset_user_out
described in BY.4.04.