MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Reply to: Project MAC
545 Technology Square
Cambridge, Mass. 02139

Telephone: (617) 253-6016

To: R. C. Daley

From: J. H. Saltzer

Date: November 23, 1973

Subject: New Multics I/O System

Your memo of November 19, 1973, to M. D. McLaren, about the new I/O system proposals just reached me. I agree with your concerns, especially the ones related to the amount of work required to clear up loose ends, convert old programs to new interfaces, and document everything.

In addition, I would like to add some specific technical concerns which the Project MAC Computer Systems Research Division has about the proposal. These concerns are in some cases quite complex technical issues, and our preliminary thinking may have overlooked an obvious way around some perceived problem. Nevertheless, it seems appropriate to bring these up to find out if they are indeed as serious as they appear on the surface.

Your general observations suggest that you would insist that ios_ be removed from the system as part of the iox_ installation (or on some schedule thereafter), and that all users interface to I/O through features of the PL/I/Fortran/Command Language interface. This insistence produces a very different picture than the one we had been assuming previously in discussions about the I/O system. If removal of ios_ occurs, our analysis suggests that Project MAC and CSR have several current applications which will stop working permanently, and several planned applications which will become impossible or awkward to implement.

If we are restricted to using PL/I language features as our I/O interface, the following difficulties seem to arise:

1. There seems to be no way to economically implement full duplex prompting on escape characters. This feature involves the user typing a non-printing control character whenever he wants attention from his program; his program then wakes up and prompts with a message, then returns to reading the same line. Similarly, use of ASCII control characters (Control Z, Control Q, etc.) as signal triggers is desired in order to successfully embed LISP and MACSYMA in Multics. What is needed is the ability to choose as delimiters a set of the ASCII control characters, and have the delimiter delivered to the program doing the read so that it may decide what action to take depending on which delimiter arrived. PL/I does not permit change in or lists of delimiter characters, and does not deliver the delimiter character to the user.

2. There is no way to do the equivalent of reset read or reset write, both of which are essential for interactive programs which permit typeahead. Even the text editors use this feature of ios_, for which I know of no corresponding PL/I language feature. Further, we would like to be able to implement the feature which allows a program which must interrupt the typist (for example, to print a warning) to be able to then echo back to him his partially typed line, so that he may continue typing, sure of what will go in. The ARPANET TELNET protocol was recently revised to permit implementation of this feature, and the proposed revision of the typewriter IOSIM would also permit implementation of this feature. Unfortunately, we see no way to implement it within the constraints of PL/I I/O.

3. In use of the network, in use of the proposed and partly implemented "dial" facility, and in the design of the newly-proposed I/O daemon it is necessary to be able to initiate reading on several I/O streams, and request a wakeup if any one of the streams produces an input wakeup character. This facility is currently obtained through a special mode call to the typewriter IOSIM, or by bypassing the typewriter IOSIM, depending on the application. The lack of a mode concept and the lack of implemented synchronization mechanisms would mean that one could not continue to use this feature from within the constraints of PL/I.

4. The present Typewriter IOSIM and the proposed new one both implement a "raw" or "vanilla" mode, which permits direct control of the terminal device by the user program. PL/I I/O calls apparently do not permit this concept. This facility has been used, for example, by APL and by programmers experimenting with new, unsupported terminal types. Another facility which would become inaccessible is the special mode of the typewriter IOSIM which inhibits output when the attention button is depressed. This special mode is used by LISP in implementing the LISP interrupt mechanism, and will be used by the APL connection feature.

Since the proposed iox_ facility generally provides only enough mechanism to implement features accessible through PL/I, most of the same objections would still hold even if the iox_ interface were considered accessible to users, except that the ability to explicitly control modes of IOSIM's would become accessible. In addition, two recently developed, but not yet installed, network IOSIM's which treat network connections as I/O streams will have to be modified; their usefulness in communicating with EBCDIC and other non-ASCII installations will be impaired if the ability to set element sizes and break characters is not available.

On the whole, I do not look forward to the effort which would be required to reimplement network software to interface to iox_; and I would anticipate considerable difficulty in convincing our staff to do that reimplementation, since they would consider the loss of facility involved to be a step backwards. In addition, we have for some time been moving toward more complete support of full-duplex terminals, and we consider full-duplex support to be very important in encouraging computer research types to consider Multics as a useful facility. For these reasons, we feel that permanent support of the facilities provided by the current ios_ interface is necessary, whether by continuing to support ios_ or by upgrading iox_ and making it user-accessible. In summary, then, this is our position:

1. A single interface which supports all I/O is preferable, but not at the expense of needed functions.
2. PL/I I/O does not provide functions we need; a deeper level interface is essential.
3. The currently proposed iox_ does not by itself provide sufficient support for our needs; some of the functions of ios_ are also needed.

Our overall recommendation, considering the amounts of work already invested and still to be invested in the various alternatives, is the following:

1. Upgrade ios_ to permit high speed calls for reading and writing, for example by the strategy suggested by D. Clark in his memo of 11/16/73.
2. Modify the PL/I file I/O support procedures to utilize the high speed ios_ calls, and provide a new PL/I file manipulation IOSIM which uses these new calls as effectively as possible to implement PL/I file I/O.
3. Encourage users where possible to use PL/IO calls rather than direct ios_ calls, after providing assurances that performance will not suffer when compared, say, with calls to ioa_ and ios_$write_ptr.
4. Begin a campaign to upgrade present system IOSIM's to uniformly implement those calls, modes, orders, and defaults which are currently not standardized.

xc: W.J. Burner
D.D. Clark
C.T. Clingen
F.J. Corbató
R.A. Freiberghouse
J.W. Gintell
M.D. McLaren
R.A. Roach
M.D. Schroeder
M.G. Smith
J.R. Steinberg
T.H. VanVleck
D.R. Vinograd
S.H. Webber
B.L. Wolman

10X → 10S mtg 11/[illegible]

1. agree to use a consistent control block à la [illegible] proposal.
2. agree to move entirely to 10 ptrs rather than [illegible].
3. agree to add open independent of attach
4. agree to add record read as well as stream read.
5. Want to provide interface which does not tend to become device dependent.
6. go to Multics standard error codes.
7. Want device dependent abilities
   - rewind, read write
   - modes (current ones) / order calls
   - breaks to be designed / order calls
   - changeable error / [illegible].
   - distinguish mode setting / [illegible] from function invoking
   - element size / order call design

New MTB to come out addressing these issues