H. Kolsky

APL/IDS SYSTEM - POUGHKEEPSIE
Department H-65, Building 956
Extension: 255-8474

May 10, 1971

Memorandum to: Dr. A. D. Falkoff

Subject: Shared Variables

Following are my thoughts regarding shared variables arrived at during the drive back from the meeting on April 27. First of all, let me compliment you for having organized a good program.

The proposal for shared variables is elegant. As far as we can see, it will be possible for us to adopt it without difficulty, and to do it in such a way that users of our system can move over to it without undue inconvenience. The biggest surprise to us was that you do not plan to treat large files on disk as completely transparent variables. We were prepared to follow you in this direction; for example, our disk reading I-Beam is equivalent to:

Z [A + i B]

Where Z can be a file on disk, A the index within it, and B the number of elements desired.

We realize that the implementation which you have used was an expedient for getting in operation quickly, and that something quite different later may be done. Sometimes, however, temporary measures become permanent before the alternatives to them are understood. For this reason, we have the following comments:

a) It is our belief that dividing a process in two parts to be put in separate partitions is nearly always damaging. It usually has two negative side effects:

   1) More main memory space is required.

   2) I/O performance suffers because much of the I/O management has to be turned over to the operating system, which has less information to go on than is available when all of the I/O management is handled within one partition.

b) The APL package provided should be complete within itself. There should be no programming required of the customer to prepare to handle cards, tapes, printer, disks, etc.

c) All customer programming should be done in APL. The customer should not need to know or use COBOL, FORTRAN, PL-1, or assembly language.

d) The two processors involved ordinarily have at least momentarily a master and slave relationship. The only exception we know is simple interrogation, such as looking at a clock, in which one processor is oblivious of the other.

e) It is very important to keep down the amount of main memory required to use APL. We know of more than one APL sale which has been lost because of the space which APL requires.

Following the above lines of reasoning, our implementation works in the following way:

- We are providing all of the necessary controls for cards, tape, disk, and printer as primitive preprogrammed I-Beams. The customer has no programming to do to prepare to use these "processors". They do not require a separate partition. They are stored on disk and with the exception of the space for DTF, etc. within the operating system, take no memory space except when called. Their management is included within the compass of the overall data management of the partition to ensure maximum efficiency.

- We draw a distinction between shared variables on disk, which require no manual attention, and cards, tape, and printer, which do. A shared variable on disk differs from other variables only in having two names, rather than the usual one. One name permits reading, and the other reading and writing. The names are synonymous with security codes, as is true for workspaces and sign-on numbers.

- Tape, card, and printer operations are presumed to be initiated only by the system operator. We have found no condition under which a remote user might reasonably be expected to initiate their operation.

- Our basic practice in the handling of tape shared variables is to transfer data on tape to disk without conversion or formatting of any kind. For example, if a user sends us a tape to be put into the system, we enter its data directly on disk as a shared variable and leave it up to the user to do later all necessary conversion and formatting in APL. This works out well, and in time will be extended to cards and the high-speed printer.

- For our Model 50 system, it appears to be efficient to transfer data directly from the workspace to disk or tape or to card or printer buffer. This eliminates the need for a buffer area in memory through which to transfer the data.

- Our thinking is in line with Mr. Perry's with respect to the amount of use of shared variables. We expect that about 50% of the users of the system at all times will be working with disk files. This means that disk traffic will be high, and that a buffering area in memory would be of significant size. Certain of the data transfer operations which we perform today between disk and workspace involve blocks of 6000 bytes.

- Where communication is required between two or more processors of equal rank, as for example in simulating the play of chess or bridge game, we are able to use the disk as an intermediary slave to provide the equivalent of the kinds of operation you demonstrated.

- The only added memory space which we require to provide shared variables is for a list of active shared variables together with the basic operating system areas, such as DTF's. Each variable occurs in the list only once, regardless of the number of its users.

- We are running into demand for high-speed data transfer from our system to other computers, 2780 printers, etc. This will be handled by treating the high-speed line as a slave processor within the APL partition.

It would seem to us that it should be possible to retain the elegance of expression which you propose and still achieve the simplicity, low memory requirement, and high performance which we hope to have.

S. W. Dunwell

SWD:jc

cc: Mr. D. Beverson
Dr. K. Iverson
Mr. H. Lyons
Mr. P. Phelps
Mr. N. Rasmussen