technical contributions

FIXING PASCAL'S I/O

by Richard J. Cichelli, Research Manager, ANPA/RI, Box 598, Easton, Pa. 18042.

There have been a flurry of articles advocating modifications to Pascal's file facility to improve its functionality for input/output. Here, questions regarding terminal I/O and relative record I/O will be discussed.

Many criticisms of Pascal's file facility contain arguments that Pascal's files don't support the full data set manipulation capabilities of the host's operating system. An alternate view of the situation is to ask if the problem to be solved can have its solution cleanly specified as an algorithm in Pascal. If so, request that the Pascal compiler/system writer provide an implementation complete enough to run the program efficiently. In short, buy compilers and computing systems to run your programs rather than write programs to instruct your (particular) computer.

Wirth created Pascal files. In the Revised Report Section 2, paragraph 10, Wirth defines them as sequences of components of the same type. Although an implementer may map Pascal files into sequential data sets, this isn't required by the definition. The Report doesn't seem to require that the ideas of I/O and files be associated. A valid Pascal implementation could exist on a system which lacks backing storage and third generation file system. If this is the case for your system and you still can run your Pascal programs, what do you care? Besides, future data base oriented systems may avoid the redundancy of a "file system". The problems of named data sets and directories are obviously best dealt with in terms of local predefined (not standard) procedures.

For legible input and output (Report section 12) Pascal has a special type of file called a text file. Text files have a special substructure and special procedures and functions. Since sequences work and Pascal has appropriate facilities for manipulating them (i.e. the Pascal file primatives), it would be very strange if you couldn't make Pascal talk to terminals. Wirth specifically mentions them in the first paragraph of section 12 and, guess what, many implementors have succeeded in implementing exactly what the report calls for and having facile terminal interaction as well. One of the techniques is called "lazy I/O" and it is fully detailed in Pascal News #13.

There are those who want to put random I/O or "direct access files" into Pascal. What's Pascal missing? Surely not random access. In the Report section 2, paragraph 6, the array is discussed and specifically called a random access structure. "But", you say, "I can't fit big direct access files in core". Every implementation of Pascal is likely to have some restrictions. Perhaps an array will need to be stored on bulk storage. Would you embed this limitation in the language and in your algorithms and programs? If you need to worry about a hierarchy of memory access facilities in these days of virtual memory, etc, then a pragma or compiler directive might be the appropriate mechanism for suggesting to a particular compiler that certain data be placed on backing store. Note: There is no prohibition to passing arrays (e.g. an implementation relative records I/O) as program parameters. See the Report section 13. Program parameters can reference any external object. It is only suggested that these are "(usually files)". Thus arrays and pointer based data structures can be external objects to Pascal programs. (The "(usually files)" reference has been removed from the current draft standard document.)

Although doing relative record I/O with Pascal arrays may seem strange at first, adding the unnecessary notion of memory hierarchies to the language is far worse. The IBM System/38 has a uniform 48 bit addressing mechanism. A System/38 applications programmer does quite well while being unaware of the storage location of his data whether it be cache, core, disk buffer or on disk. If the 38 can be said to auger the future, then certainly Pascal shouldn't take a step backwards and introduce concepts which provide no additional functionality.

In summary, fixing Pascal's I/O only requires implementing what the Report suggests.