AIPS MEMO NO. 7

NATIONAL RADIO ASTRONOMY OBSERVATORY
EDGEMONT ROAD CHARLOTTESVILLE, VIRGINIA 22901
TELEPHONE 804 296 0211 TWX 510 587 5482

September 3, 1982

To: Whom it may concern
From: WaWa
For: Whom the bell tolls
Re: Suggested changes in AIPS to make life easier on users, programmers, managers

I. FORTRAN 77

Why don't we make FORTRAN 77 (extended) the AIPS standard instead of the current FORTRAN 66?

Advantages:

A. Easier character string manipulations--You can declare CHARACTER variables or arrays and use character assign, compare, substring and concatenation operations.
B. IF-THEN-ELSE constructions
C. Some generalizations of FOR 66 that we might or might not use, e.g., multiple entry and return in subroutines, zero or negative array bounds, real variables in DO parameters, PARAMETER statements...

Disadvantages:

A. In some cases the advantages of the CHARACTER variables cannot be realized because you can't mix them in structures with non-character variables (viz. our headers). This is basically because the length in bytes of a CHAR variable is unpredictable.
B. ENCODE and DECODE are not standard FOR 77 (neither are they standard FOR 66). They are extensions on the VAX version but are probably not supported on MODCOMP and definitely not supported on IBM versions. There is, however, a relatively simple substitute ("internal transmission").
C. INTEGER*2 is not supported on MODCOMPS as presently advertised. It is supported on VAX and IBM.

Recommendations:

I think advantages A, B, and C far outweigh disadvantages A and B. Disadvantage C is fairly serious, but seeing as (1) the MODCOMP will probably be phased out during next year, and (2) I think later versions of MODCOMP FORTRAN 77 will probably support INTEGER*2, I think we should start using FORTRAN 77 experimentally starting the beginning of next year.

II. Pseudo INTEGER*4

Why not drop Pseudo-I*4 and just use normal I*4?

Advantages:

A. Pseudo-I*4 is a pain in the ass.

Disadvantages:

A. You can't (or couldn't) use normal I*4 on PDP-11s. This may become important if you want to use AIPS on the pipeline.
B. Pseudo-I*4 is convenient for addressing the FPS-120B array processor. Without it you have to use unsigned 16 bit arithmetic or some other such nonsense. Perhaps the AP's could be reprogrammed (actually the host interfaces) to use real I*4.

Recommendation:

Research the problems of the PDP-11's and the AP's. If these are not major, drop Pseudo-I*4.

III. Headers

There is a need to store various pieces of information that must be machine readable, must be fairly easy to access quickly and unambiguously, but that vary in size and structure from application to application. Examples of these are: steering and phase stopping centers if different from the tangent point, scaling factors for individual maps in a data cube, and as yet unspecified parameters for all kinds of specialized applications such as optical data. It has been suggested that we put this stuff in the history file, but this suggestion fails the tests of quick access and unambiguity. The need for flexibility suggests the use of keyword-coded, character string entries (e.g., RA-STEER=193.14563/ MAXIMA=32.3,44,-12.553.../) rather than the current position-coded, binary entries.

If we were starting ab initio I would suggest making the entire catalog headers keyword-coded, but the labor involved in restructuring the whole system to accomodate this seems excessive. Instead I suggest retaining the current binary header for defining the basic structure of the data in the file, and adding a new type of header/history record which is keyword-coded/character string but where it is the programmer's explicit responsibility to maintain brevity, consistency and unambiguity, and to document the keywords and formats he/she uses. These new records could be placed (1) in a new type of ZMIO-access file, (2) in our current catalog file as extensions to the current records, (3) in our current history file in specially marked (possibly linked) records.

I think any of these three would work but (1) seems the most confusing. (3) is probably the simplest.

Recommendation:

Do it (either (2) or (3)) on a timescale consistent with the incorporation of FORTRAN 77, since this will simplify access to the new records. FOR 77 is of course not necessary to use the new records, so the suggestions are not strongly coupled.

IV. Utility Tasks

There are a number of utility tasks that AIPS programmers, station managers and more advanced users often need. These include SETPAR, FILINI (or whatever it's called nowadays), and one or more local system text editors such as SOS or SEDIT. There should be versions of these tasks that can be run as standard AIPS tasks so you don't need to continually enter and exit AIPS. We should try to make the commands to FILINI or SETPAR look more like standard AIPSese; e.g., use free format entry of numbers and more verb-like entry of commands.

Recommendation: Do it.

V. Addressable terminals

Most of the terminals we use are random access addressable, i.e., alphanumeric data can be sent to specific locations on the displays where it replaces, in a legible fashion, older data there. We should make some use of this.

Advantages:

Information of essentially different types can be put in different parts of the screen for legibility, or convenience. For example, you might let the task roll-off information appear at the bottom of the screen (at some low-budget installations, they only allocate one terminal to AIPS). Very important messages on task status might appear in a special area (e.g., APCLN running, or crashed, or waiting for new input...). Rapidly changing data such as present cursor location might appear in a dedicated area until you finalize it by hitting a button.

Disadvantages:

You can't do it on a TEK 4012, which was our original minimum terminal. You would have to put enough information in ZDCH and SETPAR to characterize the terminals in use. It would be fairly complicated.

Recommendation:

Think about it for a while, probably do it in the long run.

VI. More general AIPS responsibilities

The present main AIPS task doesn't do much except use POPS to interpret input information, and run a few verbs. In principle, while waiting for input, which is what AIPS is doing most of the time, it could perform some background duties, of which the most important would be to maintain better control over spawned tasks. Examples might be:

Compare a list of tasks which should be running with those found by a spy-like search in order to find those which have crashed.

Clean up scratch files for crashed tasks.

Query tasks to see if any want additional input. If so, manage the use of the terminal to avoid confusion.

Recommendation: Think about it.