Date: 15 February 1997
To: X3J3
From: Van Snyder
Subject: Pre-connected I/O unit numbers

The capability advocated in Section 7 of X3J3/97-114r2, that allows access to pre-connected I/O unit numbers, was added to the "C" list during X3J3 meeting 140. Illustrative editorial changes that apply to ISO/IEC 1539:1991(E) were suggested in X3J3/97-114r2, but are not reproduced here.

Section 7 of 97-114r2: WRITE(u...) ⇒ PRINT

Every time I write a library subprogram that does output, I find myself wanting to allow output either to stdout, or to a file, or none at all. So I end up with code blocks of the form

```
IF (UNIT == 0) THEN
  WRITE (*,100) ... ! or PRINT 100,...
ELSE IF (UNIT > 0) THEN
  WRITE (UNIT,100) ...
ENDIF
```

My life would be simplified here if the standard provided a mechanism to write onto a unit and have the effect of `WRITE(*,...)` or `PRINT`, and to read from a unit and have the effect of `READ format` or `READ(*,format)`. I'll propose three mechanisms that provide the functionality that I want, have no impact on other parts of the language, are compatible to earlier versions of standard Fortran, and don't add much burden for developers.

The simplest mechanism would be if the standard defined unit numbers for these purposes. The standard presently prohibits negative unit numbers, so the interpretation of standard-conforming programs would be unchanged if Fortran 2000 were to specify that `READ(-1,format)` has the same effect as `READ format` or `READ(*,format)`, that `WRITE(-1,format)` has the same effect as `PRINT format` or `WRITE(*,format)`, and that `WRITE(-2,format)` doesn't do anything. `READ(-2,format)` could be defined to do nothing, or do the same as `READ(-1,format)` or be an error – I don't care, but somebody else might have an opinion. Or X3J3 could choose different unit numbers.

One might also want to allow different unit numbers for units equivalent to Unix's *stdout* and *stderr*.

Less simple, but also workable, would be for the standard to define a syntax of INQUIRE, say `FILE=*`, that returns a unit number that one can use to get the same effect as READ and PRINT, or provide an intrinsic function to return this value (but, lacking arguments, the intrinsic would be a little goofy). (Some implementors might want to return different unit numbers for READ and PRINT, say 5 and 6. So maybe `FILE=*` and `FILE=**`, or *two* intrinsics could be used.) I haven't given any thought to how one could extend the INQUIRE mechanism to provide a non-functional unit.

Given any of these mechanisms, I could just write

```
WRITE(UNIT,what_format_100_really_is)...
```

and assume that if the user wanted stdout, or no output, UNIT would be set accordingly.

When I proposed this for F90, somebody remarked that it would be "impossible" for some implementors. I think the (unimaginative) complainer had in mind that some implementors don't pre-open units for stdin and stdout (say, 5 and 6), and therefore "couldn't" tell me the unit numbers to use. I don't see any related problem whatsoever with specifying the meaning of unit -1 as suggested above: Just test for the unit number being -1, and jump to the code for READ or PRINT.

Otherwise, one could avoid the "it's impossible" argument outlined above thus: The standard could stipulate that negative unit numbers have system-defined effect. Implementors would thereby be free to return either pre-opened positive unit numbers for stdin etc. when INQUIRE (or an intrinsic) asks for them, or use system-dependent negative unit numbers. Anybody who puts a negative unit number into a WRITE statement, but one that wasn't gotten from INQUIRE (or an intrinsic) deserves his non-portability.