To: WG5
From: Walt Brainerd
Subject: Namelist and List-directed I/O

This is a version of X3J3/95-83, revised to reflect comments provided during discussion at the meeting.

The following things that appear to be wrong or confusing regarding namelist I/O in the draft (r4):

1. Blanks are not allowed before the terminating slash (10.9.1, (4) and (5)).

2. Comments are not allowed after the namelist group name or other places that might be reasonable.

3. There are no words restricting slash as a separator in namelist output (as there is in list-directed output).

4. The following two records constitute legal namelist input:

```
&X I=2/
J=3/
```

but there is nothing that says the second line will be ignored; indeed, the expectation is that the J=3/ line will be read as part of the next READ statement.

5. [178:23] says that a namelist value must be a literal constant, but (for example) [180:11-12] says FREE is an acceptable logical value.

6. [181:36-37] uses the term "undelimited character constant"; there is no such thing in the standard.

7. [178:19-22] says that "Each value is either a null value (10.9.1.4) or one of the forms:

c
r*c
r*

but in 10.9.1.4, it says that r* is one of the forms of a null value.

8. The first sentence of 10.9.2 says "The form of the output produced is the same as that required for input, except for the forms of real, character, and logical constants." This is false; for example, complex is also different and the rules about value separators, null values, and comments are all different.

9. The namelist forms and semantics are completely inter-mixed and the

subheads in 10.9 do not match their content. An example is the two sentences in the paragraph [179:25-26].

10. Note 10.25 says that blanks in a comment are part of some value separator.

A major editorial change is not to describe the slash as a value separator and make consistent the use "value" instead of "constant". The most significant substantive change is to allow comments at the end of any line, unless within a character value.

==

Proposal:

I. Replace [153:78] with

"The terminating slash (/) is encountered in the record read during list-directed or namelist input."

II. Replace 10.9 with the following:

10.9 Namelist formatting

The characters in one or more namelist records consist of:

(a) The character "&" optionally preceded by zero or more blanks and followed immediately by the namelist group name specified in the NAMELIST statement,

(b) One or more blanks,

(c) A sequence of zero or more name-value subsequences separated by value separators, and

(d) A terminating slash preceded by zero or more blanks.

Namelist input records also may contain comments (10.9.4).

10.9.1 Namelist name-value subsequences

A <bold>name-value subsequence</bold> consists of an object name or a subobject designator followed by an equals and followed by one or more values and value separators. The equals may optionally be preceded or followed by one or more contiguous blanks.

The end of a record has the same effect as a blank character, unless it is within a character value. Any sequence of two or more consecutive blanks is treated as a single blank, unless it is within a character value.

10.9.2 Namelist values

A <bold>null value</bold> consists of no characters.

A <bold>value</bold> is of the form

[r *] v

where v is a null value or is as described in 10.9.4.2 and 10.9.5 and r is an unsigned, nonzero, integer literal constant. Neither v nor r may have kind type parameters specified. The value v is interpreted as though it had the same kind type parameter as the corresponding list item. The r*v form is equivalent to r successive appearances of the value v. Neither of these forms may contain embedded blanks, except where permitted within the value v.

10.9.3 Namelist value separators

A <bold>namelist value separator</bold> is

(a) One or more blanks, or

(b) A comma preceded or followed by zero or more blanks, or

(c) No characters

A namelist value separator preceding or following a null value must contain a comma unless the null value follows an equals.

A namelist value separator consisting of no characters may appear only in namelist output between adjacent nondelimited character values (10.9.5).

Note: All blanks in a namelist input record are considered to be part of some value separator except for:

(a) Blanks preceding and following the "&" and namelist group name,

(b) Blanks embedded in a character value,

(c) Embedded blanks surrounding the real or imaginary part of a complex value,

(d) Leading blanks following the equals unless followed immediately by a slash or comma,

(e) Blanks between a name and the following equals,

(f) Blanks embedded in a comment, and

(g) Blanks preceding and following the terminating slash.

10.9.4 Namelist input

In addition to the parts described in 10.9, namelist input may contain comments. A <bold>namelist comment</a> begins with a "!" character and ends with the end of the line. A comment may begin at any place that an end of record may occur, except within a character value. A comment may contain any graphic character in the processor-dependent character set. A comment is ignored during namelist input.

Note: A comment that precedes the "&" and namelist group name must be in a record by itself.

Note: A slash within a namelist comment does not terminate the namelist input.

Any graphic characters representable by the processor may appear between the terminating slash and the end of the record; these characters are ignored.

10.9.4.1 Namelist input group object names

In each name-value subsequence, the name must be the name of a namelist group object list item with an optional qualification and the name with the optional qualification must not be a zero-sized array, a zero-sized array section, or a zero-length character string. The optional qualification, if any, must not contain a vector subscript.

If a processor is capable of representing letters in both upper and lower case, a group name or object name is without regard to case.

Within the input data, each name must correspond to a specific namelist group object name. Subscripts, strides, and substring range expressions used to qualify group object names must be optionally signed integer literal constants with no kind type parameters specified. If a namelist group object is an array, the input record corresponding to it may contain either the array name or the designator of a subobject of that array, using the syntax of subobject designators (R602). If the namelist group object name is the name of a variable of derived type, the name in the input record may be either the name of the variable or the designator of one of its components, indicated by qualifying the variable name with the appropriate component name. Successive qualifications may be applied as appropriate to the shape and type of the variable represented.

The order of names in the input records need not match the order of the namelist group object items. The input records need not contain all the names of the namelist group object items. The definition status of any names from the namelist-group-object-list that do not occur in the input record remains unchanged. The name in the input record may be preceded and followed by one or more optional blanks but must not contain embedded blanks.

A namelist group object name or subobject designator may appear in more than one name-value sequence.

10.9.4.2 Namelist input values

The form of the input value must be acceptable for the type of the namelist group object list item as described below. The number and forms of the input values that may follow the equals in a name-value subsequence depend on the shape and type of the object represented by the name in the input record. When the name in the input record is that of a scalar variable of an intrinsic type, the equals must not be followed by more than one value. Blanks are never used as zeros, and embedded blanks are not permitted in value except within character values and complex values as specified below.

When the name in the input record represents an array variable or a variable of derived type, the effect is as if the variable represented were expanded into a sequence of scalar list items of intrinsic data types, in the same way that formatted input/output list items are expanded (9.4.2). Each input value following the equals must then be acceptable to format specifications for the intrinsic type of the list item in the corresponding position in the expanded sequence, except as noted below. The number of values following the equals must not exceed the number of list items in the expanded sequence, but may be less; in the latter case, the effect is as if sufficient null values had been appended to match any remaining list items in the expanded sequence. For example, if the name in the input record is the name of an integer array of size 100, at most 100 values, each of which is either a digit string or a null value, may follow the equals; these values would then be assigned to the elements of the array in array element order.

When the next effective namelist group object list item is of type real, the input form of the input value is that of a numeric input field. A numeric input field is a field suitable for F editing (10.5.1.2.1) that is assumed to have no fractional digits unless a decimal point appears within the field.

When the next effective item is of type complex, the input form of the input value consists of a left parenthesis followed by an ordered pair of numeric input fields separated by a comma and followed by a right parenthesis. The first numeric input field is the real part of the complex value and the second part is the imaginary part. Each of the numeric input fields may be preceded or followed by blanks. The end of a record may occur between the real part and the comma or between the comma and the imaginary part.
When the next effective item is of type logical, the input form of the input value must not include slashes, equals, blanks, or commas among the optional

characters permitted for L editing (10.5.2).

When the next effective item is of type integer, the value in the input record is interpreted as if an Iw edit descriptor with a suitable value of w were used.

When the next effective item is of type character, the input form consists of a delimited sequence of zero or more rep-chars whose kind type parameter is implied by the kind of the corresponding list item. Character values may be continued from the end of one record to the beginning of the next record, but the end of record must not occur between a doubled apostrophe in an apostrophe-delimited value, nor between a doubled quote in a quote-delimited value. The end of the record does not cause a blank or any other character to become part of the value. The value may be continued on as many records as needed. The characters blank, comma, and slash may appear in character values.

Note: Corresponding to a namelist input list item of character data type, the character value must be delimited either with apostrophes or with quotes. The delimiter is required to avoid ambiguity between undelimited character values and object names. The value of the DELIM= specifier, if any, in the OPEN statement for an external file is ignored during namelist input (9.3.4.9).

Let len be the length of the next effective item, and let w be the length of the character value. If len is less than or equal to w, the leftmost len characters of the value are transmitted to the next effective item. If len is greater than w, the value is transmitted to the leftmost w characters of the next effective item and the remaining characters of the next effective item are filled with blanks. The effect is as though the value were assigned to the next effective item in a character assignment statement (7.5.1.4).

10.9.4.3 Namelist input data transfer

The name-value subsequences are evaluated serially, in left-to-right order. Values are assigned to the named entities using formatted data transfer as described in 10.9.4.2. Namelist input is terminated when the terminating slash is encountered. Characters between the terminating slash and the end of the record are ignored.

A null value has no effect on the definition status of the corresponding input list item. If the namelist group object list item is defined, it retains its previous value; if it is undefined, it remains undefined. A null value must not be used as either the real or imaginary part of a complex value, but a single null value may represent an entire complex value.

Note: The end of a record following a value separator, with or without intervening blanks, does not specify a null value in namelist input.

10.9.4.4 Namelist input example

```
INTEGER I; REAL X (8); CHARACTER (11) P; COMPLEX Z;
LOGICAL G
NAMELIST / TODAY / G, I, P, Z, X
READ (*, NML = TODAY)
```

The input data records are:

```
&TODAY I = 12345, X(1) = 12345, X(3:4) = 2*1.5, I=6, !
This is a comment.
P = "ISN'T_BOB'S", Z = (123,0)/
```

The results stored are:

Variable	Value
I	6
X (1)	12345.0
X (2)	unchanged
X (3)	1.5
X (4)	1.5
X (5) - X (8)	unchanged
P	ISN'T_BOB'S
Z	(123.0,0.0)
G	unchanged

10.9.5 Namelist output

The form of the output produced by namelist output shall be same as that required for namelist input, except for the forms of the values and the forms of the value separators between two nondelimited character values.

The namelist group name and the namelist group object names shall be in upper case.

The processor may begin new records as necessary. However, except for complex values and character values, the end of a record must not occur within a value or a name, and blanks must not appear within a value or a name.

Note: The length of the output records is not specified exactly and may be processor dependent.

Except for continuation of delimited character values, each output record begins with a blank character to provide carriage control when the record is printed.

Comments shall not be produced in namelist output.

No characters shall be produced after the terminating slash in namelist output.

A null value shall not be produced by namelist formatting.

Logical output values are T for the value true and F for the value false.

Integer output values are produced with the effect of an Iw edit descriptor.

Real values are produced with the effect of either an F edit descriptor or an E edit descriptor, depending on the magnitude x of the value and a range 10^d1 <= x < 10^d2, where d1 and d2 are processor-dependent integers. If the magnitude x is within this range, the value is produced using 0PFw.d; otherwise, 1PEw.dEe is used.

For numeric output, reasonable processor-dependent integer values of w, d, and e are used for each of the numeric values output.

Complex values are enclosed in parentheses with a comma separating the real and imaginary parts, each produced as defined above for real values. The end of a record may occur between the comma and the imaginary part only if the entire value is as long as, or longer than, an entire record. The only embedded blanks permitted within a complex value are between the comma and the end of a record and one blank at the beginning of the next record.

Character values produced for a file opened without a DELIM= specifier (9.3.4.9) or with a DELIM= specifier with a value of NONE:

(a) Are not delimited by apostrophes or quotation marks,

(b) Are not separated from each other by value separators,

(c) Have each internal apostrophe or quotation mark represented externally by one apostrophe or quotation mark, and

(d) Have a blank character inserted by the processor for carriage control at the beginning of any record that begins with the continuation of a character value from the preceding record.

Note: Namelist output records produced with a DELIM= specifier with a value of NONE and which contain a character value may not be acceptable as namelist input records.

Character values produced for a file opened with a DELIM= specifier with a value of QUOTE are delimited by quotes, are preceded and followed by a value separator, and have each internal quote represented on the external medium by two contiguous quotes.

Character values produced for a file opened with a DELIM= specifier with a value of APOSTROPHE are delimited by apostrophes, are preceded and followed by a value separator, and have each internal apostrophe represented on the external medium by two contiguous apostrophes. If two or more successive values in an array in an output record produced have identical values, the processor has the option of producing a repeated value of the form r*c instead of the sequence of identical values.

==

The list-directed section has a few of the same problems, but they are mostly editorial. To fix those and make it similar to namelist:

III. Replace 10.8 with the following:

10.8 List-directed formatting

The characters in one or more list-directed records constitute a sequence of values and value separators, optionally followed by zero or more blanks and a terminating slash (/).

In a list-directed input record, any graphic characters representable by the processor may appear between a terminating slash and the end of the record; these characters are ignored.

The end of a record has the same effect as a blank character, unless it is within a character value. Any sequence of two or more consecutive blanks is treated as a single blank, unless it is within a character value.

Note: List-directed input/output allows data editing according to the type of the list item instead of by a format specifier. It also allows data to be free-field, that is, separated by commas or blanks.

Note: If no list items are specified in a list-directed input/output statement, one input record is skipped or one empty output record is written.

10.8.1 List-directed values

A <bold>null value</bold> consists of no characters.

A <bold>value</bold> is of the form

[r *] v

where v is a null value or is as described in 10.8.1 and 10.8.2 and r is an unsigned, nonzero, integer literal constant. Neither v nor r may have kind type parameters specified. The value v is interpreted as though it had the same kind type parameter as the corresponding list item. The r*v form is equivalent to r successive appearances of the value v. Neither of these forms may contain embedded blanks, except where permitted within the value v.

10.8.2 List-directed value separators

A <bold>namelist value separator</bold> is

(a) One or more blanks, or

(b) A comma preceded or followed by zero or more blanks, or

(c) No characters

A list-directed value separator preceding or following a null value must contain a comma.

A list-directed value separator consisting of no characters may appear only in namelist output between adjacent nondelimited character values (10.8.4).

Note: All blanks in a list-directed input record are considered to be part of some value separator except for the following:

(a) Leading blanks in the first record read by each execution of a list-directed input statement, unless immediately followed by a slash or comma,

(b) Blanks embedded in a character value,

(c) Embedded blanks surrounding the real or imaginary part of a complex value, and

(d) Blanks preceding or following a terminating slash.

10.8.3 List-directed input

[175:14 to 176:9]. Change "constant" to "value" globally.

A null value has no effect on the definition status of the next effective item. A null value must not be used for either the real or imaginary part of a complex value, but a single null value may represent an entire complex value.

A terminating slash encountered separator during execution of a list-directed input statement causes termination of execution of that input statement after the assignment of the previous value. Any characters remaining in the current record are ignored. If there are additional items in the input list, the effect is as if null values had been supplied for them. Any implied-DO variable in the input list is defined as though enough null values had been supplied for any remaining input list items.

<shaded>An example of list-directed input is:

[176:35-177:11]

10.8.4 List-directed output

[177:13-178:5]. Change "constant" to "value" globally.

A null value shall not be produced by list-directed formatting.

A terminating slash shall not be produced by list-directed formatting.

[178:7-10]. Change "constant" to "value" globally.