Command Line Arguments & Environment Variables

by Craig T. Dedo
January 10, 1997

1. Rationale

Getting command line arguments and environmental variables is obviously desirable since it has been implemented in a vendor-specific manner on a wide variety of machines and a wide variety of compilers. This capability is already part of the standard functionality for C and C++.

This feature allows application developers a way of passing information to their program right at startup. Such information can be very valuable in configuring the program before passing control over to users or in performing other important tasks such as opening documents and files. This requirement requires direct access to the operating system. Right now, these capabilities are beyond the current definition of the Fortran language.

Similarly, providing a means for an executing program to obtain the program's startup command will allow the program to use this information in order to find related files that it needs or for other purposes.

The basic functionality for command line arguments is common among implementations. However, there is no de-facto standard means to specify it. Section 4 of this paper contains a list of some vendor-specific implementations.

Some systems also offer environmental variables, process-defined symbols, process-defined logical names, or system-defined logical names. Some of this functionality could be incorporated by an intrinsic which returns a processor-defined result when passed a character variable.

Not all environments have a command line to return. However, an implementation should return a status field and one status can be that there is no processor-defined command line to return. By analogy, the DATE_AND_TIME intrinsic is provided even though some systems do not have real-time clocks.

Although it is highly likely that windowing operating systems will dominate the computers of the future, it is unlikely that the need for this feature will go away. Most windowing operating systems already have a means of providing command line arguments and/or environmental variables at program startup.

It is the intent of this paper to provide the same functionality which is currently available in C and C++ compilers. Desirable features include:

- A user interface which is easily understood, easy to use, and which has the look-and-feel of Fortran.
- Parallel syntax between the command line and environmental variables.
- Ability to treat the command line as either a string or as a series of arguments. This is similar to the DATE/TIME vs VALUES option in the DATE_AND_TIME routine.

- Automatic parsing of the command line based on blank delimiters. Many users want this feature without the bother of doing the grunt work themselves.
- Allowing for systems which can only return the command line tail vs systems which can return the full command line.
- Ability to provide an error status including the fact that no command line can be provided or the specified environmental variable does not exist.

2. Technical Specification

This proposal adds two separate intrinsic subroutines with parallel syntax. This proposal assumes that the Allocatable Extensions TR will become part of Fortran 2000 in substantially its current form.

The subroutine GET_COMMAND_LINE obtains information from the program's command line. This information includes:
- The number of command line arguments,
- The file name of the program which is executing,
- The command line tail, i.e. that part of the command line which starts with the first non-blank character after the program name,
- Automatic parsing of the command line tail into individual command line arguments, and
- Return of the subroutine's completion status.

The subroutine GET_ENVIRONMENT obtains information from the program's environment.

Most of the arguments of these two subroutines are of type CHARACTER. Any mismatch between the length of the argument and its associated value is resolved according to the usual rules for CHARACTER assignment, i.e., the CHARACTER value is truncated or blank filled on the right as necessary. If the KIND type or character set of the target variable cannot represent the assigned value, an error occurs.

GET_COMMAND_LINE (NARGUMENTS, STARTUP_COMMAND, COMMAND_LINE, COMMAND_ARGUMENTS, ISTATUS) obtains command line arguments and related information. Command line arguments appear after the command which starts the program and are separated from each other by one or more blanks.

All five arguments are optional and are INTENT (OUT).

NARGUMENTS is a scalar of type INTEGER of any kind that is defined on the processor. It is assigned the number of command line arguments which appear after the command which starts the program. If there is no command line or command line tail, it is assigned the value zero (0).

STARTUP_COMMAND is a scalar of type CHARACTER of any kind that is defined on the processor. It is assigned the file name of the program which is executing. The value of STARTUP_COMMAND is the same value which would be returned by executing an INQUIRE (FILE=*file-name-expr*) statement on the program's file name.

COMMAND_LINE is a scalar of type CHARACTER of any kind that is defined on the processor. It is assigned the value of the command line tail, i.e. everything from the first non-blank character following the program name to the end of the command line. If there is no command line or command line tail, it is assigned all blanks.

COMMAND_ARGUMENTS is a rank one allocatable array of type CHARACTER of any kind which is defined on the processor. Each array element is assigned the respective argument of the command line, in the order in which the argument appears on the command line. Each command line argument is separated from the other arguments by blanks. A command line argument which is enclosed by quotes or apostrophes is treated as a single argument, even if it contains embedded blanks. If COMMAND_ARGUMENTS is not allocated at the time of a call to GET_COMMAND_LINE, it is allocated with a dimension of NARGUMENTS, and each element has the length of the longest argument. If there is no command line or command line tail and COMMAND_ARGUMENTS is not already allocated, COMMAND_ARGUMENTS has a dimension of one and is assigned all blanks.

ISTATUS is a scalar of type INTEGER of any kind that is defined on the processor. On return from the subroutine, ISTATUS is assigned the operating system completion status if it is available. Otherwise, the value of ISTATUS is processor dependent.

GET_ENVIRONMENT (NAME, VALUE , ISTATUS) obtains the value of the named environment variable.

NAME is a scalar of type CHARACTER of any kind that is defined on the processor. It is an INTENT (IN) argument. It contains the name of the environment variable for which the value is required.

VALUE is a scalar of type CHARACTER of any kind that is defined on the processor. It is an INTENT (OUT) argument. It is assigned the value of the environment variable which is contained in NAME.

ISTATUS is a scalar of type INTEGER of any kind that is defined on the processor. It is an INTENT (OUT) argument. On return from the subroutine, ISTATUS is assigned the operating system completion status if it is available. Otherwise, the value of ISTATUS is processor dependent.

*Example*s
1. Assume that the operating system's status code for "success" is 1, the executing program's file is $1$DKA100:[DEVELOPMENT.EXAMPLES]DEMO.EXE;15, and the program has the following command line tail:

```
/LIST  "Here comes Fortran 2000!"  EXPERIMENTAL:[FORTRAN_2000]TESTFILE.TXT
```

If successful, the reference GET_COMMAND_LINE (NARGUMENTS=N, STARTUP_COMMAND=S, COMMAND_LINE=C, COMMAND_ARGUMENTS=A, ISTATUS=I) assigns the following values to the arguments:

Argument	Value
N	3
S	$1$DKA100:[DEVELOPMENT.EXAMPLES]DEMO.EXE;15

C	/LIST "Here comes Fortran 2000!" EXPERIMENTAL:[FORTRAN_2000]TESTFILE.TXT
A(1)	/LIST
A(2)	Here comes Fortran 2000!
A(3)	EXPERIMENTAL:[FORTRAN_2000]TESTFILE.TXT
I	1

2. Assume that the operating system's status code for "success" is 0, the value of the PATH environment variable is "C:\WINNT40;C:\WINNT40\SYSTEM32;D:\", and the value of E is PATH. If successful, the reference GET_ENVIRONMENT (NAME=E, VALUE=V, ISTATUS=I) assigns the following values to the arguments

Argument	Value
V	C:\WINNT40;C:\WINNT40\SYSTEM32;D:\
I	0

3. Proposed Edits to be Operated on Later

These edits are preliminary and are provided primarily as a basis for discussion. They are with respect to the Fortran 95 Committee Draft, X3J3 / 96-007r1.

Add the following:

13.14.38a GET_COMMAND_LINE ([NARGUMENTS, STARTUP_COMMAND, COMMAND_LINE, COMMAND_ARGUMENTS, ISTATUS])

Description. Obtains the command line and related information from the operating system.

Class. Subroutine.

Arguments.

NARGUMENTS (optional)
shall be scalar and of type INTEGER of any kind that is defined on the processor. It is an INTENT (OUT) argument. It is assigned the number of command line arguments which appear after the command which starts the program. If there is no command line or command line tail, it is assigned the value zero (0).

STARTUP_COMMAND (optional)
shall be scalar and of type CHARACTER of any kind that is defined on the processor. It is an INTENT (OUT) argument. It is assigned the file name of the program which is executing. The value of STARTUP_COMMAND is the same value which would be returned by executing an INQUIRE (FILE=*file-name-expr*) statement on the program's file name.

COMMAND_LINE (optional)
shall be a scalar of type CHARACTER of any kind that is defined on the processor. It is an INTENT (OUT) argument. It is assigned the value of the command line tail, i.e. everything from the first non-blank character following the program name to the end of the command line. If there is no command line or command line tail, it is assigned all blanks.

COMMAND_ARGUMENTS (optional)

shall be a rank one allocatable array of type CHARACTER of any kind which is defined on the processor. It is an INTENT (OUT) argument. Each array element is assigned the respective argument of the command line, in the order in which the argument appears on the command line. Each command line argument is separated from the other arguments by blanks. A command line argument which is enclosed by quotes or apostrophes is treated as a single argument, even if it contains embedded blanks. If COMMAND_ARGUMENTS is not allocated at the time of a call to GET_COMMAND_LINE, it is allocated with a dimension of NARGUMENTS, and each element has the length of the longest argument. If there is no command line or command line tail and COMMAND_ARGUMENTS is not already allocated, COMMAND_ARGUMENTS has a dimension of one and is assigned all blanks.

ISTATUS (optional)
shall be a scalar of type INTEGER of any kind that is defined on the processor. It is an INTENT (OUT) argument. On return from the subroutine, ISTATUS is assigned the operating system completion status if it is available. Otherwise, ISTATUS is assigned a processor-dependent value.

13.14.38b GET_ENVIRONMENT (NAME, VALUE [, ISTATUS])

Description. Obtains the value of a named environment variable.

Class. Subroutine.

Arguments.

NAME	shall be scalar and of type CHARACTER of any kind that is defined on the processor. It is an INTENT (IN) argument. It contains the name of the environment variable for which the value is required.
VALUE	shall be a scalar of type CHARACTER of any kind that is defined on the processor. It is an INTENT (OUT) argument. It is assigned the value of the environment variable which is contained in NAME.

ISTATUS (optional)
shall be a scalar of type INTEGER of any kind that is defined on the processor. It is an INTENT (OUT) argument. On return from the subroutine, ISTATUS is assigned the operating system completion status if it is available. Otherwise, ISTATUS is assigned a processor-dependent value.

[End of Proposed Edit]

4. Summary of Vendor-Specific Implementations

Here is a summary of some vendor-specific implementations. This information was prepared by David Mattoon as part of the material contained in X3J3 JOR Item 016.

- Digital Fortran (formerly DEC Fortran and VAX Fortran) for OpenVMS requires that the program be invoked as a "foreign command." Subroutine LIB$GET_FOREIGN (COMMAND_LINE, PROMPT, COMMAND_LEN, FLAGS) is called where COMMAND_LINE returns the command line tail, PROMPT displays a prompt for the user to supply a command line tail interactively, COMMAND_LEN optionally returns the length of the command line tail, and FLAGS is an argument for a "utility command collector" not needed for this application.

- IBM AIX/6000: GETARG(I, C) is a subroutine where I specifies which command line argument to return (0=program name), C is an argument of type character and will contain, upon return from GETARG, the command line argument. Subroutine GETENV ('ENVNAM', RESULT) stores in character variable RESULT the value of the environmental variable ENVNAM in the profile file of the current directory.

- HP has a function IARGC() which returns the number of arguments and a subroutine GETARG (THSARG, ARG) which returns the THSARG-th argument from the built-in ARG array.

- Lahey Fortran: Subroutine GETCL returns a string containing the command tail; everything after the command and the blank separator.

- Microsoft Fortran: NARGS()is a function which returns the number of arguments. GETARG (THSARG, ARGMNT, STATUS) is a subroutine where THSARG specifies which command line argument is desired (0=program name), ARGMNT is the actual command line argument, and STATUS is a status: if < 0, an error occurred. If > 0, it is the length of the command line argument returned

5. References

International Business Machines Corporation. December 1993. *AIX/XL Fortran Compiler/6000 Language Reference Version 3 Release 1.* North York, ON: International Business Machines Corporation. pp. 401, 441.

Lahey Computer Systems. 1995. *Fortran 90 Language Reference Revision B.* Incline Village, NV: Lahey Computer Systems. pp. 131, 263.

X3J3 / 96-004r1, X3J3 Journal of Requirements, Items 016, 040, and 041.