Experimental FORTRAN 2003 Interfaces for netCDF 3

Richard Weed, Ph.D.
Associate Research Professor
Engineering Research Center
Mississippi State University
May, 2006

1.0 Introduction

The software in this distribution of netCDF implements experimental FORTRAN 2003 versions of the netCDF version 3 FORTRAN interfaces and the netCDF version 2 compatibility interfaces previously implemented as C jackets using the cfortran.h macros. The new interfaces use the C interoperability facility that is an intrinsic part of the FORTRAN 2003 standard (Refs 1 and 2). This facility allows FORTRAN programmers to call C functions directly without the need of C wrapper or jacket routines to handle namespace mangling and data type conversion. Therefore, the FORTRAN interfaces can be implemented in standard conforming FORTRAN without resorting to the current combination of C and the cfortran.h macro package (Ref. 4) that is highly system and compiler dependent. Based on the author's own experience and comments found in the support section of the netCDF web site, finding a compiler macro for cfortran.h that will correctly build the FORTRAN interface is sometimes a hit or miss proposition. This is particularly true when trying to port netCDF to a new system. With the FORTRAN 2003 C interoperability facility, all that is required is a set of FORTRAN 2003 explicit interfaces that define bindings of the C function names and global data to equivalent FORTRAN names and data. The need to generate C wrappers to handle the namespace mangling is eliminated. In addition, the interoperability of FORTRAN and C data types and pointers is provided by the FORTRAN 2003 ISO_C_BINDING intrinsic system module through a set of predefined FORTRAN data KIND parameters and a set of functions for associating C and FORTRAN pointers and data. Users who are not familiar with either FORTRAN 2003, FORTRAN 95, or FORTRAN 90 should consult the references given at the end of this document.

Example: A typical FORTRAN 2003 interface to a C function cfun

C interface:

```
int cfun(int arg1, const float *arg2, double *arg3, void *arg4)
```

FORTRAN 2003 interface to cfun:

```
Interface
  Function cfun(arg1, arg2, arg3, arg4) BIND(C)
   USE iso_c_binding, ONLY: C_INT, C_FLOAT, C_DOUBLE, C_PTR
   Implicit NONE
```

```
   Integer(C_INT), Intent(IN), VALUE :: arg1 ! pass arg1 in by value
   Real(C_FLOAT),  Intent(IN)  :: arg2       ! pass arg2 in as a const pointer
   Real(C_DOUBLE), Intent(OUT) :: arg3       ! pass arg3 back as a pointer
   Type(C_PTR),  VALUE :: arg4               ! void passed in as value of C_PTR type
    Integer(C_INT) :: cfun                   ! arg4 acts like a pointer to a pointer
  End Function cfun
End Interface
```

1.1 Overview of the new interface software

All the software related to the new interfaces are contained in the f03 subdirectory. The explicit FORTRAN 2003 interfaces to the netCDF version 2 and 3 C functions are defined in module_netcdf_nc_interfaces_.F90 and module_netcdf_fortv2_c_interfaces_.F90 using the FORTRAN 2003 C interoperability facility. The other source files in the f03 directory contain FORTRAN implementations of the C jackets used in the current netCDF releases. The purpose of these routines is to serve as a bridge to the C functions contained in the libsrc directory that form the basis the netCDF library. They handle such tasks as swapping the storage order of various vectors used by netCDF to define how data is stored to match the C order, incrementing and decrementing of id numbers to account for the fact that C arrays start at zero and not one, and insuring data is passed correctly to the C functions. The FORTRAN 2003 interfaces will insure that the data passed to C is interoperable with the expected C data type and that the procedure used to pass the data, either as a pointer or by value, is consistent with the method expected by the C routines. The netCDF 3 nf_ interface names and version 2 compatibility names used in the old C wrappers are duplicated verbatim in the new FORTRAN interfaces. Therefore, these routines can be used to compile legacy programs on systems that have FORTRAN compilers that support the FORTRAN 2003 C interoperability facility without modification to the code.

As an aid to FORTRAN 90/95/2003 programmers who like to use modules to store global data and explicit interfaces, a set of FORTRAN modules are included that define the netCDF parameters for error flags etc. and provide explicit interfaces to the Version 3 FORTRAN interface routines. These modules can be used to replace the netcdf.inc file. The data and Version 3 interface modules are defined in module_netcdf_nf_data_.F90 and module_netcdf_nf_interfaces_.F90. These modules are separate to allow programmers who do not wish to use the explicit interfaces to access the required netCDF data via the netcdf_nf_data module alone. Explicit interfaces cannot be provided for the Version 2 routines because they allow data of different types to be passed to the underlying C routines by a single FORTRAN routine. In C, the dummy arguments for these data are void pointers. To mimic this in FORTRAN, an INTEGER array with assumed size arguments is used to provide a pass-through address for the actual arguments. In FORTRAN 90/95/2003, the use of explicit interfaces force the dummy arguments in a subroutine to conform in type to the actual arguments.

A third module, module_netcdf_f03_.F90, USE associates the other two modules to provide one module that can be referenced in users codes when users need both the netcdf_nf_data and netcdf_nf_interfaces modules in a given program or routine.

For backwards compatibility, the netcdf.inc and ncconfig.inc files that are built along with the old FORTRAN interfaces are retained. They can be used with the new interfaces without modification.

2.0 Differences between the netCDF FORTRAN 2003 interfaces and the existing FORTRAN 90 interfaces

The new FORTRAN 2003 interfaces were designed as replacements for the existing C based Version 3 nf_ interfaces as well as the Version 2 compatibility interfaces. Therefore, all arrays are passed as classical FORTRAN assumed size arrays. This allows the interfaces to be used in existing FORTRAN 77 or FORTRAN 90/95 based codes with little or no modification. All the routines can be declared external and used like other FORTRAN routines without the need to give the compiler explicit interface as is done in the FORTRAN 90 implementation. However, the code must still be compiled with a FORTRAN 2003 compliant compiler or one that supports the FORTRAN 2003 C interoperability facility as an extension.

The netCDF FORTRAN 90 interfaces were designed to support passing data via FORTRAN 90 assumed shape arrays. This means the user must always USE associate the netcdf module to provide the interfaces required when using assumed shape arrays. In addition, the FORTRAN 90 interfaces do not access the underlying netCDF C routines directly. Instead, they use the old C based FORTRAN interfaces to pass data to the netCDF C functions. The FORTRAN 90 interfaces also utilize the ability of FORTRAN 90 to specify optional arguments in function and subroutine calls to provide an interface that merges calls to the different implementations of the netcdf put and get functions into one routine.

The new FORTRAN 2003 interfaces can be used either explicitly in legacy codes that are compiled with a FORTRAN 2003 compliant compiler or implicitly through the FORTRAN 90 interfaces when the new FORTRAN interfaces are included in libnetcdf.a. The combination of the new FORTRAN 2003 interfaces to the underlying netCDF C routines and the existing FORTRAN 90 interfaces provide FORTRAN programmers interfaces to netCDF that adhere to the new FORTRAN 2003 standard.

3.0 Building the new FORTRAN interfaces.

The new interfaces are designed to replace the existing code in the fortran subdirectory. The old interfaces will be built by default unless a -DFortran2003 is specified on both the FPPFLAGS and CPPFLAGS environment variables prior to running configure. If the Fortran2003 define variable is not set, all the code in the f03 directory will be compiled as empty files. The default Makefile in the f03 directory is designed to function like the other Makefiles in the netCDF distribution. The only major requirement is a compiler that supports the FORTRAN 2003 C interoperability facility and the FORTRAN 2003 IMPORT statement. At this time, compilers that support these features are scarce. I recommend using the freeware g95 compiler if your native compiler does not support the

FORTRAN 2003 C interoperability facility (Ref. 5).

Although the new FORTRAN interfaces should compile on any system without modification, there are three system dependencies that the user must take into account when building the new FORTRAN interfaces. These dependencies are related to what the length of the C ptrdiff_t data type is for a given system and if the FORTRAN compiler supports byte (integer*1) and integer*2 data types. ptrdiff_t is not a supported data type in the FORTRAN 2003 C interoperability facility so we have to make a KIND parameter for it in the netcdf_nc_interfaces module. By default, we assume that the C ptrdiff_t data type is equivalent to the C_INTPTR_T KIND parameter defined by the FORTRAN 2003 standard. A new parameter named C_PTRDIFF_T is set equal to C_INTPRT_T in module_netcdf_nc_interfaces_.F90. This parameter is used whenever the C interfaces expect an argument of type ptrdiff_t.

Support for byte and integer*2 data types is included by defining two preprocessor define macros, HAVE_INT1 and HAVE_INT2 prior to running configure. These should be set on the FPPFLAGS environment variable -DHAVE_INT1 and/or -DHAVE_INT2 if your system supports integer*1 and/or integer*2. If you don't set these macros, it is assumed your compiler doesn't support them and the default integer size is used to satisfy the interface for the the schar and short routines. Attempts to use the schar and short routines on systems that don't support the byte or integer*2 data types will result either in a compiler error or the functions returning a NF_EBADTYPE error flag. Remember, these define macros should be set on FPPFLAGS not CPPFLAGS. You also need to set a value for the FPP environment variable before running configure. You can also edit module_netcdf_nf_interfaces_.F90 and module_netcdf_nc_interfaces_.F90 to hardwire in the correct values for your system if you don't want to fool with the cpp directives. The impacted sections are at the top of each file and consist of only a few lines of code.

The following example illustrates how to build netCDF with the new FORTRAN interfaces on a Linux system using the g95 compiler. At this time, no attempt has been made to modify the configure scripts to automatically detect a FORTRAN 2003 compliant compiler and set the Fortran2003 define variable for you so you must set it manually before running configure.

```
> setenv FC g95
> setenv F90 g95
> setenv FPP "g95 -E"
> setenv FPPFLAGS  "-DFortran2003 -DHAVE_INT1 -DHAVE_INT2"
> setenv CPPFLAGS  "-DFortran2003"
> setenv F90FLAGS  "-O3 -Wno=155,157"
> setenv FFLAGS  "-O3 -Wno=155,157"
> ./configure
> make
> make test
> make install
```

If you don't need the Version 2 interfaces, add -DNO_NETCDF_2 to both the FPPFLAGS and CPPFLAGS environment variables. An alternate standalone makefile is provided that will create a library named libnetcdfnf.a in the f03 directory. You will need to edit the local macros.make file to set the appropriate compiler flags and macros.

4.0 Using the FORTRAN 2003 interfaces

As stated previously, the new FORTRAN 2003 interfaces were designed to be a direct replacement for the old C based interfaces. Therefore, they can be used in existing legacy FORTRAN 77 codes with little or no modifications to the code. The only requirements are that you compile the legacy code with a FORTRAN compiler that supports the FORTRAN 2003 C interoperability facility and that you include a path to the netCDF installation include directory as follows:

```
> setenv NETCDF /your_path/netcdf-3.6.1-f03
> g95 -c -I$NETCDF/include oldcode.f
```

The include path is required to insure that the compiler has access to the appropriate module files used by the netcdf_nc_interfaces routines. Veteran FORTRAN 90 programmers might want to use the netcdf_nf_interfaces module to provide explicit interfaces to the nf_ routines along with netcdf_nf_data module to access the default netCDF data normally defined in the netcdf.inc include file. Alternatively, you can just use the netcdf_f03 module to include both the interface and the data modules by implicit USE association

4.1 Generic interface support for arrays of characters in the FORTRAN interfaces

By default, the FORTRAN interfaces assume all character strings including arrays of strings are passed as a single assumed length string. However, some older FORTRAN codes pass strings as an array of individual characters. Special versions of the Version 3 text put and get routines are provided that allow strings to be passed as arrays of individual characters. These routines have an _a appended to the name equivalent Version 3 subroutine name (i.e. nf_get_vars_text_a instead of nf_get_vars_text, etc.). The netcdf_nf_interfaces module provides a generic interface name for these routines that allow the array of characters string format to be processed using the default text routine name (i.e. nf_get_vars_text).

5.0 Testing and verification of the new interfaces

The new interfaces have been tested on a Cray X1 system using Cray's native compilers and on Linux and Mac OS X systems using the g95 compiler. The new interfaces were used to successfully compile and run the nf_test and ftest programs found in the nf_test directory. No modifications were made to the FORTRAN code contained in nf_test. Modifications were restricted to the C routines in nf_test that provide utility functions for the test programs. The test program for the FORTRAN 90 netCDF interfaces which call the default FORTRAN interfaces also ran successfully. In addition, a set of libraries that use

the FORTRAN 90 interfaces to create data files in the Sandia National Laboratories EXODUS database format were built using the current software. A series of test programs that use the EXODUS libraries all ran successfully.

6. Modifications to the existing netCDF code base

None of the existing C code in the libsrc directory was modified in any way. I occasionally looked at the code in libsrc to verify that the interfaces given in the netCDF manuals are correct but that was the only time I touched any of the base netCDF C code. The strength of the FORTRAN 2003 C interoperability facility is that all you need to know about the C function you want to call is its interface, i. e. name, argument and result data types, how the data is passed, and the names of any global variables or structures it needs set prior to being called. Therefore, the FORTRAN 2003 interoperability facility allowed me to write the new interfaces without modification to the code that forms the basis of netCDF.

The only existing netCDF code modified were the C routines in the fortran directory and the fortlib.c routine in the nf_test directory. These modifications were restricted to adding preprocessor #ifndef Fortran2003 statements around the cfortran.h macro definitions in the various source files to include or exclude them as a function of how netCDF is built and the addition of a new C function in fort-v2compat.c to support the conversion of the version 2 imap parameters in the new FORTRAN interfaces. The modifications allowed me to keep the old interfaces as the default build option. The fort-v2compat.c routine is the only routine from the old FORTRAN interface software that is required by the new interfaces. The C functions in fort-v2compat.c that were called by the old interfaces are retained when you build the new interfaces. New FORTRAN 2003 C interoperability interfaces were generated for the C routines called by the new Version 2 interfaces. Similar modifications were made to fortlib.c in the nf_test directory. The Makefiles for these directories along with the root netCDF Makefile and rules.make file were modified to reflect the changes required to implement the new interfaces. In total, these modifications eliminate the use of cfortran.h in all phases of building and testing the netCDF FORTRAN interfaces when you select the new FORTRAN 2003 interfaces as a build option.

7.0 The questions you are dying to ask

"Why did you write this software ? "

This started out as a learning exercise to teach myself something about the new FORTRAN 2003 C interoperability features. It snowballed into the current software package. Based on what I've learned so far, this is the best way I've seen to implement interfaces for C programs/routines called by FORTRAN. It eliminates most of the system dependencies that can occur when you try to implement FORTRAN callable C wrappers around the C functions and gives you a much cleaner and (IMHO) easier to maintain set of interfaces that conform to the FORTRAN standard. Plus; as an old FORTRAN programmer, I think the world of scientific computing would be a much nicer place if there was a lot less C code in it. This is one small attempt at making that dream a reality.

A second motivation for this software was the pain and suffering the author endured in his first attempts to build netCDF on a Cray X1. Although cfortran.h was probably a good idea in the days prior to FORTRAN 2003; in my opinion, in doesn't make much sense to continue using it when you can use the FORTRAN 2003 C interoperability facility to reduce or eliminate most if not all of the system dependencies created with the cfortran.h based interfaces.

"Which version of netCDF are the new FORTRAN interfaces based on ?"

The current version of the Fortran 2003 interfaces is based on netCDF 3.6.1. Support for version 3.6.0 can be obtained by providing a dummy entry point for the nc_inq_format function from 3.6.1. I'll try to keep the code current with each new netCDF release.

"What are your plans for the software ?"

Well, I don't have any except to make it available to the netCDF community for evaluation. If at some future time when FORTRAN 2003 compliant compilers are the norm, they wish to use it to replace the existing FORTRAN interface, then great. If not, thats O.K. too. I've accomplished my original objective and learned enough about the new C interoperability facility to be comfortable using it in other projects. If nothing else, this software can serve as an example for others who wish to learn more about this new feature of FORTRAN.

"What do you consider is an appropriate status level for this software ?"

Beta. Until I can get access to more FORTRAN 2003 compliant compilers I won't know if the current code will work on other systems and compilers besides the X1 and Linux and Mac OS X with g95. Although all the test programs I've tried to date have run successfully, there are still a couple of features of the current implementation that I'm not sure will work with all future FORTRAN 2003 compilers. The first issue arises from logic I implemented to mimic the action of some C utilities used by the old interfaces to convert the FORTRAN arrays that define counts, strides, maps, etc. to the format required by C. For certain conditions, these utilities pass a NULL pointer to C. To implement this behavior in the new interfaces, I had to create a C_PTR type to hold either the address of the target array that can be obtained using the C_LOC function or a null pointer defined by the C_NULL_PTR intrinsic variable. The resulting pointer (and not the actual array) is passed by value to the appropriate C routine. For a while I was puzzled by the need to explicitly use pass by value for the C_PTR variables. It started to make sense when I thought of these variables as pointers to pointers as in C. Therefore, you have to pass the contents of the C_PTR variable which requires the VALUE attribute in the C interoperability interfaces. I'm also not comfortable with using an INTEGER array as the pass-through argument in the Version 2 interfaces. Both the C_PTR logic and the INTEGER arrays appear to work, but I'm not sure that all compiler writers will interpret the standard the way that Cray and g95 do.

"Who are you and what do you do for a living ?"

I work for the Engineering Research Center at Mississippi State University and hold the academic rank of Associate Research Professor. I work at the DoD High Performance Computing and Modernization Programs (HPCMP) Major Shared Resource Center (MSRC) located at the Army Engineer Research and Development Center (ERDC) in Vicksburg, MS. I serve as the Computational Structural Mechanics On-Site for the HPCMP User Productivity Enhancement and Technology Transfer (PET) program.

"Who paid you to do this work ?"

Most of this work was done “after-hours” on my own time but I did work on it some during regular hours. Therefore, I'm obliged to include the following acknowledgment.

“ This work was funded under U.S. Department of Defense Contract No. N62306-01-D7110. Therefore, the U.S. Government reserves the right to use, modify, and distribute this software without restriction.”

“What kind of license or copyright applies to this code ?”

Since this is a derived work, I don't think I can or should impose a license other than what the netCDF folks at UCAR currently require. I've added a copyright statement for my stuff because I think my university requires it but I'll remove it if UCAR objects. My only requirement for using this software is that I get acknowledged as the original author of the code and you don't try to claim this work as your own.

" How do I contact you about this code?"

Address questions, comments, or bug reports to me via email at:

rweed@erc.msstate.edu

Acknowledgment

A special thanks to Andy Vaught, developer of g95, for his work to bring g95 to the world and having the forethought to include the FORTRAN 2003 C interoperability facility in his compiler. I wish some of the major compiler and hardware vendors (this means you Portland Group, Absoft, SGI, etc.) shared Andy's initiative.

8.0 References

1. Metcalf, M., Reid, J., Cohen, M., Fortran 95/2003 explained, Oxford Press, 2004.
2. "Working Draft - Fortran 2003 Standard", J3 document 04-007, www.j3-fortran.org
3. Adams, J, Brainerd, W, et. al., Fortran 95 Handbook, MIT Press, 1997
4. http://www-zeus.desy.de/~burow/cfortran
5. www.g95.org