Upgrade to GFORTRAN and Fortran 2003

G. L. Mesina

2019 IRUG Meeting
Idaho Falls, ID
April 18 – April 19, 2019

INL/CON-19-53482

Outline

- Reasons/Advantages
- History of Upgrades
- Issues
- Results
- Conclusions

History of Architectural Upgrades

- FACT: Must keep concurrency with evolving computer industry or become obsolescent and non-working.
- Fortran 66: original RELAP5 coding
- Fortran 77: conversion after the compilers stabilized in mid-80’s
- Developed “32-bit int / 64-bit real” equivalence in late 80’s
- Adapt to new machines as they become available, mid-80’s & ongoing
  - CDC; Cray; Cray2; DEC: RISC, RISC2, Alpha; HP; IBM (various); SGI; Stardent; SUN-Solaris; Windows; Apple Macintosh (Briefly)
- Ongoing Operating System Adaptations
  - CDC: NOS, NOS/BE, etc.
  - Unix: UNICOS, IBM, HP, SGI, SUN, DEC
  - Windows: 95, 98, ME, XP, 7, …
  - LINUX: Red Hat, SUSE, CYGWIN, …

History of Architectural Upgrades

- Processing mode
  - Scalar, original
  - Vector, mid-80s through mid-00s
  - SMD Parallel (Cray dirs., Open-MP), late 80s to mid-00’s
  - DMD Parallel: PVM coupling to self, early 00’s & ongoing
- Coupling with other programs – PVM Executive, early 00’s & ongoing
- Graphics
  - Whole plant: NPA (1990’s), RGUI (late 90’s to mid-00’s),
  - Input-builders (SNAP, etc.), Plot programs (XMGR, APTplot, etc),
- Restructuring to strongly modular coding, mid-00’s
- Resizable: Fortran 90/95, modules, derived types, pointers…, late-00’s
- Refactoring: ongoing

What is the Planned Upgrade?

- Current status
- RELAP5-3D currently guarantees to build with an Intel Fortran 95 compiler equipped with certain Fortran 2003 extensions
  - Intel compilers released since 2013
- It will run on a Linux or Windows 7 Operating System
  - Installation with MSVS or CYGWIN on Windows

- Upgrade ADDS CAPABILITY to these
- Capability to build with GNU Fortran Compiler
- “Strict” Fortran 2003 standard
  - Strictness of application of the standard varies with compiler

Why Upgrade?

- Longterm Viability – GNU Fortran will be around
  - GNU converts to C-language then compiles, C underlies most O/S and will be around
  - Compiler Vendors for evolving Fortran declining in number
    - **ANSI Fortran 2018 standard** released last year
    - **Only** Cray, GNU, IBM, and Intel have some features
    - PGI (NVIDIA),Flang, and NAG support Fortran 2003
- Incorporation into MOOSE herd
  - INL HPC Cluster fully supports GFORTRAN
  - Access to all MOOSE coding
- Reliability – the probability of failure-free operation for a specified period of time in a specified environment
  - Testing on two different compilers and operating systems reveals errors that just one compiler or O/S would not
  - These are solved before the code is released

Why Upgrade to Fortran 2003 Standard?

- Software quality – Code written to an **ANSI standard** survives
  - Vendors add extensions to the language that, years later, either:
    - Become unsupported
    - Subtly change meaning and code operation
  - **Library quality** software is written in **ANSI Fortran standards**
    - Even some FORTRAN66 library software still compiles & runs on current compilers and O/S
- Portability – ANSI Standard software works on evolving platforms
  - It disallows specialized coding that accesses special hardware that does not survive computer evolution
- Maintainability – Easier and less time-consuming to maintain
  - Disallows vendor extensions that become unavailable and must be rewritten

Operation and Issues

- Development of GNU and Fortran 2003 compilation capability
  - Mostly manual with assists from scripts where possible
  - Proceed directory by directory, upgrading all files within.
  - Order of upgrades induced by usage precedent.
    1. XDR – eXtended Data Representation, machine-indep. Binary
    2. Modules – Directory of Common F90 modules
    3. Envrl – service subprograms: solvers, interpolators, fluid properties
    4. LApack – some math subprograms
    5. Rellic – RELAP5-3D license control
    6. Jacdir – Jacobian matrix calculation
    7. Relap – Program input, physics calculations, and output
    8. Polate – auxiliary standalone fluid property generator
    9. Fluids – Generators for the many fluids RELAP5-3D can use
    10. R5exec – PVM coupling capability

Operation and Issues

- Develop GNU compilation capability first then added Fortran 2003 in first 5 directories
  - Develop an understanding of what was involved
- Did both GNU & Fortran 2003 capabilities at once thereafter.
- REQUIREMENT:
  - Test that code runs with both Intel and GNU compilers
  - Done incrementally. When problem arises, stop and fix
- REQUIREMENT on Development Environment
  - GIT for “version control”
  - CIVET for testing

Preparation

- Comment: both GIT and CIVET have steep learning curves
- CIVET source code requirements
  - No trailing whitespace allowed. All removed
  - No tabs allowed in source code. All replaced
  - Certain keywords disallowed. Removed or replaced.
- Add a GFORTRAN option to all major installation scripts
  - Some new scripts had to be created because the Makefile only accessed IFORT. E.G. LAPACK, Jacobian, polate
- Add Fortran 2003 compiler flag to IFORT and GFORTRAN
- Split lines of source code that exceeded132 character length limit.

Issues

- Level of compiler matters.
  - Several GNU compilers on the HPC.
  - Default compiler could not handle some Fortran 2003 construct properly
  - Cannot mix two (very) different levels of GNU Fortran
- Name mangling of C-language coding
  - Location prefix and postfix underscores prevented linking with GNU compiled Fortran at first
- Equivalence of numbers and characters is not allowed in Fortran 2003
  - Remove character from equivalence w numbers (R-level)
  - Use the internal read or write to transfer where needed
- Some transfer functions in PIB (XDR) pass real to integer and vice-versa
  - Have to use Fortran TRANSFER function to move bits from one to other
  - Important in data-type transformation module for plotting

Issues

- Star-before-length declaration no longer allowed
  - ERROR: real*8, character*20, etc.
  - ->   real(8), character(20), etc.
- Declaration array shapes must be right. E.G. in fluids/D2O/cof.f90
  - ERROR: real(sdk), parameter :: a(10,7) = (/ 70 numbers /)
    - Left side is matrix. Right side is vector
  - ->   real (sdk), parameter :: a_temp(70) = (/ 70 numbers /)
  - ->   real(sdk), parameter :: a(10,7) = reshape (a_temp, [10, 7])
- Declaration initialization of character variables requires right number of characters. E.G.
  - ERROR: character(8), dimension(2) :: filenms = (/ 'beta','kappa'/)
  - ->   character(8), dimension(2) :: filenms = (/ 'beta   ','kappa   '/)
- New IEEE modules provide many constants, such as NaNs, for various uses.

Issues

- Call arguments must EXACTLY match the type of the dummy arguments
  - Attributes must match, such as dimensionality, pointer, etc.
  - No more passing a scalar to a length one vector, vector to matrix
  - Kind matters. Cannot pass 16-byte or 4-byte to an 8-byte dummy
  - Mismatched character length can cause link error or failure to run
- GFORTRAN compiler flag for default 8-byte reals turns “double precision” declaration statement into 16-byte reals
  - Turned “D” exponents into “E” exponents. 1.0D0 -> 1.0E0
  - Turned dabs, dexp, dsqrt, dlog, etc. into abs, exp, sqrt, log, …

Issues

- To pass a 4-byte number, put value in 4-byte variable & pass it
  - E.G: call openPibExportFile(err,0,tpfname,pname,vers,desc)
  - -> integer(ptik), save :: fnum = 0
  - -> call openPibExportFile(err,fnum,tpfname,pname,vers,desc)
- Statement functions are not allowed in Fortran 2003
  - Turn them into contained (internal) function subprograms
- Access to O/S procedures superceded by Fortran intrinsics
  - getarg replaced by get_command_argument
  - iargc replaced by command_argument_count
- Jumps into a “body” block of code from outside is an error
  - E.G. if-then-block, else-block, do-loop body

Issues

- Elimination of Obsolescent constructs
  - Assign keyword
  - Indexed GO TO statement
  - Old platform specific statements
- Formats
  - Cannot continue a character string to the next line. Must break
  - Commas required between format specifiers, even at end of line
  - Format specifier “x” not allowed. Replaced by “1x”
  - Field length required
    - “10 format (a10)” not “10 format (a)”
    - read (5,'(a10,x,i5)’) name, j not read (5,'(a10,x,i)‘ name, j

Summary

- Upgrade progress:
  - Directories upgraded: 8 of 10
    - Relap not finished and r5exec
  - Changed files: 1224 of 8118
- Comparisons on Linux between compiling with IFORT and GFORTRAN
  - Fluid asci table files, *.pr, identical, except H2O 1967
  - All non-restart problems run
- Remaining work
  - Fix restart with GFORTRAN and Fortran 2003
  - Upgrade r5exec
  - Possibly fix 1967 H2O generator.

Summary

- Question to IRUG?
  - Should we fix 1967 H2O generator or keep using the “ASCII” trick?
- The “ASCII” trick is to convert a binary file, such as tpfh2o2, to ASCII
  - Use program stb2a of the fluids directory
  - Output is called a_tpfh2o2
- Thereafter on installation, stb2a inputs a_tpfh2o2 and outputs tpfh2o2
- Currently, the “ASCII” trick is ALREADY used for 1967 water.
- Fixing the generator allows us to generate water properties from a better set of grid points for more accuracy in the future
  - That would change results for all problems that use 1967 water.