Date (9/16/2021 or 9/17/2021

Dr. George Mesina

RELAP5-3D Architectural Upgrades through Gnu Fortran Adaptation

Idaho National Laboratory

Outline

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

Reasons to upgrade code architectural

- Must keep concurrent with evolving computer industry or become obsolescent and non-working.
  - Programs that do not adapt cease to function
    - NPA (Nuclear Plant Analyzer), XMGR5, TkXMGR5, RGUI
    - Proprietary Unix, Old Windows (E.G. Windows 2000)
- Must answer modern user needs or get replaced
  - Old languages: Algol, Cobol, Simula
  - AOL, My Space, etc.
  - B&W visuals, slow-running, fixed memory size, etc.

History of RELAP5 architectural upgrades

Massive important upgrades

- Fortran 66: original RELAP5 coding
- Fortran 77: conversion after the compilers stabilized in mid-80's
- Fortran 90: Early 2000's
- Fortran 95: 2010
- G-Fortran: 2019
  - Required Fortran 2003 transformation
  - Elements of Fortran 2008 added

History of RELAP5 architectural upgrades

- Adapt to new machines as they become available, mid-80's & ongoing (Large efforts)
  - CDC; Cray; Cray2; DEC: RISC, RISC2, Alpha; HP; IBM (various); SGI; Stardent; SUN-Solaris; Windows; Apple Macintosh (Briefly)
  - This included taking advantage of special native hardware and proprietary compilers.
- Ongoing Operating System Adaptations (Large efforts)
  - CDC: NOS, NOS/BE, etc.
  - Unix: UNICOS, IBM, HP, SGI, SUN, DEC
  - Windows: 95, 98, ME, XP, 7, 10, …
  - LINUX: Red Hat, SUSE, CYGWIN, …

History of RELAP5 architectural upgrades

- Processing mode (massive efforts)
  - 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, 1990's - 2012
- Coupling with other programs (massive efforts)
  - PVM Executive, ongoing
  - Special connections: STAR-CCM+, PHISICS, MOOSE (begun)
  - Via & disk-file transfer

History of RELAP5 architectural upgrades

- Restructuring to strongly modular coding, mid-00's
- Resizable Memory: Fortran 90/95, modules, derived types, pointers…, mid-00's
- "Perfect time stepping" based on integers
  - This synchronizes time advancement among coupled codes
- Refactoring: ongoing

Advantages of upgrade to G-Fortran

- **Versatility**: Able to build with another mainline Fortran Compiler increases reliability
  - **Reliability** – the probability of failure-free operation for a specified period of time in a specified environment
  - Each compiler catches runtime errors the other's miss and so eliminates future User Problems
- **Longevity** – GNU Fortran will be around as long a C and Linux
  - GNU translates Fortran to C, then compiles it in C.
- **Fortran 2003 standard**
  - Strictness of application of the standard varies with compiler

Advantages of upgrade to G-Fortran

- **Software quality** – Code written to an **ANSI standard** survives
  - Vendors extensions to the language become, years later:
    - Unsupported
    - Redefined by future standards to perform differently
  - **Library quality** software is written w **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 maintaina dn develop

Development of GNU and Fortran 2003 compilation capability

- Mostly manual work 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

Operation and Issues

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

Preparation for the upgrade

- 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 created because the Makefile of some sub-directories 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 overcome: Compiler failings

- Level of compiler matters.
  - Several GNU compilers available in INL HPC.
  - Default compiler could not handle some Fortran 2003 construct
  - Cannot mix two (very) different levels of GNU Fortran
- Scripts and Makefiles
  - Work out compatible naming convention for Fortran source files
- Name mangling of C-language coding
  - Location prefix and postfix underscores prevented linking with GNU compiled Fortran at first

Issues overcome: Declaration Issues

- GFORTRAN compiler flag for default 8-byte reals turns “double precision” declaration statement into 16-byte reals
  - FIX: Turned “D” exponents into “E” exponents. 1.0D0 -> 1.0E0
  - FIX 2: Turned dabs, dexp, dsqrt, dlog, etc. into abs, exp, sqrt, log, …
- Star-before-length declaration no longer allowed
  - ERROR: real*8, character*20, etc.
  - FIX: real(8), character(20), etc.

Issues overcome: Declaration Issues

- Mismatched array shapes disallowed
  - ERROR: real(sdk), parameter :: a(10,7) = (/ vector of 70 numbers /)
  - FIX: real(sdk), parameter :: a(10,7) = reshape (a_temp, [10, 7])
- Equivalence of numbers to characters disallowed in Fortran 2003
  - Remove character from equivalence w numbers (R-level)
  - Use the internal read or write to transfer where needed

Issues overcome: Subprogram type issues

- Call arguments & dummy arguments types and attributes (dimensionality, pointer, etc) must match EXACTLY
  - Kind matters: Cannot pass 16-byte or 4-byte to an 8-byte dummy or character of one length to character of another: link or runtime error
  - Some PIB (XDR) transfer functions pass real to integer or vice-versa
    - Use Fortran TRANSFER function to move bits from one to other
  - Cannot scalar to a length one vector, or vector to matrix (ENVRL)
- Dummy variable type adjust for constants when linking
  - ERROR: call openPibExportFile(err,0,tpfname,pname,vers,desc)
  - Compiler sets 0 to default, but dummy arg is type pitk, so declare a variable of dummy's type, set it to the constant, and pass it.

Issues overcome: Subprogram issues

- 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
- New IEEE modules provide many constants, such as NaNs, for various uses, though accessing the IEEE module proves tricky.
  - Intel and GNU Fortran at odds, many ways that work for one fails for the other

Issues overcome: Format issues

- 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

Issues overcome: Execution issues

- Jumps into a “body” block of code from outside is an error
  - E.G. if-then-block, else-block, do-loop body
- Initialize all logical variables because uninitialized can default to true or false depending on O/S and level of Fortran compiler
  - Caused restart and strip problems to fail until all were tracked down and initialized

Issues overcome: Operating system issues

- GMAKE sees files with the mod extension as source code for programming language MODULA and, at random
  - Creates a modula compiler command that fails and aborts the build
  - This happens with newtype.mod (frequently, not always) and rarely with any other mod-file
- Use of cpp, not fpp, required with GNU Fortran in fluids directories where more than one pre-compiler directive is invoked.

Issues overcome: Ifort vs Gfort issues

- Format
  - Gnu Fortran won't print floating point in z-format, only integers
  - Intel Fortran does not allow 16-byte integers (verification hexadecimal notation)
  - FIX: A new if-def in verifymod.
- Made uniform set of Makefiles for fluids directory
  - Had to overcome naming convention issues for suffixes
    - F90 and f90 allowed for Fortan 90+ with Intel, not GNU
    - F03 allowed with GNU not Intel

Summary

- All 10 directories converted
- Converted to Fortran 2003 first, then upgraded to GNU Fortran
- Intel and GNU Fortran compilers have incompatible differences
  - Found workarounds through numerous coding experiments
- Massive code modification
  - Over 50% (1230 of 2394) Fortran source code files changed.
- Comparisons on Linux between compiling with IFORT and GFORTRAN
  - Fluid asci table files, *.pr, identical, except H2O 1967
  - All standard installation problems run
- Code builds and runs correctly with both compilers