RELAP5-3D Conversion to Fortran 90

RELAP5 International Users Seminar

Dr. George L. Mesina

August 16-18, 2006

Outline of Presentation

- Goals & Benefits (User and Developer)
- Three Types of FORTRAN 90 Conversion
  - Fixed Length Commons, FA-Files, Rewrites
- Progress and Measurements
- Future Plans

- Acknowledge contributions from:
  - Dr. Richard Riemke and Dr. Paul Murray
  - Peter Cebull (in near future)

Ultimate Goal & Benefits

- **Ultimate Goal: modernize RELAP5-3D**
  - **Improve code legibility and understandability**
  - **Reduce development and maintenance costs**
  - **Extend code longevity**
  - **Take advantage of hardware and software developments**
- **Project goal: convert RELAP5-3D to FORTRAN 90**
  - **Machine independence via F90 intrinsics**
  - **Eliminate memory restrictions**

Goals and Benefits

- Intermediate project goals
  - Eliminate FA "container" array
    - Replace "internal files" with modules
    - Use pointers for plots, trips, controls, etc.
  - Simplify code w/o changing calculations
    - Reorganize databases
    - Apply structured programming
    - Reduce coding via FORTRAN 90 capabilities
  - Replace obsolete coding constructs

Project Goals and Benefits

- User benefits
  - Program expands memory to fit problem (no fixed size limit except for input constraints)
  - Code runs on latest platforms for decades to come
  - Can parallelize across hydrodynamic systems for speed-up
- Developer benefits
  - Much easier to read, understand, modify
  - Modern programming practices/language reduce learning time for new developers

Transient Conversion Methodology

- **Subtasks based on types of conversion**
  - **Fixed length common, FA-Files, rewrites**
- **Fixed length common block COMDECKS have only scalars and short fixed-length arrays.**
  - **Create a module w/ no equivalences.**
- **FA-Files have scalars and variable length arrays organized in sometimes complex patterns.**
  - **Create resizable derived types in modules.**
- **Rewrite: write a new subroutine that does the same thing better.**

Fixed Length Common

- Contents – 3 sections:
  - Data declaration; Variable dictionary; Contained subroutines (initialization)
- SCDAP complication – not part of F90 conversion
  - Maintain COMDECKS for SCDAP usage
  - Pre-compiler flags protect modules & COMDECK
  - Size of COMDECK section sextupled
- RESTART-PLOT file order must be preserved
  - Declared derived type with SEQUENCE statement
  - Scalars become pointers to derived type.

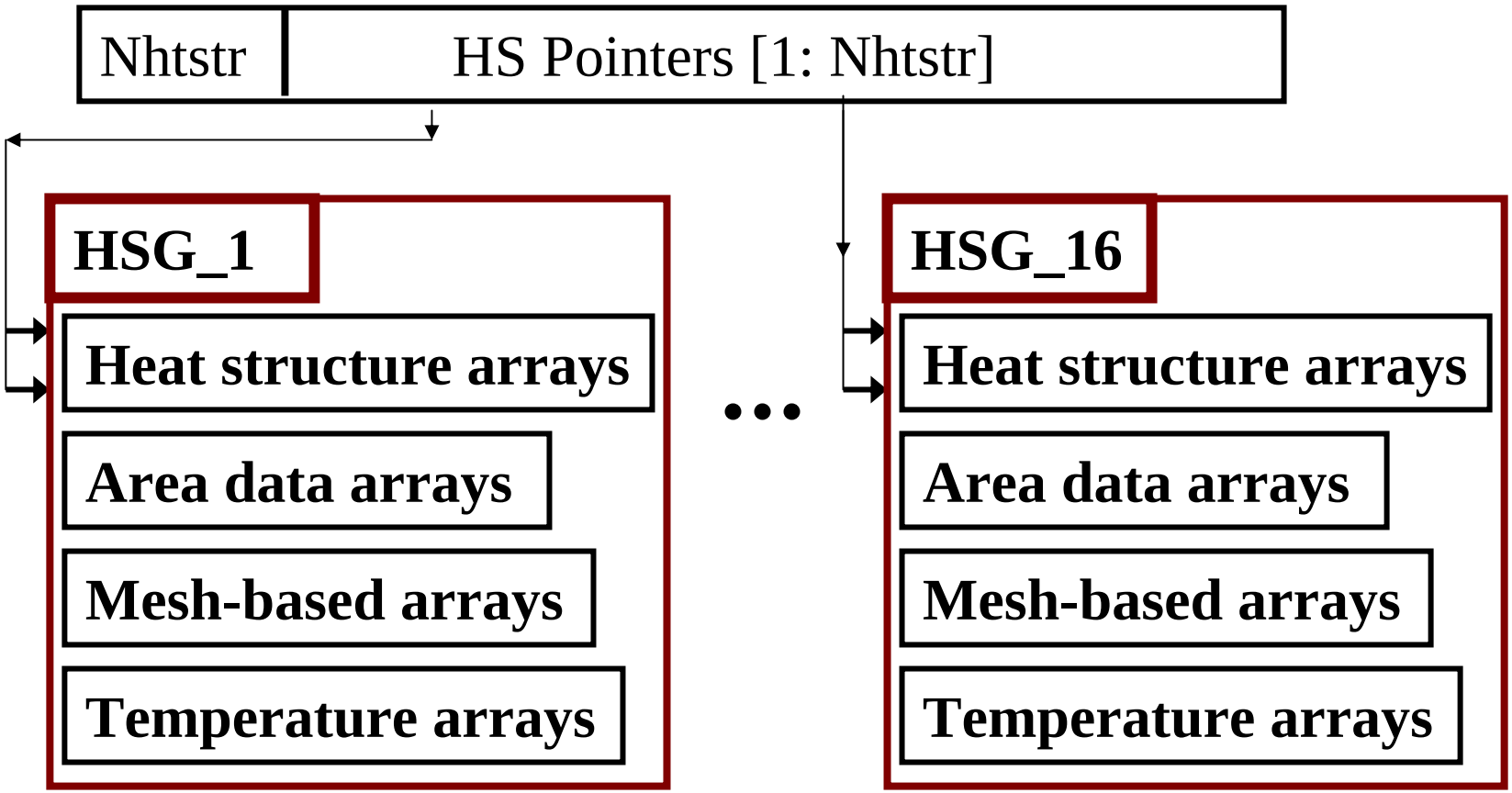
FA-File Conversion

FA-Oriented Database

- **Database organized into *internal files* called FA-files**
- **An FA-file is a collection of data related by physical or computational meaning**
  - **File 4 = control volume data, File 8 = heat data**
- **All FA-file arrays are *equivalenced* to the FA-array**
- **Difficult to understand**
  - **Equivalence of real/integer/character/logical data**
  - **Non-contiguous arrays, conditional arrays**
  - **Pointers (use of FA-indices) w/in & w/o FA-file**
  - **Layout of arrays in FA-file often complex**

Example: typpwr2.i, File 8, HSG_1

Example: typpwr2.i, FA-File 8



- **There are numerous pointers (FA index arrays) between the HSG sections *and* to other FA-files.**

Methodology to convert an FA-file

Decipher → Redesign → Convert → Test

1. **Decipher FA-file**
   - **Read manual and comment comdeck**
   - **Usage in the code reveals database's structure**
2. **Redesign database & implement as F90 module**
   - **Organize data into derived type arrays & scalars**
   - **Create size calculations (where unavailable)**
   - **Create memory allocation subroutines**
   - **Create data transfer subroutines that move data between FA-file and module.**

New Heat Structures Database

- Data is grouped by physical meaning and is contiguous.
- Much simpler structure.

HS derived type array, 146 attributes (former arrays) for each HS
HS_1 HS_2 ... HS_K HS_{K+1} ... ... HS_H

HSG Pointers/Geometry derived type array (4 geom. attributes) HSG starting ordinal in HS, mesh, and temp arrays are 'pointers'

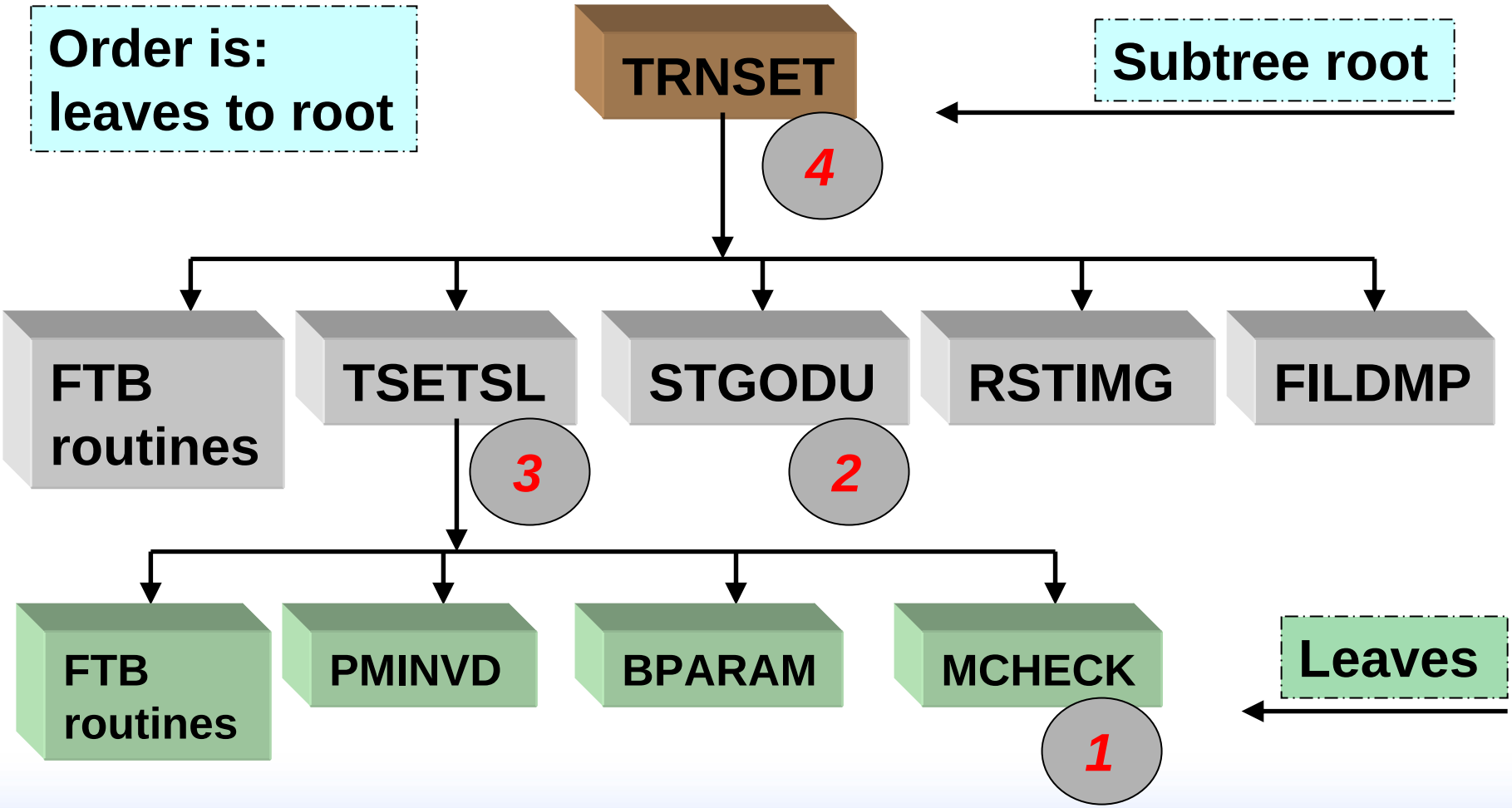
Mesh Point Data derived type array (8 mesh attributes)
X_1 X_2 ... X_G X_{G+1} ... ... X_P

Temperature Data derived type array (2 attributes, old & new)
T_1 T_2 ... T_{GK} T_{GK+1} ... ... T_N

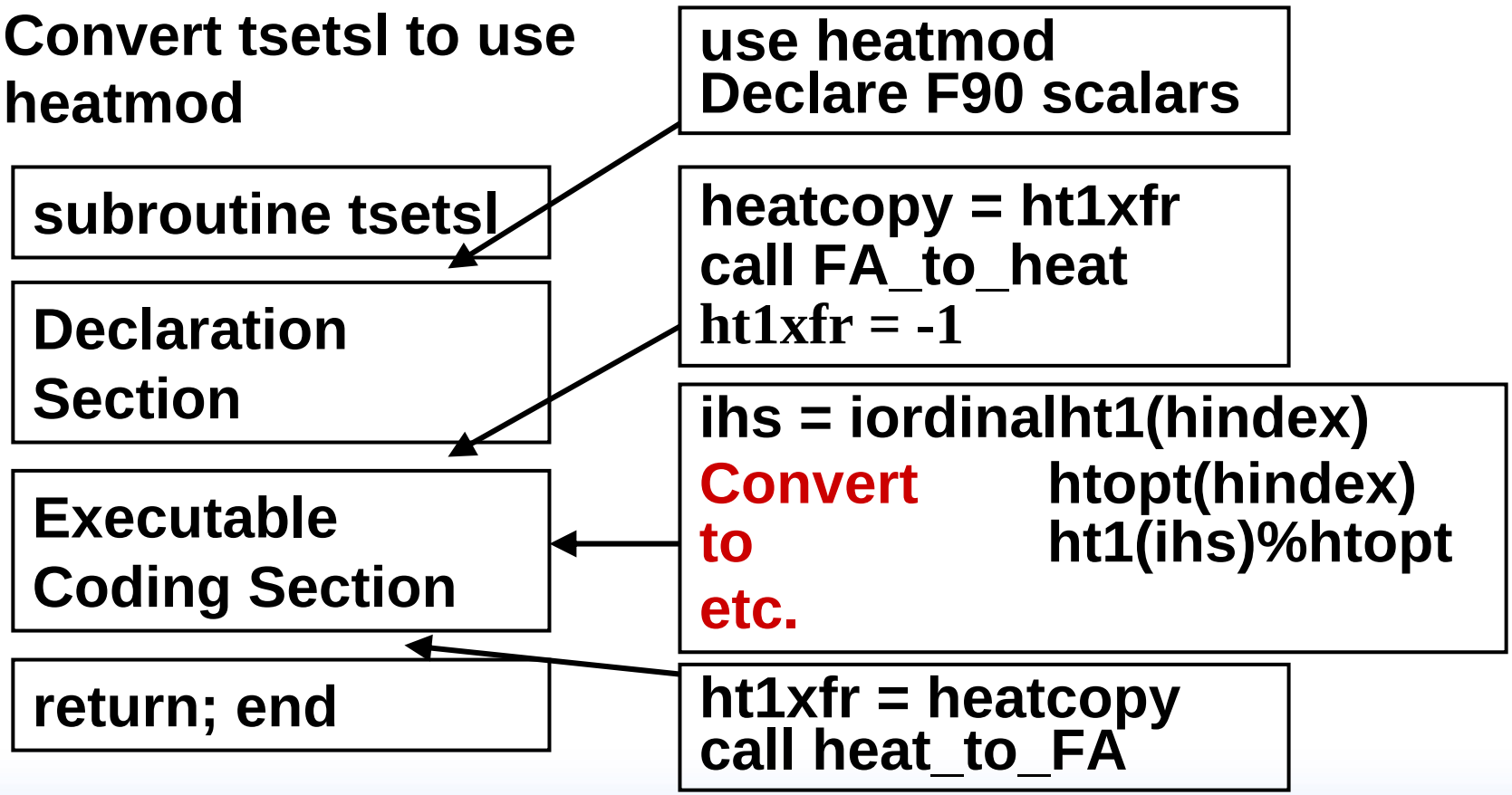
Methodology to convert an FA-file

- Convert source code
  - Add initialization, size, and memory to modmem
  - Convert subroutines that use the FA-file.
    - Add use statement, XFR routines & controls
    - Convert FA arrays to module references
      - Replace FA-indices with ordinals.
      - Handle special issues (indexing, etc.)
  - *Convert subroutines in the proper order*.
- Test and debug to ensure that all test problems produce identical calculations

Conversion Order of an FA File Subtree



Normal File-Conversion Actions



Classifying FA-files

1. **Standard – single fixed stride, all data same length**
   - **FA-Files: 2, 4, 5, 7, 10, 14, 24, 28, 30, 35**
2. **Interwoven – several fixed strides, different lengths**
   - **FA-Files: 20, 25, 37**
3. **Complex – varying strides, 2+ arrays equivalent to one FA index, direct use of FA/IA, 3+ comdecks, . . .**
   - **FA-Files: 1, 3, 6, 8, 9, 11, 12, 13, 18, 21, 27, 29, 31, 32, 33, 38, 43, 44, 47**
4. **Removable – *absorb*, delete, or temporarily disable**
   - **FA-Files: 15, *16,* 17, 19, 22, 23, 26, 34, 36, 39, 40, *41, 42,* 45, 46**

FA Kind vs. How Work Is Done

FA Kind	Module Construction	Source Conversion
Standard	Straightforward. Can automate.	100s to 1000s of lines. Can automate. Some manual changes.
Interwoven	More difficult. Can automate but must modify manually.	Fewer lines. Often manual.
Complex	Very difficult. Always manual.	Always manual.

- Beyond simple conversion, some subroutines must be rewritten.
  - For example: CONVAR, SCNREQ, Restart.

Subroutine Rewrite

Example: SCNREQ, Heat Section

```
      do 380 i=1,nmt4
        if (alph .eq. t4(i)) go to 381
  380  continue
      go to 400
  381  if (itypi .lt. 0) go to 530
      if (filid(8) .eq. 0.0) go to 950
      num1 = num
      num2 = 0
      if (i .eq. 1 .or. i .eq. 11 .or. i .eq. 12 .or. i .eq. 13 .or.
     & i .eq. 14) go to 384
      num1 = num/100
      num2 = num - num1*100
  384  j1 = filndx(8)
      j2 = j1 + nhtstr(j1) - 1
      do 382 j = j1,j2
        j2 = ihtptr(j) + filndx(8)
        if (num1 .eq. htstno(j2)) go to 383
  382  continue
      go to 953
  383  scod(1) = 8
      j1 = locf(fa(filndx(8)))
c  If t4 = httemp
      if (i .eq. 5) go to 305
      if (num2 .gt. 1) go to 952
c  t4 = htvat, htrnr, htchf, hthtc, httemp, htmode, htrg, htgamw, stant,
c       pecl, htpowg, h2gen, oxti, oxto
      go to (301,302,303,304,305,306,307,308,1309,1310,1311,1312,1313,
     & 1314), i
      call fabend
  530  go to (531,532,532,533,534,527,532,528,1535,1536,1537,1538,1539,
     & 1540), i
      call fabend
```

Is request for heat data?

Special data

Heat structure number valid?

Jump to code that sets FA index, label & conv. factor

Example: SCNREQ

- The heat structure section has 105 lines of code.
  - 36 statement labels, 35 GO TO statements
  - Logic paths difficult to follow.
- Each user request section is similar.
- Next user request section found by scrolling forward.
- SCNREQ is difficult to read, understand, modify
  - Minimal comments and character strings
  - Redundant data, inconsistent names, etc.

Example: IREQUEST vs. SCNREQ

- **User request sections are contained subroutines.**

```
      if ( any(alph == t1) ) then
!  General Scalar Quantities
         call reqquan (ialph)
      else if ( any(alph == t2) ) then
!  Volumes
         call reqvol (ivol)
      else if ( any(alph == t3) ) then
!  Junctions
         call reqjun (ijct)
      else if ( any(alph == t4) ) then
!  Heat structures
         call reqheat (iht1)
```

SCNREQ loop

```
         do 380 i=1,nmt4
         if (alph .eq. t4(i)) go to 381
380   continue
         go to 400
```

F90 intrinsic replaces SCNREQ loop

Example: IREQUEST vs. SCNREQ

- **SCNREQ special data handler is hard to understand**

  if (i .eq. 1 .or. i .eq. 11 .or. i .eq. 12 .or. i .eq. 13 .or. i .eq. 14) go to 384

- **The IREQUEST handler is more obvious.**

  if (any(alph == (/ 'htvat', 'htpowg', 'h2gen', 'oxti', 'oxto‘ /))) then

- **SCNREQ assigned GOTO replaced by a CASE statement and a pointer replaces FA-indexing.**

```
select case (alph)
case ('h2gen')
   p => ht1(iht1)%h2gen
   labelType = 'M’
case ('htchf')
```



```
1312  scod(2) = locf(h2gen(j2)) - j1
      if (ityp .eq. 0) go to 1000
1538  conv = cunmf
      labl(33:48) = unms(l)
      write (labl(17:32),'(i10)')num1
      go to 355
```

Measurements of SCNREQ Rewrite

Category		SCNREQ		IREQUEST		Improvement	
Lines of code			2981		1563		1.9
Comments	%	203	6.8%	382	24.4%	1.9	3.6
GOTO stmts	%	717	24.1%	0	0.0%		
Statement labels	%	572	19.2%	6	0.4%	95.3	50.5
Ave. Nesting Depth			46		11		4.2
Max Nesting Depth			138		17		8.1
McCabe Index			687		55		12.5

Progress & Measurements: July 06

- **Task D: Rewrite Special Subroutines**
  - **SCNREQ and CONVAR rewritten**
  - **BPLU half rewritten (10 subroutines)**

- **Task B: Fixed length common blocks**

Task B	FA-File/ Modules	Files
Conversions to F90	46	1889
% Finished	100%	100%

Progress and Measurements

As of August 2006

Task C: FA-file conversion	FA-File/ Modules	Files
Conversions to F90	28.5	684
Absorbed/ deleted/ disabled	15	
Remaining conversions	3.5	190
% finished	93%	77%

July 06 Progress on the F90 Project

- **Permanent modifications**
  - **Number of F90 Modules added: 70**
  - **Lines of code in modules added: 11449**
  - **Lines of USE statements added: 3664**
  - **Lines of source code with %: 29537**
- **Temporary conversion additions**
  - **Lines using IORDINAL functions: 2767**
  - **Lines of data transfer added: 2721**
- **Total *of these* conversions & additions: 50138**

Future Plans

- Complete restructuring
- Complete conversion of transient
- Convert input
- Eliminate FA array
- Convert environmental, fluids, peripherals
- Clean-up FORTRAN 90 conversion
- Final report & paper