Restructuring RELAP5-3D

RELAP5 International Users Seminar

Joshua M. Hykes
Penn State University

Aug 16-18, 2006

Idaho National Laboratory

Structure of Restructuring

- Purpose
- Description of Modular Programming
- Means of restructuring
- Algorithm for applying FOR_STRUCT
- Results

Purpose of Restructuring Effort

- Simplify the coding for increased ease of:
  - Reading & understanding
  - Maintenance
  - Development
- Reduce chance of errors

Essentially, making the developer's job easier.

Modular Programming: Definition

Object-Oriented Programming (OOP)

- **Classes**
- **Hierarchies**

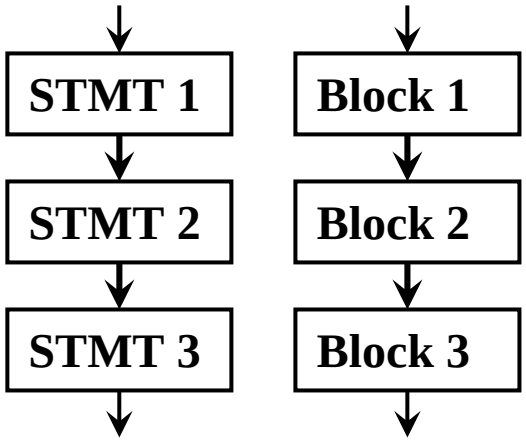
Structured (Procedural) Programming

Blocks of code with only:

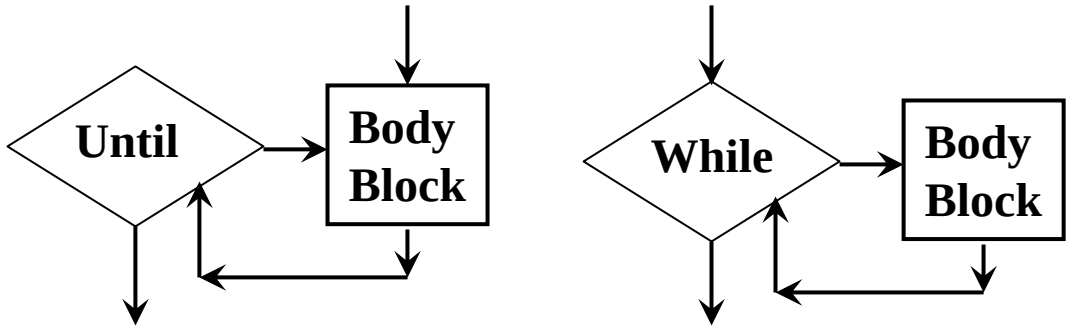
- **1 entry**
- **1 exit (debated)**
- **Sequence**
- **Selection**
- **Iteration**

Structured Blocks

Sequential

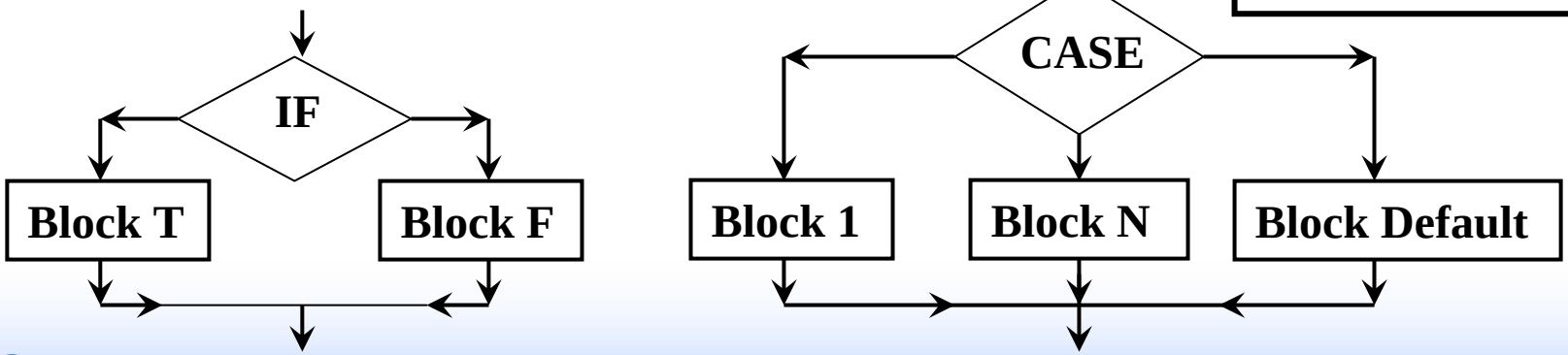


Iteration / Loop



One entry
One exit

If / Case

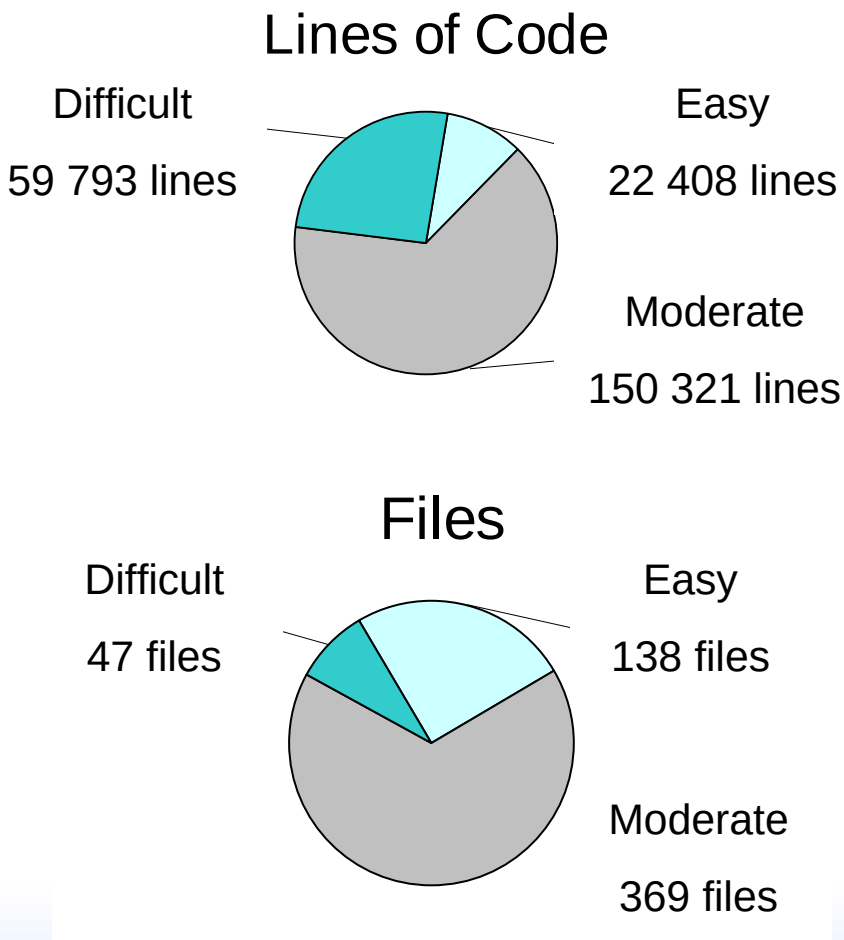


Structured Programming and RELAP: An Oxymoron?

- **RELAP was largely coded without any concept of structured programming.**
  - **~6200 `go to` statements**
  - **Computed `go to` statements**
  - **Arithmetic `if` statements**

Other complications

- **Pre-compiler directives**
  - **0-58 unique PCD's in source files**
- **Some newer (Fortran 90) coding**
- **Sheer size**
  - **Up to 4800 lines of code in source files**



Road to Restructuring

- **FOR_STRUCT version 2.1.1 (FS2.1.1) – a commercial software package for restructuring F77 and earlier Fortran.**
- **Unix shell scripts – take care of some of the extra problems**
- **Manual changes**
  - **`contains` subroutines**
  - **Moved `endif`'s**
  - **Lines over the 72 column limit**

FOR_STRUCT Restructuring

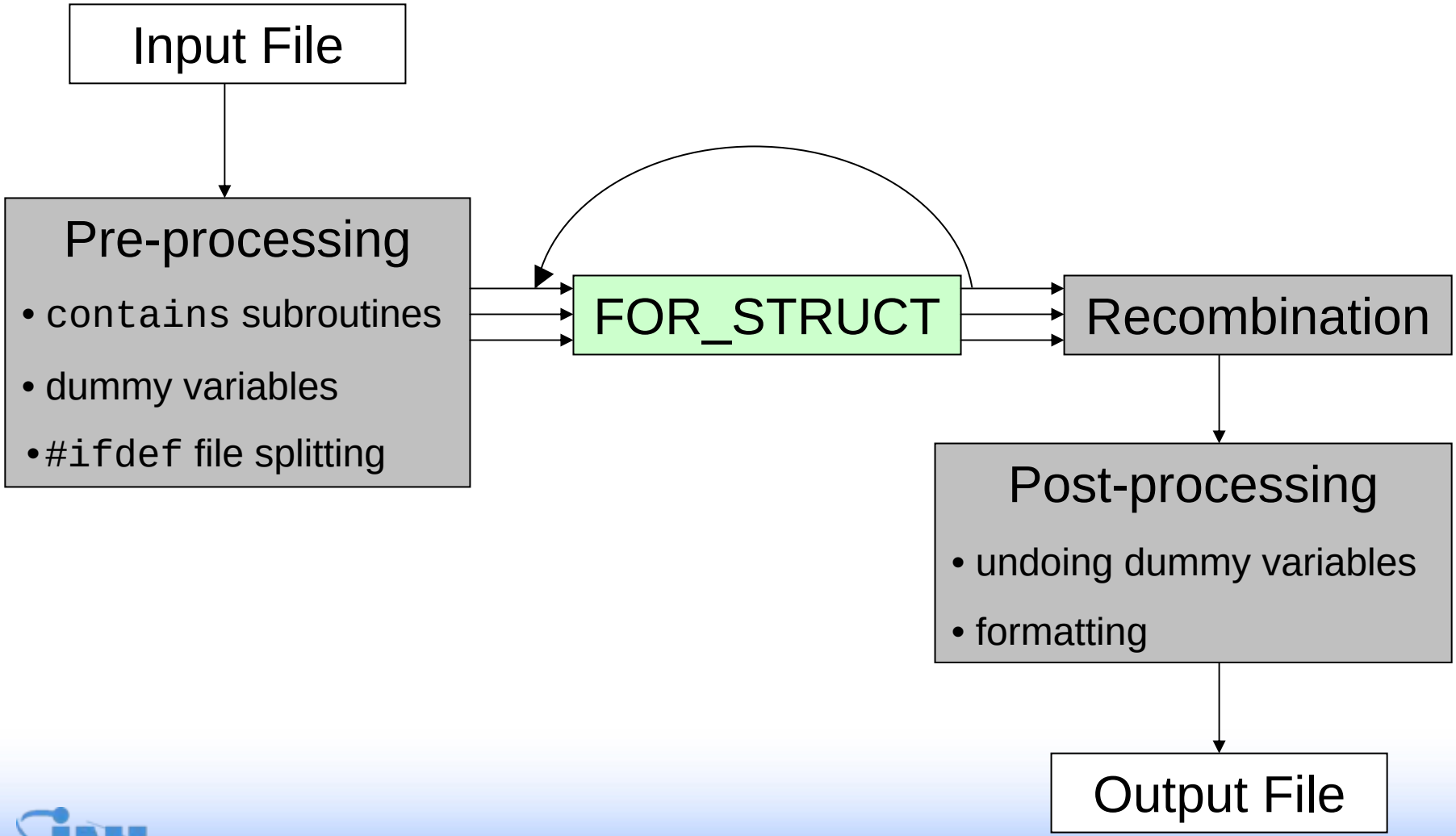
```
do...continue
go to
Computed go to
Arithmetic if
```

```
do…enddo
cycle & exit
do while…enddo
if then…elseif…else…endif
```

FOR_STRUCT Failings

- Fails to completely restructure convoluted code
- Converts case statements
- Cannot restructure with pre-compiler directives
- Crashes on many of the F90 syntax changes

Abridged Algorithm



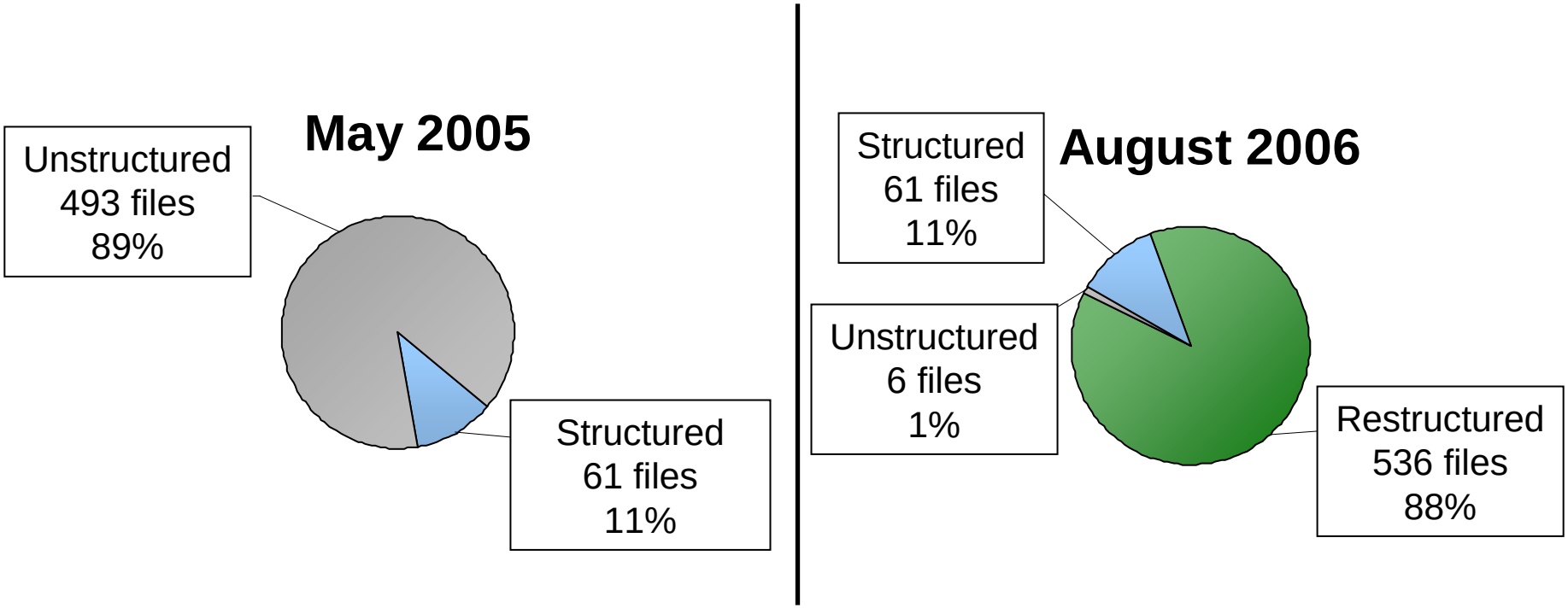
Scripting Solutions

- Prepare coding so that FS2.1.1 can handle it
  - Example:
    - Change derived-type variables (with %) to dummy variables.
    - After processing, substitute back to original.
- Pass multiple copies through FS2.1.1 to handle pre-compiler directives
  - Turn on different options for the different runs.
- Apply FS2.1.1 iteratively.

Manual Manipulations

- `contains` subroutines
  - F90 feature – no new specifications required
  - Improves modularity
  - Decreases chances of FS2.1.1 errors
  - Further increases high-level understanding of code
  - Limitation: subroutine must not include `go to` statements to other sections of the code
- Nomadic `endif`
  - FS2.1.1 tends to misplace `endif` or `enddo` statements when followed by a `#endif`

Results: Progress



Results: Complexity

- **McCabe Cyclomatic Complexity - the number of independent logic paths through the code.**

Mean values for the converted code:

McCabe Cyclomatic Index	**50.3**	⇧ **1.6**
Maximum Nested Depth	**13.7**	⇧ **5.0**
Average Nested Depth	**6.1**	⇧ **2.6**

Cursory Conclusions

- **Pre-compiler directives complicate restructuring.**
- **`Contains` subroutines increase modularity and simplify restructuring, with a few limitations.**
- **Restructuring has little effect on Cyclomatic complexity.**
- **Unix scripting is extremely powerful, but…**
- **There's no substitute for blood, sweat, & tears.**