Preprocessor, Take 2

/JOR

Why do people use preprocessing

- Adjusting external names for C interoperability
  - Largely obviated by C interop features in the language
- Platform/OS specific code
- Debug or other variants of code
- A crude way of implementing templates

What do people use today?

- C preprocessor (cpp) syntax
- Implementations vary
  - cpp with a "Fortran mode" switch
  - fpp Fortran-aware separate preprocessor
  - Implemented directly in the compiler

Didn't we do this already?

- Conditional Compilation ("CoCo"), optional Part 3 of Fortran 2003

```
?? IF (DEVELOPING) THEN
?? ! The following output statement was used when
?? ! developing the code
 WRITE(UNIT=*,FMT=*) 'The value of A is', A
?? END IF
```

- Universally ignored
- Withdrawn as of Fortran 2008

Let's try this again

- Goals for 202Y
    - Define cpp-style preprocessor in the standard
    - Ideally, most existing uses of preprocessing will "just work", or need minimal changes
    - Feature will not be optional, but implementations are encouraged to offer an option to "do it the old way"
    - "Minimum Viable Product" – don't try to do everything

Fundamental features

cpp directives and features:

- __LINE__ and __FILE__
- #line
- #ifdef and #ifndef
- #define and #undef
- #if
- #include
- #error
- ##operator (token concatenation without spaces)
- #operator (creates character literal from token)
- Macro expansion

Macro expansion

- Are tokens case-sensitive?
  - If FOO is defined as a macro, is foo replaced?
  - All existing implementations are case-sensitive
  - Many users explicitly use uppercase for macros and don't expect them to be case-insensitive
  - JOR recommends yes, tokens are case-sensitive

Macro expansion

- Are tokens in character literals replaced?
  - No tested implementations do this
  - JOR recommends no, they are not
- What about in Hollerith if processor supports them?
  - Majority of implementations do not do this
  - JOR recommends same behavior as with character literals

Macro expansion

- Recognition of tokens for macro replacement is not performed for:
  - *letter-spec-list* in `IMPLICIT`
  - Continuation character in column 6 of fixed-form source

Macro expansion

- Are tokens in OpenMP and other directives replaced?
    - All tested implementations do this
    - JOR recommends that directives are treated like statements

Decisions to be made

- Is // processed as concatenation or a C comment?
    - No tested implementations treat // as a comment
    - JOR recommends that // is concatenation

Decisions to be made

- Do `INCLUDE` files get preprocessed?
  - No tested implementations do this by default
  - But, if preprocessing is standard it should apply to INCLUDE files
  - JOR recommends that preprocessing applies to `INCLUDE` files
    - Standard should recommend processors provide an option to control this
    - Should also add a way to indicate it in source (see later)

Decisions to be made

- Is `!` the C "not" operator or a Fortran comment delimiter?
  - Tested implementations treat `!` as "not" in `#if and #elif` expressions, as a source character elsewhere, not a comment introducer in a preprocessor directive
  - JOR recommends to follow existing practice

Decisions to be made

- Are multiline comments bracketed by `/*   */` supported?
    - Some (all?) tested implementations do this
    - Large body of code examined contains no uses of this feature
    - JOR is uncertain

Decisions to be made

- What happens if macro expansion extends source line past standard limit (72/10,000)?
  - Most tested implementations generate continuation lines
  - JOR recommends that preprocessing generate continuation lines when appropriate

Decisions to be made

- Should lines that look like comments but are recognized as directives by the processor, such as OpenMP directives, be preprocessed?
- JOR recommends that the standard add a note saying that such lines are treated as if they were statements, including generating appropriate continuation

Decisions to be made

- In fixed-form source, are spaces embedded in tokens significant?
    - For example, are FOO and FO O treated the same?
    - No tested implementation ignores embedded spaces
    - JOR recommends that spaces in tokens are significant

What else might be useful?

- `GET_PROCEDURE_NAME()` intrinsic for the current program unit (often requested, not really preprocessor)
- `#pragma` directive to supply options – [no]freeform? –[no]include?
- What else?

Questions and comments?