To: J3 J3/23-124
From: R. Bleikamp & JOR
Subject: List and status of JOR's (potential) work items for F202y
Date: 2022-February-20

Reference: https://j3-fortran.org/forum/viewtopic.php?f=9&t=106&p=525#p525

Recent updates are in dark red.

Work item name	Status / recent activity	References
Pre-processor	Straw vote results below See next table line below for current plans:	See https://j3-fortran.org/doc/year/22/22-186.pdf (or newest revision thereof)

Straw vote results: (yes/no/undecided)
Is a very cpp like preprocessor acceptable (2/9/6)
Is a preprocessor mostly similar to existing fpp's acceptable (14/0/3)
Is a somewhat more Fortran friendly preprocessor acceptable (12/3/2)
Is an extremely Fortran friendly acceptable (5/5/7)

JoR is actively attempting to decide how Fortran friendly the preprocessor should be and pondering other high level decisions. Active investigations include:

1. Understand what is being used in real codes. Gary K. has analyzed many Fortran benchmarks/applications and has collected stats on which cpp directives are used (frequency), …
2. How to add **fpp** to the standard. Jon S. will be looking into this.
3. What are the implications of using Fortran tokens as the basis for token replacement, rather than C/C++ tokens? Lorri M. will be researching this.

Future steps:

4. Flesh out the potential requirement set, based both on what we learn above, and the features that have been proposed.
5. Flesh out the potential phases of the preprocessor, in a kind of functional programming way. That is, have some view of a shell-like pipeline: collecting-input | process-directives | expand-macros | reformat-output. Again, it doesn't have to be implemented this way, but it helps us describe the activities independently.
6. Map the requirement set to the phases.
7. Outline the consequences to the requirements to the implementation of the phases. There will be both plusses and minuses in terms of what the users get, and what the implementers have to do.
8. Begin documenting individual directives. Steve L. has volunteered to help with this.
9. Figure out if a looping construct is needed.

JOR will be soliciting input from J3 along the way. Volunteers may be needed too.

fpp Update: Feb 2023 – JoR has decided to proceed with a "somewhat more Fortran Friendly" approach. More or less adopting what existing existing fpp's do today, when the most Fortran friendly flags are used. This includes:

- Better handling of fixed form. Token expansion will not cause expanded lines to treat text beyond col 72 as commentary.
- Fortran tokens will be the recognized tokens for token replacement, not C language tokens. This makes the definition of the preprocessor easier. Case will be ignored when identifying tokens. This also implies insignificant blanks are ignored when scanning for tokens (fixed form).
- We believe this enhanced functionality will not adversely affect many users who are using a less Fortran friendly pre-processor now, and will provide a much more portable preprocessor that will be widely adopted by the user community. Comments are welcome. Send them to Lorri M.
- Gary K has gathered about 34 million lines of Fortran, with 400,000+ cpp-like preprocessor directives and is analyzing these codes.

Other Work items JoR will pursue	**Status / recent activity**	**References**
change F.P. model to be IEEE 754	Not started. Easy to do, low priority.	https://github.com/j3-fortran/fortran_proposals/issues/268
Remove some processor dependencies from Annex A	Not started.	This will be a low priority background task.
Immutable values	Not started.	https://github.com/j3-fortran/fortran_proposals/issues/221
scan/prefix sum	Brad Richardson is actively working this item.	https://github.com/j3-fortran/fortran_proposals/issues/273 Latest: https://j3-fortran.org/doc/year/23/23-113.txt
log2: just log2, or survey math.h and see what other base 2 intrinsics are missing from fortran.	Brad Richardson is pursuing this item. Not just base 2 intrinsics, but IEEE-754 more generally.	https://github.com/j3-fortran/fortran_proposals/issues/222 Van's paper 22-105 is subsumed by 23-111r1 https://j3-fortran.org/doc/year/23/23-111r1.txt

Work Items JoR is undecided about for inclusion in F202Y	Status / recent activity	References
ASSERT	Not started. Need to evaluate Magne's comments.	https://github.com/j3-fortran/fortran_p ... /issues/70; New details - viewtopic.php?f=9&t=113
scan clause for do concurrent reduce	Not started. Need use cases, and possibly a volunteer to drive this item.	https://github.com/j3-fortran/fortran_proposals/issues/224
Disallow use of specific new F202y features in a program unit that uses any deprecated/deleted features	JoR is undecided if this is a desirable feature. Leaning towards NOT pursuing this.	https://github.com/j3-fortran/fortran_proposals/issues/280

Work items adopted by other subgroups

Program specified default kinds for constants and intrinsic types.	Adopted by **DATA**

Work items that JoR is NOT planning on recommending (currently) for inclusion in F202y. Interested parties should contact JoR (rich@bleikamp.net) to arrange a time to present their views to the subgroup.

Work item name	Status / recent activity	References
Surprising results for UBOUND and LBOUND when arg has zero extent	JoR is leaning towards dropping this feature. A compelling use case would change our mind.	https://github.com/j3-fortran/fortran_proposals/issues/254
intrinsic to return the name of your caller, current procedure name, ...	JoR decided not to pursue this. Again, a compelling use case might change our mind. Overhead and possibly requiring debugging info is a concern. Seems like a companion processor (debugger) can do some of this.	https://github.com/j3-fortran/fortran_proposals/issues/180
Deprecate D format edit descriptor	the D edit descriptor serves no useful purpose anymore. But removing it from the standard may not be trivial.	https://github.com/j3-fortran/fortran_proposals/issues/226
constexpr	JoR needs to research this more. We want to know when C++ initializes constexprs. JoR would like to see a compelling use case. Seems expensive to implement if initialization happens at compile time. **Until JoR determines this is easier than we think to implement, this feature will remain in the Not Recommended catagory.**	https://github.com/j3-fortran/fortran_p ... issues/214https://fortran-lang.discourse.group/t/ ... fortranfan and from a DATA subgroup item https://github.com/j3-fortran/fortran_p ... issues/253
comments in list directed input	similar to namelist, but undelimited character input data may be a problem / incompatibility)	**Van's email description:** see the next table row.
	We allow comments in namelist input. In list-directed input, one can put comments after the last item desired by putting them after the slash that terminates the input. If one is reading several arrays, say one array per line, with one list-directed input statement, one cannot put a slash and comment on each line because that terminates the input. Would there be a problem to allow comments in list-directed input, beginning with "!" as in namelist input? JOR may reconsider if no backwards compatibility issues exist.	