WG 14 N1844

2014/06/12, 12:00 EDT, 9:00 PDT:

Attendees: Rajan, Fred, Marius, David, Jim, Ian, Mike

New agenda items:
None.

Old action items:
David: Part 5: (From last meeting): Complete exception specification with the full syntax dealing with scope and sub-exceptions. Include a discussion document with reasons choices and alternatives. - Partially done (more of an outline. Sent on 2014/05/12). Keep open.
David: Part 5: For fast_subnormal, change to "allows abrupt underflow" -> "allows (but not requires) abrupt underflow". Done.
David: Part 5: Put in exception category in the sub-exceptions as the prefix. Ex. FE_DIV_ZERO -> FE_DIVBYZERO_DIVIDES. Done.
David: Part 5: Add in FE_INVALID_OTHER and FE_DIVBYZERO_OTHER. Done.
David: Part 5: OPTFLAG -> MAY_FLAG, NOFLAG -> NO_FLAG, SUBSTITUTEXOR -> SUBSTITUTE_XOR. Done first two, last is pending issue resolution.

Next meeting:
July 15th (Tuesday), 2014, 12:00 EDT, 9:00 PDT
Same teleconference number.

New action items:
David: Syntax.txt: Add in the beginning something that gives the purpose of the document. Ex. The CFP group is asking for feedback from WG14 for ...
David: Syntax.txt: Semantics: Change exception1/exception2 to exceptions1/exceptions2
David: Syntax.txt: Add to the end of the document other ideas considered.
David: Syntax.txt: Add a sentence to handle thread and object state considerations.
David: Syntax.txt: Add a sentence about ASAP vs deferred exception handling (try-catch vs try-patch).

Discussion:
Part 1: Reviewing draft as edited by ISO. Jim has sent back comments on changes made.

Part 2: DTS ballot issued.

Part 3: PDTS ballot issued.

Part 4: PDTS ballot issued.

Part 5: (Email discussion based) (http://www.validlab.com/cfp/*.txt)
http://www.validlab.com/cfp/syntax.txt
*ToDo: David: Add in the beginning something that gives the purpose of the document. Ex. The CFP study group is asking for feedback from WG14 for ...
*ToDo: David: Semantics: Change exception1/exception2 to exceptions1/exceptions2
Should we add the other things we considered like:
1) PL/I, Basic and other languages "ON event DO/GOTO/GOSUB handler"
2) Exception handler callback function registration (like signal handlers)
3) Using SIGFPE signal handling

- 4) Testing flags with the existing C floating point environment handling functions and adding in new flags like underflow at the end of the document? Yes
- *ToDo: David: Add to the end of the document other ideas considered.
- Mike: Avoiding extra nesting is advantageous if we don't do try/catch.
- Fred: How does what we do work with threads and object state?
  - Rajan: Should be written/stored to variables in the scope of the exception handling be in an indeterminate state only?
  - Rajan: Also side effects are not known in sequencing.
  - Yes, we should cover both cases for object state.
  - For threads, this (exception handling) should be thread local like the existing C Standard floating point environment being thread local.
  - Fred: Signal handling in a multi-threaded program results in undefined behaviour.
- David will make the changes discussed and post it for review before sending it out to the wider WG14 group.

Substitution (pre/xor):
- David: Haven't looked at pre-substitution since it is cheaper to do exception/test-and-branch.
  - Ex. Instead of "SUBSTITUTE(z = x / y, DIV_ZERO, z = x' / y')" do "try { z = x / y; } catch (DIV_ZERO) { z = x' / y' }"
- Easier to get one set of changes made rather than new syntax for each part with their attendant issues and potential problems. This means the exception handling approach is better.

Raise-no-flag:
- David: Requires trapping, otherwise flags would be raised.
- Shows that the programmer has explicitly thought about the flags. i.e. opt-out vs opt-in
- Can be done with a pragma or try/catch (if catch does not raise a flag for the caught exception).

We can document how to use the exception handling (ex. try-catch) to handle a lot of these items (substitution, may-flag, raise-no-flag, etc.).

Should we have two versions (asap-try-catch vs try-catch)? An ASAP and a deferred one?
- Perhaps have different catch's? catch vs deferred_catch/patch or a #pragma that identifies the following code as ASAP or not.
- Loops (indeterminate or large limits) may want to do asap while smaller ones may want deferred. Should we give the compiler hints or directives for this?
- We should do both mechanisms since not doing asap means you can use the flag checking instead.
- catch can be implemented in patch, except for exact underflow

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative for Canada