WG 14 N1849

2014/07/15, 9:00 PDT / 12:00 EDT:
Attendees: Rajan, Jim, David, Fred, Mike, Ian

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: SUBSTITUTEXOR -> SUBSTITUTE_XOR. Pending issue resolution. - Leave open
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 ... - Done
David: Syntax.txt: Semantics: Change exception1/exception2 to exceptions1/exceptions2 - Done
David: Syntax.txt: Add to the end of the document other ideas considered. - Done
David: Syntax.txt: Add a sentence to handle thread and object state considerations. - Done
David: Syntax.txt: Add a sentence about ASAP vs deferred exception handling (try-catch vs try-patch). - Done

Next Meeting:
August 14th (Thursday), 2014, 12:00 EST, 9:00 PDT
Same teleconference number.

New action items:
Rajan: Check part 2 ballot status (at least for Canada) - Done: Vote closed on June 2nd for Canada. Canada voted approve with no comments.
David: Survey the C standard to see how much would need to change to support try/catch.
David: Talk to Douglas to see if there are other concerns with try/catch.
All: Look for another form for attributes to code other than try/catch or pragmas.

Discussion:
Part 1: Supposed to be published today.

Part 2: *ToDo: Rajan: Check on status when internet access is available.

Part 3: PDTS ballot issued. US vote: 12/0/0

Part 4: PDTS ballot issued. US vote: 12/0/0

Jim is making changes to parts 2-4 to match what ISO changed for part 1.

Part 5: (Email discussion based) (http://www.validlab.com/cfp/*.txt)
http://www.validlab.com/cfp/syntax.txt
Blaines note: Any tests and branches will result in a lot of rework for one pass compilers.
Expected that setting and handling traps will be the common implementation.
WG14 comment summary:
Don't like pragma.
Don't want try/catch unless it is the same as C++.

FE exception handler.
David: Should be useful for optimization for making the common case fast, and only check the exceptional cases.
David: Note that this would help a lot for R's underlying implementation helping big data.
David: Exception handler clutters up the normal case.
Go with fe_try/fe_catch allowing fe_patch, fe_substitute with changes for one pass compilers.
Note: We would need to go through the C standard to find all the places to put in the syntax.
ToDo: David to survey the C standard to see how much would need to change to support try/catch.
ToDo: David to talk to Douglas to see if there are other concerns with try/catch.

Should automatic exception handling propagate to called functions? For example, rounding modes are not for externally compiled CU's.
We could make it undefined, propagate it, or not propagate it.
Implementation wise, with hardware traps, propagating makes sense. For tests, not propagating makes sense.
For CU's where you don't have the code and hence can't recompile, it can be impossible to set the tests.
Fits in with the rounding directions as well if we make it not propagate.
Can we have a function attribute type thing that allows the function to be called with and without propagating contexts.
This would be another complication to add, so maybe we can make it an extension or as a possible future enahncement.
Can FE_ENV access pragma's handle this?
Does not seem to.

Precedence:
try/catch can nest and work with parent blocks.
pragma can be file scope and try/catch can override it for example.
Outer level can be for all exceptions, inner can be only for invalid, while another inner (or sibling) could be for divide by zero.
For exception vs subexception, the more specific one should have precendence and would be the only one executed. No throws allowed.

Abrupt underflow and presubstition do not apply for catch blocks. Note that we still need to work with optimization and rounding modes and other things. More than a try, it is instead special handling code.
Currently this all fits in with pragmas but it seems WG14 doesn't like them.

Note that the try/catch can be converted to pragmas or vice versa rather than having both. Or a third type of syntax with similar semantics.

ToDo: All: Look for another form for attributes to code other than try/catch or pragmas.

Implementation:
The Sun model is good reference for implementation but not for users.
Sub-exception can be tests while exceptions can be traps with possibly the trap handler checking for sub-exceptions. Motorola and ppc did split the invalid exceptions up, but this is likely not common amongst other implementations (ex. x86).
Can set a side structure that is updated per operation, and then the handler can look up the sub-exception in that structure.
Some are function based so the opcode is not enough.
Is the next instruction or other information available to trap handlers?

Normally the register image is given to the handler. Can walk up the stack to see __func__ for the function name to determine if it is one of those sub-exceptions.

Deferred would almost have to be flags and hence high level exceptions not necessarily sub-exceptions.

This can be implementation defined (the sub-exception support).

Regards,

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