WG 14 N1858

2014/08/14, 9:00 AM PDT / 12:00 PM EDT:
Attendees: Rajan, Jim, David, Fred, Marius, Ian

New agenda items:
Jacob Navia's note - Added

Sticky/Keep open/Long term action items:
David: Part 5: 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

Lasts meeting action items:
Rajan: Check part 2 ballot status (at least for Canada). - Done (Ballot passed)
David: Survey the C++ standard to see how much would need to change to support try/catch. - Done (Note sent on 2014/07/25)
David: Talk to Douglas to see if there are other concerns with try/catch. - Not done. Close the item due to discussion to happen regarding part 5 today.
All: Look for another form for attributes to code other than try/catch or pragmas. - Done

New action items:
Rajan: Draft a response to Jacob Navia's note and send it to the CFP group before sending it to the wider WG14 group.
David: Part 5: Prepare a complete specification for alternate exception handling by the end of August.

Next Meeting:
September 16th (Tuesday), 2014, 12:00 EST, 9:00 PDT
Same teleconference number.

Discussion:
Part 1: Published.

Part 2: DTS ballot is done. Moving to publish to send to ISO for their ISO format edits.

Part 3: PDTS ballot issued.

Part 4: PDTS ballot issued.

Jacob Navia's note on 2014/08/14 4:34AM CST:
*ToDo: Rajan: Draft a response to Jacob Navia's note and send it to the CFP group before sending it to the wider WG14 group.

Part 5: (Email discussion based) (http://www.validlab.com/cfp/*.txt)
Based on the proposal of going forward with #pragma's.
#pragma's allow the optimization and other aspects as well so a single solution form.
Problem of scope of expressions in the #pragma.
The scope is for identifiers and area of application of the pragma (like the static rounding mode).

Ex. goto or substitution for an exception may have different scopes (exception area vs identifier location).

Jim: Similar to the function issue with the static rounding mode.

David: Nested blocks could have overridden a label (or variable) name where the exception could happen in the parent or child block.

The substitution or exception can refer to the variable, but not knowing which one.

Rajan: The jumped to location should follow the standard C scoping rules so it would only refer to one specific object.

The pragma can refer to variables declared before or after the pragma.

Presubstitution: "In this block, make this presubstitution." The variables can be before or after the #pragma.

Regular C requires it to be after the declaration to know what the variable is.

Can require nested block after all the declarations with the pragma as the first statement.

For example f1 and g1 are calculated already somewhere (perhaps before the pragma, or perhaps inside since they change per iteration)

f1/g1 are substituted in place of f/g if g = 0.

It can be done either way.

Implicit blocking or explicit? Explicit makes it easier for the implementors and possibly clearer for the programmer.

Consensus: Require the #pragma right after the opening curly brace. This means the variables it refers to must have been declared before the block immediately containing the #pragma.

Labels: Like goto's, one pass compilers already have to handle it so it would not be more complicated work.

Can treat jumping to the label it as if it was an explicit goto.

Functions and variables have to be declared before use in C but labels do not. This will not be new if we treat our FP labels the same as the goto labels.

Goto type label scope is function scope. It can be referred to anywhere in the function (even before it's implicit declaration).

Alternatives to try/catch/pragma:

In the reflector discussion, the goto style trap handler set a bit and the programmer tests that bit.

It is the same as the flags with including the sub-expressions and exact underflows with new functions to check them.

David: This removes the ASAP property.

Jim: Can still do this through a pragma which breaks to the checking block upon any exception.

David: The save/clear/test/restore requirement is still there. Don't want to drown the normal case.

Rajan: Wasn't the discussion with the macro's to hide the complexity?

It won't work that well with macros. Nested macros have problems too. Handling the exceptions cause the issues.

Should we put this in a committee draft form now for the mailing?

It can help, but we may not be ready.

Concerns about tying our hands for future changes.

Possible better responses from WG14 as a proposal.

*ToDo: Part 5: David: Prepare a complete specification for alternate exception handling by the end of August.

Reformat after the September teleconference for submission to the WG14 mailing.

Should we make the remainder (optimization, etc.) as part 6?

Better if we don't since we have approval for 5 parts and we have listed it as 5 parts in the existing earlier parts including published ones.

We can put in blank headed sections for the remaining parts of part 5 so all of WG14

knows they are coming.

Subexceptions:

If no feature test macro is there for a subexception, then it is not supported.

Fear is making it optional means it may never be implemented.

Will part 5 be more likely to be implemented if the individual subexceptions are optional?

It may be easy on some hardware for some of the subexceptions so they may just do those.

Even if it is required, implementers may still do most of part 5 and mention they are not conforming on aspects x, y, z.

Consensus: Require all the subexceptions.

ASAP vs deferred:

Can allow the implementation to choose which way.

Some times you definitely want deferred, so you need to be able to choose.

Regards,

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