EDITORIAL CHANGES CONCERNING EXCEPTIONS

James W. Welch
WATCOM International Corp.

jww@watcom.on.ca

Doc. No. WG21/N0779, X3J16/95-0179

Abstract

This paper suggests editorial changes to the Clause 15 [except], to Clause 18 [lib.language.support], and to Clause 19 [lib.diagnostics]. The intent is clarify that exception-processing can be “stacked” and to clarify that exceptions can be thrown in exceptional circumstances, as long as they are completely handled. I do not consider any of the proposals to be substantive. A formal paper has been written in case the committee chooses to consider the changes to be substantive.

Overview

The current draft is imprecise in that it often says that particular actions are performed when a function throws an exception. The intent is rather to perform the corresponding actions when an exception is both thrown and is not handled in the dynamic context of the function activation.

For example, the current draft says that the unexpected() function is called when a function throws an exception that is not listed in the exception specification for the function. The draft should say that the unexpected() function is called when that function attempts to exit via an uncaught exception whose type is not listed in the exception-specification.

Proposals

Add new paragraphs at the end of section 15.1 [except.throw]:

At certain times there may be more than one exception that has been thrown and not yet finished. Dynamically nested try blocks may be executed while a handler has not yet completed its execution.

A function activation is said to exit via an uncaught exception when an exception is thrown and not handled by a handler in the dynamic context of that activation. The exception may be a new one resulting from a throw or may be an unfinished exception that has been rethrown. In certain circumstances (see 15.4, 15.5), a function can be activated that is not permitted to exit via an uncaught exception.
Add a new sentence at the end of paragraph 1 in section 15.2 [except.ctor]:

Similarly, any temporary copies of exceptions that are finished are also destructed.

Change sentence 1 in paragraph 10 in 15.4 [except.spec]:

from: The function unexpected() may throw an exception ...
to: The function unexpected() may exit via an uncaught exception ...

Change paragraph 12 in 15.4 [except.spec] to read:

A function with no exception-specification is allowed to exit via any uncaught exception. A function with an empty exception specification is not allowed to exit via any uncaught exception.

Add a paragraph to the end of 15.5.1 [except.terminate]

The terminate() function may not exit via an uncaught exception.

Change sentence 1 in 15.5.2 [except.unexpected] to read:

From: ... throws an exception ...
To: ... exits with an uncaught exception ...

Change sentence paragraph 2 in 15.5.2 [except.unexpected] to read:

The unexpected() function shall not return, but it may exit via an uncaught exception. If that exception is allowed by the exception specification that was previously violated, then the search for a new handler will continue at the call of the function whose exception specification was violated. Otherwise, if the exception specification does not include the name of the predefined exception bad_exception, then terminate() is called. Otherwise, an object of predefined type bad_exception is constructed and the search will continue at the call of the function whose exception specification was violated.

Note: the replacement criteria that was previously present in this paragraph has been omitted since it was incomplete (it did not handle a second exception thrown by unexpected()) and because the destruction is now handled by the new sentence in 15.2.

Change paragraph 3 in 15.5.2 [except.unexpected] to read:

Thus, an exception specification guarantees that the function will exit via an uncaught exception whose type is in the specification. If the function exception specification includes the name bad_exception then an attempt to exit via an
uncaught exception whose type is not in that list may cause an object of 
bad_exception to be instead used in the subsequent search for a handler.

Change the first sentence in 17.3.4.8 [lib.res.on.exception.handling] to read:

   Any of the functions defined in the C++ Standard Library can report a failure by 
   exiting via an uncaught exception of the type(s) described in their Throws: 
   paragraph and/or their exception specification (15.4).

Change the second point in the Required Behavior subclause in section 18.4.2.2 
[lib.new.handler] to read:

   - exit via an uncaught exception of type bad_alloc or a class derived from 
     bad_alloc

Change the first two points in the Required Behavior subclause in section 18.6.1.2 
[lib.unexpected.handler] to read:

   - exit via an uncaught exception that satisfies the exception specification
   - exit via a bad_exception exception

Change the first sentence in 18.6.1.4 [lib.unexpected] to read:

   - Called by the implementation when a function attempts to exit via an uncaught 
     exception whose type is not listed in the exception specification

Change three Notes subclauses in 19.1.1 to read as follows:

   From: Notes: does not throw any exceptions
   To:    Notes: does not exit via an uncaught exception

Conclusions

This paper attempts to make more precise some of the language in the indicated 
subclauses. Consequently, a reader can conclude that the exceptions can be thrown and 
handled (without exiting via an uncaught exception) by the constructor used to initialize 
the memory for an exception, by a destructor which is destroying that memory, and by 
handler functions.