Doc No : X3J16/95-0092 WG21/N0692
Date : 19.05.95
Project : Programmming Language C++
Ref.Doc : X3J16/95-0093 WG21/N0693
Reply to : Manfred Lichtmannegger
manfred.lichtmannegger@mch.sni.de

Title: Predicate throwing
==========

Problem:

To use exceptions to report failures of destructors and to avoid termination by (not nested) double failures/exceptions in a portable way.

Discussion:

Following 15.5.1 [except.terminate] the function terminate is called during throwing an exception, when a second (uncaught) exception is thrown.

This critical phase begins after the construction of the thrown exception, includes the (direct) destructor calls through stack unwinding and ends with a possible call of a copy constructor by the matching catch clause.

Termination is not adequate for a high available system. You can avoid it, if you guarantee that no uncaught exception is by an exception copy constructor and inside destructors. As a consequence no destructor or function or method called in a destructor should throw an exception which is not caught by the destructor.

If you use the common technique "resource aquisition is initialization" you can have failures in destructors - for instance by flushing buffers (or update in database in my original problem). As destructors are called implicitly, throwing exceptions seems to be more adequate to report such rare failures then to check a global error variable after closing a block, after calling a function (destruction of call-by-value-arguments) or calling delete.

There are idioms to handle this problems: Increment a counter before throw and decrement the counter in each catch clause, to check whether you are already in the process of throwing. If the counter is not zero instead of throwing an exception with disastrous consequences queue this exception and unqueue it in the catch clause which handles the first exception.

This is only a sketch of such an idiom, more elaborated variants are possible. But the idiom breaks if not all throws and catches are conforming. If you work with a third-party-library using exceptions you are again in danger.

Solution:

Add a predicate throwing (*) to the language support library, which returns true if you are in the critical phase of throwing, i.e. starting after the evaluation of the throw expression, ending after the initialization the matching catch clause.

(*) Alternative names proposed on the reflector are in_throw (rtm), is_it_safe_to_throw (krk), stack_unwinding (lichtman).

Classification:

It is a small and simple "extension", cheap in terms of implementation and runtime costs, which offers the programmer an important hook.

(I would classify it as a "must have" as the problem "To use exceptions to report failures of destructors and to avoid termination by double failures/exceptions in a portable way" is a real problem.)

Relations:

Document X3J16/95-0093 WG21/N0693 addresses the issue of (re)throwing pending errors, which is a nice-to-have. As you always can introduce rethrowing guards it is not really essential. The predicate throwing instead is an essential hook/lever to avoid (not nested) double exceptions.

Proposal:
---

1) add footnote

15.5.1 The terminate() function [except.terminate]

In the following situations exception handling must be abandoned for less subtle error handling techniques:

- when a exception handling mechanism, after completing evaluation of the object to be thrown but before completing the initialization of the exception-declaration in the matching handler XX) , calls a user function that exits via an uncaught exception,

XX) i.e. when throwing (18.6.3) returns true

2) add paragraph

15.5.3 The throwing() function [except.throwing]

The predicate

bool throwing()

returns true after completing evaluation of the object to be thrown until completing the initialization of the exception-declaration in the matching handler. This includes the stack unwinding.

3) add paragraph

18.6.3 throwing [lib.throwing]

bool throwing()

Returns: true after completing evaluation of the object to be thrown until completing the initialization of the exception-declaration in the matching.

Notes: When throwing is true throwing an exception can result in a call of

terminate (15.5.1).