Proposal for C23
WG14 N 2809

Title:	Annex K Repairs
Author, affiliation:	Robert C. Seacord, NCC Group
Date:	2021-10-4
Proposal category:	Defect
Target audience:	Implementers
Abstract:	Fix `set_constraint_handler_s` in multithreaded environments
Prior art:	C

Annex K Repairs

Reply-to: Robert C. Seacord (rcseacord@gmail.com)
Document No: **N 2809**
Reference Documents: N1173, N1969, N2225, N2336
Date: 2021-10-4

Two papers discuss the cons (*Updated Field Experiences with Annex K -- Bounds Checking* [N1969]) and pros (*Bounds-checking Interfaces: Field Experience and Future Directions* [N2336]) of retaining Annex K. The committee has been evenly divided on two occasions about removing Annex K. As there is no consensus to remove it, it is now necessary to repair it.

Change Log

2021-10-4:

- Initial version (**N 2810**)

Introduction and Rationale

Annex K originated as ISO/IEC TR 24731-1 — Extensions to the C library — Part 1: Bounds-checking interfaces, published in 2007 [ISO/IEC TR 24731-1]. The technical report was incorporated into C11 with only minimal changes. In retrospect, these changes were too minimal because too little consideration was given to the introduction of threads in this version of the standard. As a result, the behavior of Annex K in a multithreaded environment is undefined.

As described in N1866 [N1866], the `set_constraint_handler_s` function is specified to set the global process-wide runtime-constraint handler and return a pointer to the previously registered process-wide handler.

The default handler in place until the first call to `set_constraint_handler_s` with a non-null argument is implementation-defined.

One of the intended uses of `set_constraint_handler_s` documented in the TR 24731 Rationale [N1173] is to let programmers temporarily replace the registered handler with `ignore_handler_s` that does nothing and simply returns to the caller (or a user- defined handler), handle runtime constraint checking in their own code, and restore the original handler. This use case is particularly important in code that must avoid the risk of the constraint handler terminating the program, potentially abnormally.

However, because the process-wide handler is shared among all threads in a program, this use case isn't possible in multithreaded programs where code (possibly isolated functions or entire libraries often written by different teams of programmers with little knowledge of each other's implementation details) run in separate threads.

Martin Sebor proposed a N1962 2015/09/24 Sebor, Follow-up on N1866 Thread Safety of set_constraint_handler_s [N1962]. Although this solution had committee support, it was temporarily abandoned when Martin suggested eliminating Annex K altogether [N1969]. The solution described in N1969 solves the problem in a portable manner but relies on some invention (not unlike ISO/IEC TR 24731-1). This solution is proposed as "Proposed Wording 1". A competing solution based on *Thread-local state for getenv, strtok, and set_constraint handler* [N2225] is proposed as "Proposed Wording 2". This solution is less portable but relies less on invention.

These are competing proposals and, at most, only one of these two proposed wordings should be adopted.

Microsoft

Annex K and ISO/IEC TR 24731-1 before it is based on the Microsoft implementation. Unfortunately, significant changes were made to the Microsoft implementation during the standardization process, making the existing Microsoft implementation nonconforming to the standard. One area in which there were significant changes was in the runtime constraint handlers.

The current Microsoft implementation of these functions uses an invalid parameter handler. The Microsoft solution is analogous to wording proposal 1 and the invention required to create a thread local version of the set constraint handler function from the existing implementation is roughly the same as what was required to invent the existing `set_constraint_handler_s`:

https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/get-invalid-parameter-handler-get-thread-local-invalid-parameter-handler?view=msvc-160

```
_invalid_parameter_handler _get_invalid_parameter_handler();
```

```
_invalid_parameter_handler _get_thread_local_invalid_parameter_handler();
```

https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/set-invalid-parameter-handler-set-thread-local-invalid-parameter-handler?view=msvc-160

```
_invalid_parameter_handler
_set_invalid_parameter_handler(_invalid_parameter_handler);
```

```
_invalid_parameter_handler
_set_thread_local_invalid_parameter_handler(_invalid_parameter_handler);
```

https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/invalid-parameter-functions?view=msvc-160

```
void _invalid_parameter(const wchar_t * expression, const wchar_t * function_name,
const wchar_t * file_name, unsigned int line_number, uintptr_t);
```

```
void _invalid_parameter_noinfo();
```

```
void _invalid_parameter_noinfo_noreturn();
```

```
void _invoke_watson(const wchar_t * expression, const wchar_t * function_name,
const wchar_t * file_name, unsigned int line_number, uintptr_t);
```

Proposed Wording 1

The wording proposed is a diff from WG14 N2596. Green text is new text, while ~~red text~~ is deleted text.

Add to the end of §B.21 General utilities <stdlib.h>, a declaration of a function thrd_set_constraint_handler_s as indicated below.

```
constraint_handler_t set_constraint_handler_s(constraint_handler_t handler);
```

```
constraint_handler_t thrd_set_constraint_handler_s(constraint_handler_t handler);
```

Change §K.3.1.4 Runtime-constraint violations, paragraph 2, as follows:

If a runtime-constraint is violated, the implementation shall call the currently registered runtime-constraint handler (see `set_constraint_handler_s` and `thrd_set_constraint_handler_s` in `<stdlib.h>`).

Change the Description of the set_constraint_handler_s function in §K.3.6.1.1, paragraph 2, as follows:

The `set_constraint_handler_s` function sets the runtime-constraint handler to be handler for the calling thread and any threads yet to be created by it, for every other existing thread in the program that has not yet made a call to any function defined in Annex K or has been created by a thread that has made such a call, and for all threads that are yet to be created by such threads. The runtime-constraint handler is the function to be called when a library function invoked by the same thread detects a runtime-constraint violation. Only the most recent handler registered ~~with~~ by a call to `set_constraint_handler_s` or `thrd_set_constraint_handler_s` is called when a runtime-constraint violation occurs.

Change §K.3.6.1.1, paragraph 4, as follows:

The implementation-defined ~~has a~~ default runtime-constraint handler ~~that~~ is used if ~~no calls to~~ it has not been replaced by a call to the `set_constraint_handler_s` or `thrd_set_constraint_handler_s` function ~~have been made~~.

Add a new section titled The thrd_set_constraint_handler_s function, immediately following §K.3.6.1.1, with the following text:

K.3.6.1.2 The thrd_set_constraint_handler_s function

Synopsis

```
#define __STDC_WANT_LIB_EXT1__ 1

#include <stdlib.h>

constraint_handler_t thrd_set_constraint_handler_s(constraint_handler_t handler);
```

Description

The `thrd_set_constraint_handler_s` function behaves the same way as the `set_constraint_handler_s` function except that it sets the runtime-constraint handler to handler only for the calling thread and for any threads that are yet to be created by the calling thread. The function has no effect on other threads in the program.Footnote) The remaining effects of the two functions are identical, as are their return values.

Footnote) The function is required to be provided even by implementations that define the `__STDC_NO_THREADS__` macro to 1.

Change Footnote 441 in §K.3.6.2.1, paragraph 2 as follows:

If the runtime-constraint handler for a thread is set to the `ignore_handler_s` function, any library function call in which a runtime-constraint violation occurs will return to its caller.

Proposed Wording 2

In K.3.6.1.1 (The set_constraint_handler_s function), add:

Only the most recent handler registered with `set_constraint_handler_s` is called when a runtime-constraint violation occurs. The registered constraint handler has thread storage duration. The registered constraint handler for a newly created thread shall be the same as the registered constraint handler of the current thread at the time of creation.

4.0 Acknowledgements

I would like to recognize the following people for their help with this work: David Keaton, Martin Sebor, and Aaron Ballman.

5.0 References

[ISO/IEC TR 24731-1] ISO/IEC TR 24731-1:2007 Information technology — Programming languages, their environments and system software interfaces — Extensions to the C library — Part 1: Bounds-checking interfaces. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=38841

[N1173] N1173 2006/04/18 Meyers, Rationale for TR 24731. URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1173.pdf

[N1962] N1962 2015/09/24 Sebor, Follow-up on N1866 Thread Safety of set_constraint_handler_s. URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1962.htm

[N1969] N1969 2015/09/30 Sebor, Updated Field Experiences With Annex K -- Bounds Checking. URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1969.htm

[N2225] N2225 2018/03/26 Sebor, Thread-local state for getenv, strtok, and set_constraint handler. URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2225.htm

[N2336] N2336 2019/02/03 Seacord, Bounds-checking Interfaces: Field Experience and Future Directions. URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2336.pdf