Proposed Encoding Error Behavior for the `sprintf_s` Function

David Keaton

2005-09-27

1. Introduction

1.1 Purpose

Currently, **`sprintf_s`** returns zero in all error cases. This document proposes improving the ability to automatically remediate code by changing this behavior to return a negative value if and only if **`sprintf`** would have returned a negative value, and zero in all other error cases.

1.2 Scope

This document presents a revision to WG14 Technical Report 24731, *C Library Extension 1*, and follows all rules and guidelines of that technical report except where explicitly noted herein.

1.3 References

1. WG14/N1135, Technical Report 24731, *C Library Extension 1*.
2. ISO/IEC 9899:1999+TC1+TC2, *Programming Languages—C*.

1.4 Rationale

1.4.1 Existing Code

The **`sprintf_s`** function was introduced to support automatic remediation of code that calls **`sprintf`**.

One of the idioms to be remediated looks like the following. It has been observed to occur large numbers of times in millions of lines of embedded code.

```
count = sprintf(buf, format_string, args);
count += sprintf(buf + count, format_string, args);
```

This code appears in environments where there cannot be encoding errors, either because

the locales do not have invalid character values (such as 8-bit-only character sets), or because the only characters used are ones that cannot create encoding errors in any locale, or because of some other favorable combination of characters and locale. Otherwise, **`sprintf`** could return a negative value and break this code. Note that for drop-in replacement, **`sprintf_s`** must not return a negative value in any situation where **`sprintf`** would not have done so.

Because of idioms like the one above, the most robust way to handle most errors is to write a null character into the first position of the buffer and return zero. This keeps the character count from changing to a possibly nonsensical value before it is used again by the next **`sprintf`** call in the sequence.

However, in environments where an encoding error can occur, the following code is found.

```
count = sprintf(buf, format_string, args);
if (count < 0) {
     // An encoding error occurred.
}
```

Therefore, for drop-in replacement, **`sprintf_s`** must return a negative value in cases where **`sprintf`** would have done so.

The goal is to pick a behavior for **`sprintf_s`** that permits automatic remediation of both types of usage for **`sprintf`**.

1.4.2 Existing TR Behavior

As specified in the TR, **`sprintf_s`** returns zero for all error conditions. This allows drop-in replacement of **`sprintf`** for the first idiom above, but not the second.

1.4.3 New Behavior

The proposed new behavior for **`sprintf_s`** is to return a negative value upon encountering an encoding error. This is the only case where **`sprintf`** can return a negative value as specified in the C standard.

1.4.4 Changes Required by Implication

The **vsprintf_s**, **swprintf_s**, and **vswprintf_s** functions are analogous and need to be changed in the same way.

The wide character functions **swprintf** and **vswprintf** return a negative value in one additional case, where the buffer is too small to hold the result. Therefore, to remain consistent with the principle of having a drop-in replacement, **swprintf_s** and **vswprintf_s** need to return a negative value in this case as well.

1.4.5 Revision History

Revision 1: Inverted the order of the first sentence of proposed edits, to put the "if" clause first.

2. Edits to the TR

The TR is proposed to be changed as follows.

In section 6.5.3.6, replace paragraph 6 with the following.

> If no runtime-constraint violation occurred, the **sprintf_s** function returns the number of characters written in the array, not counting the terminating null character. If an encoding error occurred, **sprintf_s** returns a negative value. If any other runtime-constraint violation occurred, **sprintf_s** returns zero.

In section 6.5.3.13, replace paragraph 6 with the following.

> If no runtime-constraint violation occurred, the **vsprintf_s** function returns the number of characters written in the array, not counting the terminating null character. If an encoding error occurred, **vsprintf_s** returns a negative value. If any other runtime-constraint violation occurred, **vsprintf_s** returns zero.

In section 6.9.1.4, replace paragraph 6 with the following.

> If no runtime-constraint violation occurred, the **swprintf_s** function returns the number of wide characters written in the array, not counting the terminating null wide

character. If an encoding error occurred or if **n** or more wide characters are requested to be written, **swprintf_s** returns a negative value. If any other runtime-constraint violation occurred, **swprintf_s** returns zero.

In section 6.9.1.9, replace paragraph 6 with the following.

If no runtime-constraint violation occurred, the **vswprintf_s** function returns the number of wide characters written in the array, not counting the terminating null wide character. If an encoding error occurred or if **n** or more wide characters are requested to be written, **vswprintf_s** returns a negative value. If any other runtime-constraint violation occurred, **vswprintf_s** returns zero.