LoadString can load strings with embedded nulls, but your wrapper function might not

devblogs.microsoft.com/oldnewthing/20091009-00

October 9, 2009

Raymond Chen

Whenever somebody reports that the `SHFileOperation` function or the `lpstrFilter` member of the `OPENFILENAME` structure is not working, my psychic powers tell me that they failed to manage the double-null-terminated strings.

Since string resources take the form of a counted string, they can contain embedded null characters, since the null character is not being used as the string terminator. The `LoadString` function knows about this, but other functions might not.

Here's one example:

```
TCHAR szFilters[80];
strcpy_s(szFilters, 80, "Text files\0*.txt\0All files\0*.*\0");
// ... or ...
strlcpy(szFilters, "Text files\0*.txt\0All files\0*.*\0", 80);
```

The problem is that you're using a function which operates on null-terminated strings but you're giving it a double-null-terminated string. Of course, it will stop copying at the first null terminator, and the result is that `szFilters` is not a valid double-null-terminated string.

Here's another example:

```
sprintf_s(szFilters, 80, "%s\0*.txt\0%s\0*.*\0", "Text files", "All files");
```

Same thing here. Functions from the `sprintf` family take a null-terminated string as the format string. If you "embed" a null character into the format string, the `sprintf` function will treat it as the end of the format string and stop processing.

Here's a more subtle example:

```
CString strFilter;
strFilter.LoadString(g_hinst, IDS_FILE_FILTER);
```

There is no obvious double-null-termination bug here, but there is if you look deeper.

```
BOOL CString::LoadString(UINT nID)
{
  // try fixed buffer first (to avoid wasting space in the heap)
  TCHAR szTemp[256];
  int nCount =  sizeof(szTemp) / sizeof(szTemp[0]);
  int nLen = _LoadString(nID, szTemp, nCount);
  if (nCount - nLen > CHAR_FUDGE)
  {
    *this = szTemp;
    return nLen > 0;
  }



  // try buffer size of 512, then larger size until entire string is retrieved
  int nSize = 256;
  do
  {
    nSize += 256;
    nLen = _LoadString(nID, GetBuffer(nSize - 1), nSize);
  } while (nSize - nLen <= CHAR_FUDGE);
  ReleaseBuffer();

   return nLen > 0;
}
```

Observe that this function loads the string into a temporary buffer, and then if it succeeds, stores the result via the `operator=` operator, which assumes a null-terminated string. If your string resource contains embedded nulls, the `operator=` operator will stop at the first null.

The mistake here was taking a class designed for null-terminated strings and using it for something that isn't a null-terminated string. After all, it's called a `CString` and not a `CDoubleNullTerminatedString`.

Raymond Chen

Follow