The evolution of the text size limits related to the standard static control

devblogs.microsoft.com/oldnewthing/20161229-00

December 29, 2016

Raymond Chen

Michael Quinlan wondered about the text size limits related to the standard static control.[1]

We start with the resource format, since that was the limiting factor in the original problem. The original 16-bit resource format represented strings as null-terminated sequences of bytes, so in theory they could have been arbitrarily large. However, 16-bit string resources were limited to 255 characters because they used a byte count for string length. My guess is that the resource compiled took this as a cue that nobody would need strings longer than 255 characters, so it avoided the complexity of having to deal with a dynamic string buffer, and when it needed to parse a string in the resource file, it did so into a fixed-size 256-byte buffer.

I happen to still have a copy of the original 16-bit resource compiler, so I can actually verify my theory. Here's what I found:

There was a "read next token" function that placed the result into a global variable. Parsing was done by asking to read the next token (making it the current token), and then and then studying the current token. If the token was a string, the characters of the string went into a buffer of size `MAXSTR + 1`. And since string resources have a maximum length of 255, `MAXSTR` was 255.

Although the limit of 255 characters did not apply to dialog controls, the common string parser stopped at 255 characters. In theory, the common string parser could have used dynamic memory allocation to accommodate the actual string length, but remember that we're 16-bit code here. Machines had 256KB of memory, and no memory block could exceed 64KB. Code in this era did relatively little dynamic memory allocation; static memory allocation was the norm. It's like everybody was working on an embedded system.

Anyway, that's where the 255-character limit for strings in resource files comes from. But that's not a limit of the resource file format or of static text controls. It's just a limit of the resource compiler. You can write your own resource compiler that generates long strings if you like.

Okay, so what about the static text control? The original 16-bit static text control had a text size limit of 64KB because 16-bit. This limit carried forward to Windows 95 because the static text control in Windows 95 was basically a 16-bit control with some 32-bit smarts.

On the other hand, Windows NT's standard controls were 32-bit all the way through (and also Unicode). The limits consequently went up from 64KB to 4GB. Some messages needed to be revised in order to be able to express strings longer than 64KB, For example, the old `EM_GETSEL` message returned the start and end positions of the selection as two 16-bit integers packed into a single 32-bit value. This wouldn't work for strings longer than 65535 characters, so the message was redesigned so that the `wParam` and `lParam` are pointers to 32-bit integers that receive the start and end of the selection.

Anyway, now that the 16-bit world is far behind us, we don't need to worry about the 64KB limit for static and edit controls. The controls can now take all the text you give it.[2]

[1] And then for some reason Erkin Alp Güney said that I'm "employed as a PR guy." I found this statement very strange, because not only am I not employed as a PR guy, I have basically no contact with PR at all. The only contact I do have is that occasionally they will send me a message saying that they are upset at something I wrote. I remember that they were very upset about my story that shared some trivia about the //build 2011 conference because it (horrors) talked about some things that went wrong. (And as Tom West noted, it wouldn't seem to be a good idea for PR to employ someone with the social skills of a thermonuclear device.)

[2] Well, technically no. If you give a string longer than 4GB, then it won't be able to handle that. So more accurately, it can handle all the text you would give it, *provided you're not doing something ridiculous*. I mean, you really wouldn't want to manipulate 4GB of data in the form of one giant string. And no human being would be able to read it all anyway.

Raymond Chen

Follow