How a bullet turns into a beep

devblogs.microsoft.com/oldnewthing/20070104-00

January 4, 2007

Raymond Chen

Here's a minor mystery:

```
echo •
```

That last character is U+2022. Select that line with the mouse, right-click, and select Copy to copy it to the clipboard. Now go to a command prompt and paste it and hit Enter.

You'd expect a • to be printed, but instead you get a beep. What happened?

Here's another clue. Run this program.

```
class Mystery {
 public static void Main() {
  System.Console.WriteLine("\x2022");
 }
}
```

Hm, there's that beep again. How about this program:

```
#include <stdio.h>
#include <windows.h>
int __cdecl main(int argc, char **argv)
{
 char ch;
 if (WideCharToMultiByte(CP_OEMCP, 0, L"\x2022", 1,
                         &ch,  1, NULL, NULL) == 1) {
  printf("%d\n", ch);
 }
 return 0;
}
```

Run this program and it prints "7".

By now you should have figured out what's going on. In the OEM code page, the bullet character is being converted to a beep. But why is that?

What you're seeing is `MB_USEGLYPHCHARS` in reverse. Michael Kaplan discussed `MB_USEGLYPHCHARS` a while ago. It determines whether certain characters should be treated as control characters or as printable characters when converting to Unicode. For example, it controls whether the ASCII bell character 0x07 should be converted to the Unicode bell character U+0007 or to the Unicode bullet U+2022. You need the `MB_USEGLYPHCHARS` flag to decide which way to go when converting **to** Unicode, but there is no corresponding ambiguity when converting **from** Unicode. When converting **from** Unicode, both U+0007 and U+2022 map to the ASCII bell character.

"But converting a bullet to 0x07 is clearly wrong. I mean, who expects a printable character to turn into a control character?"

Well, you're assuming that the code who does the conversion is going to interpret it as a control character. The code might treat it as a glyph character, like this:

```
// starting with the scratch program
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{
 HFONT hfPrev = SelectFont(pps->hdc, GetStockFont(OEM_FIXED_FONT));
 TextOut(pps->hdc, 0, 0, "\x07", 1);
 SelectFont(pps->hdc, hfPrev);
}
```

Run this program and you get a happy bullet in the corner of the window. The `TextOut` function does not interpret control characters as control characters; it interprets them as glyphs.

The `WideCharToMultiByte` function doesn't know what you're going to do with the string it produces. It converts the character and leaves you to decide what to do next. There doesn't appear to be a `WC_DONTUSEGLYPHCHARS` flag, so you're going to get glyph characters whether you like it or not.

(Postscript: You can see this happening in reverse from the command prompt. Then again, since this problem is itself a reversal, I guess you could say the behavior is happening in the forward direction now... Type `echo ^A` where you actually type Ctrl+A where I wrote `^A`. The result: A smiling face, U+263A.)

Raymond Chen

Follow