Why am I getting mojibake when I try to create a window?

devblogs.microsoft.com/oldnewthing/20190110-00

January 10, 2019

Raymond Chen

A customer was compiling their program as Unicode, but since their data was almost all in ASCII, they were using the ANSI versions of the APIs. They registered their class with the `RegisterClassA` function and created it with the `CreateWindowExA` function. But the window title came out as Chinese mojibake.

But not all strings were coming out corrupted. Strings they passed to `MessageBoxA`, for example, were displayed correctly.

The customer shared their code that registered the window class and created the window.

```
WNDCLASSA wc = { };
wc.style = 0;
wc.lpfnWndProc = AwesomeWndProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = MyAwesomeInstance;
wc.hIcon = MyAwesomeIcon;
wc.hCursor = LoadCursor(nullptr, IDC_ARROW);
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
wc.lpszClassName = "MyAwesomeClass";
if (!RegisterClassA(&wc)) return FALSE;

hwnd = CreateWindowExA(
    0, "MyAwesomeClass", "My awesome title",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFALT, CW_USEDEFALT,
    CW_USEDEFALT, CW_USEDEFALT,
    nullptr, nullptr, MyAwesomeInstance, 0);
```

Chinese mojibake usually means that somebody took an ANSI string and misinterpreted it as a Unicode string. The puzzle is to figure out where it happened.

My psychic powers told me that the answer was in code they didn't provide:

> Can you check that your window procedure is calling the correct `DefWindowProc`?

Indeed, that was the source of the problem. Their window procedure was registered with `RegisterClassA`, which means that it is an ANSI window procedure and will be given ANSI window messages. Their window procedure was finishing with a call to `DefWindow-Proc`, which due to the project's configuration as Unicode, meant that they were calling `DefWindowProcW`. They were passing ANSI window messages to a Unicode function, and that was the source of the mojibake.

The fix for this specific problem was to finish with a call to the `DefWindowProcA` function.

But really, the real problem is that they compiled a program as Unicode, while carefully avoiding every Unicode feature. If you miss a spot and accidentally do a Unicode thing, you might get lucky and trigger a compiler warning. Or you might be unlucky, and everything will compile, and something will subtly go wrong at runtime. And chasing down all those subtle errors will be time-consumingl.

I don't know why they set it to Unicode and then try to avoid all the Unicode stuff. If they dislike Unicode so much, they may as well be clear about it: Set the project to ANSI.

Raymond Chen

Follow