The importance of the FORMAT_MESSAGE_IGNORE_INSERTS flag

devblogs.microsoft.com/oldnewthing/20071128-00

November 28, 2007

Raymond Chen

You can use the `FormatMessage` message with the `FORMAT_MESSAGE_FROM_SYSTEM` flag to indicate that the message number you passed is an error code and that the message should be looked up in the system message table. This is a specific case of the more general case where you are not in control of the message, and when you are not in control of the message, you had better pass the `FORMAT_MESSAGE_IGNORE_INSERTS` flag.

Let's look at what happens when you don't.

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
int __cdecl main(int argc, char **argv)
{
 TCHAR buffer[1024];
 DWORD dwError = ERROR_BAD_EXE_FORMAT;
 DWORD dwFlags = FORMAT_MESSAGE_FROM_SYSTEM;
 DWORD dwResult = FormatMessage(dwFlags, NULL, dwError,
                                0, buffer, 1024, NULL);
 if (dwResult) {
  _tprintf(_T("Message is \"%s\"\n"), buffer);
 } else {
  _tprintf(_T("Failed! Error code %d\n"), GetLastError());
 }
 return 0;
}
```

If you run this program, you'll get

```
Failed! Error code 87
```

Error 87 is `ERROR_INVALID_PARAMETER`. What went wrong? Let's pass the `FORMAT_MESSAGE_IGNORE_INSERTS` flag to see what the message was. Change the value of `dwFlags` to

```
DWORD dwFlags = FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS;
```

and run the program again. This time you get

```
Message is "%1 is not a valid Win32 application.
"
```

Aha, now we see the problem. The message corresponding to `ERROR_BAD_EXE_FORMAT` contains an insertion `%1`. If you don't pass the `FORMAT_MESSAGE_IGNORE_INSERTS` flag, the `FormatMessage` function will insert the first parameter in the argument list (or argument array). But we didn't pass an argument list, so the function fails.

Actually, we got lucky. If we had passed an argument list or argument array, the function would have inserted the corresponding string, even if the argument list we passed didn't have a string in the first position.

If you are not in control of the format string, then you must pass `FORMAT_MESSAGE_IGNORE_INSERTS` to prevent the `%1` from causing trouble. If somebody was being particularly evil, they might decide to give you a format string that contains a `%9`, which is almost certainly more insertions than you provided. The result is a buffer overflow and probably a crash.

This may have been obvious to some people, in the same way that you shouldn't pass a string outside your control as the format string to the `printf` function, but I felt it worth mentioning.

Raymond Chen

Follow