Why does the BackupWrite function take a pointer to a modifiable buffer when it shouldn't be modifying the buffer?

devblogs.microsoft.com/oldnewthing/20150703-00

July 3, 2015

Raymond Chen

The `BackupWrite` function takes a non-const pointer to the buffer to be written to the file being restored. Will it actually modify the buffer? Assuming it doesn't, why wasn't it declared const? It would be much more convenient if it took a const pointer to the buffer, so that people with const buffers didn't have to `const_cast` every time they called the function. Would changing the parameter from non-const to const create any compatibility problems?

Okay, let's take the questions in order.

Will it actually modify the buffer? No.

Why wasn't it declared const? My colleague Aaron Margosis explained that the function dates back to Windows NT 3.1, when const-correctness was rarely considered. A lot of functions from that area (particularly in the kernel) suffer from the same problem. For example, the computer name passed to the `RegConnectRegistry` function is a non-const pointer even though the function never writes to it.

Last question: Can the parameter be changed from non-const to const without breaking compatibility?

It would not cause problems from a binary compatibility standpoint, because a const pointer and a non-const pointer take the same physical form in Win32. However, it breaks source code compatiblity. Consider the following code fragment:

```
BOOL WINAPI TestModeBackupWrite(
  HANDLE hFile,
  LPBYTE lpBuffer,
  DWORD nNumberOfBytesToWrite,
  LPDWORD lpNumberOfBytesWritten,
  BOOL bAbort,
  BOOL bProcessSecurity,
  LPVOID *lpContext)
{
 ... simulate a BackupWrite ...
 return TRUE;
}

BOOL (WINAPI *BACKUPWRITEPROC)(HANDLE, LPBYTE, DWORD,
                 LPDWORD, BOOL, BOOL, LPVOID *);
BACKUPWRITEPROC TestableBackupWrite;

void SetTestMode(bool testing)
{
 if (testing) {
  TestableBackupWrite = TestModeBackupWrite;
 } else {
  TestableBackupWrite = BackupWrite;
 }
}
```

The idea here is that the program can be run in test mode, say to do a simulated restore. (You see this sort of thing a lot with DVD-burning software.) The program uses `TestableBackupWrite` whenever it wants to write to a file being restored from backup. In test mode, `TestableBackupWrite` points to the `TestModeBackupWrite` function; in normal mode, it points to the `BackupWrite` function.

If the second parameter were changed from `LPBYTE` to `const BYTE *`, then the above code would hit a compiler error.

Mind you, maybe it's worth breaking some source code in order to get better const-correctness, but for now, the cost/benefit tradeoff biases toward leaving things alone.

Raymond Chen

Follow