How do I convert a synchronous file handle into an asynchronous one?

devblogs.microsoft.com/oldnewthing/20130812-00

August 12, 2013

Raymond Chen

Say you opened a file in synchronous mode, and then you realize that you want to issue asynchronous I/O on it, too. One way to do this is to call `CreateFile` a second time with the `FILE_FLAG_OVERLAPPED`, but this requires you to know the file name, and the file name may not be readily available to the function that wants to do the conversion, or it may not even be valid any longer if the file has been renamed in the meantime.

Enter `ReOpenFile`. This basically lets you do a `CreateFile` based on another handle rather than a file name. It differs from `DuplicateHandle` because it actually goes and opens the file again (as opposed to merely creating another reference to the same file object in the kernel). This means that you have the opportunity to choose new handle attributes, like whether you want the handle to be synchronous or asynchronous.

```
#include <windows.h>
#include <stdio.h>
int __cdecl main(int, char **)
{
 HANDLE h = CreateFile("test", GENERIC_WRITE, FILE_SHARE_READ, NULL,
                       CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
 HANDLE h2 = ReOpenFile(h, GENERIC_READ, FILE_SHARE_READ |
                        FILE_SHARE_WRITE, FILE_FLAG_OVERLAPPED);
 DWORD cbResult;
 WriteFile(h, "!", 1, &cbResult, NULL);
 OVERLAPPED o = { 0 };
 o.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
 char ch = 0;
 BOOL fRc = ReadFile(h2, &ch, 1, &cbResult, &o);
 if (fRc) {
  printf("read completed synchronously\n");
 } else if (GetLastError() == ERROR_IO_PENDING) {
  printf("read proceeding asynchronously\n");
 } else {
  printf("read failed\n");
 }
 GetOverlappedResult(h2, &o, &cbResult, TRUE);
 printf("Result was %c\n", ch);
 CloseHandle(o.hEvent);
 CloseHandle(h2);
 CloseHandle(h);
 return 0;
}
```

The program opens a test file for writing, and then uses the `ReOpenFile` function to open the same file for reading. (Since you are opening the file twice, be careful to choose compatible sharing modes.) We synchronously write an exclamation point to the file via the first handle, and then we asynchronously read it back with the second handle.

It's really not that exciting.

Raymond Chen

Follow