Answer to quick puzzle about security and synchronization

devblogs.microsoft.com/oldnewthing/20050607-34

June 7, 2005

Raymond Chen

As many people quickly figured out, the reason why the the `WaitForSingleObject` returns immediately is that the call is failing. The reason is that the second process opened the handle with `EVENT_MODIFY_STATE` access, which grants permission to call the `SetEvent` function, the `ResetEvent` function, and the fatally flawed `PulseEvent` function, but it doesn't include `SYNCHRONIZE` access, which is necessary if you intend to synchronize on the object (*i.e.*, wait on it).

The fix is for Process B to ask for `SYNCHRONIZE` access instead of `EVENT_MODIFY_STATE`.

The fact that it's happening in a second process is a red herring. You can put this code in the same process and it will fail/succeed in the same way:

```
HANDLE hEventA = CreateEvent(NULL, FALSE, TRUE, TEXT("MyNamedEvent"));
HANDLE hEventB = OpenEvent(EVENT_MODIFY_STATE, FALSE, TEXT("MyNamedEvent"));
WaitForSingleObject(hEventB, INFINITE); // fails
```

Indeed, the fact that the object is named is a red herring. It has nothing to do with named/unnamed objects.

```
HANDLE hEventA = CreateEvent(NULL, FALSE, TRUE, NULL);
HANDLE hEventB;
DuplicateHandle(GetCurrentProcess(), hEventA,
                GetCurrentProcess(), &hEventB,
                EVENT_MODIFY_STATE, FALSE, 0);
WaitForSingleObject(hEventB, INFINITE); // fails
```

In all three cases, the fix is to change `EVENT_MODIFY_STATE` to `SYNCHRONIZE`.

Raymond Chen

Follow