A helper template function to wait for WaitOnAddress in a loop

devblogs.microsoft.com/oldnewthing/20180118-00

January 18, 2018

Raymond Chen

The `WaitOnAddress` function suffers from the problem of spurious wake-ups. This means that most uses of the `WaitOnAddress` function are of the form "while the value is bad, wait for it to change."

There is a subtlety here, because you have to capture the value, then make your decision based on the captured value, and if you decide that you want to wait some more, you need to pass the captured value to `WaitOnAddress`. The extra capturing is necessary to avoid a race condition if you determine that the value is bad, but before you can call `WaitOnAddress`, the value becomes good.

Here's a simple helper function to encapsulate the loop:

```
template<typename T, typename TLambda>
void WaitForValueByAddress(T& value, TLambda&& is_okay)
{
  auto capturedValue = value;
  while (!is_okay(capturedValue)) {
    WaitOnAddress(&value, &capturedValue, sizeof(value), INFINITE);
    capturedValue = value;
  }
}
```

The assumption here is that `T` is a simple value type like `int32_t`. If you pass a funky class, then we're going to be copying it, which is probably a bad idea given that the variable is going to be asynchronously modified (possibly while we are copying it).

The predicate evaluates the value: Return `true` if it acceptable, or return `false` to reject it and wait some more.

Here's a sample usage:

```
int32_t someValue;

void WaitForValueToBecomeZero()
{
  WaitForValueByAddress(someValue, [](auto&& v) { return v == 0; });
}
```

Raymond Chen

Follow