BOOL vs. VARIANT_BOOL vs. BOOLEAN vs. bool

devblogs.microsoft.com/oldnewthing/20041222-00

December 22, 2004

Raymond Chen

Still more ways of saying the same thing. Why so many?

Because each was invented by different people at different times to solve different problems.

`BOOL` is the oldest one. Its definition is simply

```
typedef int BOOL;
```

The C programming language uses "int" as its boolean type, and Windows 1.0 was written back when C was the cool language for systems programming.

Next came `BOOLEAN`.

```
typedef BYTE  BOOLEAN;
```

This type was introduced by the OS/2 NT team when they decided to write a new operating system from scratch. It lingers in Win32 in the places where the original NT design peeks through, like the security subsystem and interacting with drivers.

Off to the side came `VARIANT_BOOL`.

```
typedef short VARIANT_BOOL;
#define VARIANT_TRUE ((VARIANT_BOOL)-1)
#define VARIANT_FALSE ((VARIANT_BOOL)0)
```

This was developed by the Visual Basic folks. Basic uses `-1` to represent "true" and `0` to represent "false", and `VARIANT_BOOL` was designed to preserve this behavior.

Common bug: When manipulating `VARIANT`s of type `VT_BOOL`, and you want to set a boolean value to "true", you must use `VARIANT_TRUE`. Many people mistakenly use `TRUE` or `true`, which are not the same thing as `VARIANT_TRUE`. You can cause problem with scripting languages if you get them confused. (For symmetry, you should also use `VARIANT_FALSE` instead of `FALSE` or `false`. All three have the same numerical value, however. Consequently, a mistake when manipulating "false" values is not fatal.)

Newest on the scene is `bool`, which is a C++ data type that has the value `true` or `false`. You won't see this used much (if at all) in Win32 because Win32 tries to remain C-compatible.

(Note that C-compatible isn't the same as C-friendly. Although you can do COM from C, it isn't fun.)

Raymond Chen

Follow