Why do atoms start at 0xC000?

devblogs.microsoft.com/oldnewthing/20080429-00

April 29, 2008

Raymond Chen

There are two types of atoms, so-called *integer atoms*, which are just small integers, and, um, atoms that don't have a special adjective; these plain vanilla atoms come from functions like `AddAtom`. (For the purpose of this discussion, I'll call them *string atoms*.) The atom zero is invalid (`INVALID_ATOM`); atoms 1 through `MAXINTATOM-1` † are integer atoms, and atoms from `MAXINTATOM` through `0xFFFF` are string atoms. Why is the value of `MAXINTATOM` `0xC000` ? The reason has its roots in 16-bit Windows. Atoms are kept in a, well, atom table. The details of the atom table aren't important aside from the fact that the nodes in the atom table are allocated from the heap, and each node corresponds to one string in that atom table. Since we're working in 16-bit Windows, the pointers in the atom table are 16-bit pointers, and all memory blocks in the 16-bit heap are 4-byte aligned. Alignment on 4-byte boundaries means that the bottom two bits of the address are always zero. Therefore, there are only 14 bits of value in the node pointer. Take that pointer, shift it right two places, and set the top two bits, and there you have your atom. Conversely, to convert an atom back to a node, strip off the top two bits and shift the result left two places. Why encode the pointer this way? Well, you have 14 bits of information and you want to return a 16-bit value. You have two bits to play with, so your decisions are where to put those bits and what values they should have. It'd be convenient if all the integer atoms were contiguous and all the string were contiguous, to make range checking easier. Now you're down to two options. You have a 49152-value range of integer atoms and a 16384-value range of string atoms. Either you put the integer atoms at the low end (`0x0000-0xBFFF`) and the string atoms at the high end (`0xC000-0xFFFF`), or you put the string atoms at the low end (`0x0000-0x3FFF`) and the integer atoms at the high end (`0x4000-0xFFFF`). You probably don't want zero to be a valid string atom, since that's the most likely value for an uninitialized atom variable, so putting the string atoms at the top of the range wins out. Now, with the conversion to Win32, the old implementation of atoms was thrown out. Atoms are no longer encoded pointers, but the new implementation still had to adhere to the breakdown of the 16-bit atom space into integer atoms and string atoms.

Over the next few entries, we'll take a look at other consequences of the way string atoms are assigned and surprising things you can do with atoms. (Not necessarily good things, but still surprising.)

Footnotes

†The `MAXINTATOM` symbol adheres to the classic Hungarian convention that “max” is one more than the highest legal value.

Raymond Chen

Follow