On the ambiguity of uniqueness

devblogs.microsoft.com/oldnewthing/20051214-15

December 14, 2005

Raymond Chen

The MSDN documentation for `System.Object.GetHashCode` says

> [T]he implementation of GetHashCode provided by the String class returns unique hash codes for unique string values.

This is another case of ambiguous use of the word “unique”. The intended meaning is “for each string value, the same hash code is returned”.

Even though “unique” means “one and only one”, the domain in which the term applies is often left unsaid, as here, where the domain of comparison is “all the hash codes returned for a specific string value”. If you instead misinterpreted the domain as “all the hash codes returned for all string values”, then you end up erroneously concluding that no two strings hash to the same value.

Another conflicting sense of “unique” is “you get the same one each time” as opposed to “you get a different one each time”.

- `GetCurrentProcessId` returns a unique value that identifies the process. You get the same one each time.
- `CoCreateGuid` returns a unique GUID. You get a different one each time.

In the original C standard, `malloc(0)` is permitted to return `NULL` or “a unique pointer”. What does “unique” mean here? Does it mean that the non- `NULL` return value is always the same? Can I `assert(malloc(0) == malloc(0))` ? Or does it mean that the non- `NULL` return value is a value distinct from any other return value from `malloc()` ?

In Technical Corrigendum 1, this ambiguity was resolved by removing the word “unique”. Instead, the specification says “as if the size were some nonzero value” which makes it clear that it is the second interpretation that is intended.

My suggestion: Don’t use the word “unique”. It’s too ambiguous.

Raymond Chen

Follow