Writing a sort comparison function, redux

devblogs.microsoft.com/oldnewthing/20090508-00

May 8, 2009

Raymond Chen

Prerequisites: Experience with sort functions such as `qsort`.

Overqualifications: Familiarity with sort algorithms.

If you break the rules for sort comparison functions, then don't expect the result of a sort call to be meaningful. In fact, if you *really* mess it up, you can corrupt memory or go into an infinite loop.

> My program hangs when I used this sort comparison function (pseudocode):
>
> ```
> int compare(x, y)
> {
>  return x >y ? +1 : -1;
> }
> ```
>
> What's even more strange is that sometimes instead of hanging, the program crashes. It all works correctly if I add one line:
>
> ```
> int compare2(x, y)
> {
>  if (x == y) return 0; // added
>  return x >y ? +1 : -1;
> }
> ```
>
> What's going on? The array I am sorting contains no duplicate elements, so the two items `x` and `y` should never be equal.

First of all, your first comparison function clearly violates the requirements for being a comparison function: It must return zero if the two items compare equal. As a result, the behavior is undefined, and hanging, crashing, or returning an incorrect result are all legitimate behavior.

But let's dig a little deeper to see why hanging and crashing are plausible responses to an invalid sort comparison function.

A common sorting algorithm is quicksort. I'll leave you to go off and learn on your own how the quicksort algorithm works. Come back when you're done.

Okay, welcome back. The central step in quicksort is the partition, and some variants of the quicksort algorithm rely on the partition element comparing equal to itself in order to remove a comparison from the inner loop. For example, an index may walk through the array looking for an element greater than or equal to the partition, fully expecting at least one such element to be found because in the worst case, it will find the partition itself. But if your comparison function fails to report the partition as equal to itself, this search may run off the end of the array and eventually crash with an access violation when it reaches invalid memory.

That explains the crash, but what about the hang? Well, notice that the comparison function is also inconsistent. In particular, the anti-symmetry rule is violated: `compare(x, y)` and `compare(y, x)` return the same value (as opposed to the opposite value) if `x==y`. The function returns `-1` both times, saying "`x` is less than y" and "`y` is less than x" simultaneously. This inconsistency can easily send a sort algorithm running back and forth trying to find the right place for the partition.

The moral of the story is the same: Your comparison function must meet the criteria for a proper comparison function. If you fail to do this, then the results you get will be very strange indeed.

Raymond Chen

Follow