Subject: Re: An idea for heap-checking stuffs
Posted by mirek on Sun, 04 May 2008 16:23:56 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 04 May 2008 06:06After last bug (still unresolved) of memory corruption on upp compiled under ubuntu hardy, I was thinking about some heap checking stuffs that IMO could be incorporated in upp.
So, the idea (I don't know if somebody already did it!) :

1) Set up a new building flag, for example HEAPCHECK.

Good idea to actviate heap checks in release mode.

Quote:
3) On each dynamic allocation, reserve some more bytes, some before and some after returned pointer, and fill with known data.
For example, if I need 10 bytes, I could reserve 20, like this :

DDDDDAAAAAAAAADDDDD
    |
returned pointer here

DDDDD represent the 'spare' allocated bytes, filled with known values.

This is already done (in debug mode).

Quote:
4) Keep a linked list not only for freed data but also for allocated data. I know that this can slow down much the code, but.... it's just when needed for debugging.

Yep, this is also done BTW, I am using these links as sentinels. And it is also used to check leaks.

Quote:
5) create 2 functions, FreeCheck() and UsedCheck() that scans the free and used allocated space and checks for values on DDDDD fileds.

MemoryCheck.

Quote:
6) allow the ability to switch on/off the heap checking on each allocation/free of memory. That one would slow down much the code, but would also allow to find corruptions just a little after they happens.

Each allocation has serial number. You can set breakpoint to this serial number to catch the very allocation that gets corrupted later.

Quote:
Up to here, not much work in upp code, IMHO.
The best would also be to add

Indeed. Only MEMORYCHECK in release.

Quote:
7) Add container's methods entry/exit pointer checking. So, for each method called for a container, check container's pointers on method entry and exit. That one would catch 99% of pointer misusage as soon as it happens. Of course, that last one would mean to add a lot of (conditional) code to upp core.

That is a good idea. A lot of code involved thought ("each method"

Mirek