

Re: when can realloc fail?

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-04/msg00162.html

- *From:* Nelu <spamahead@xxxxxxxxxx>
 - *Date:* Sun, 01 Apr 2007 20:47:05 -0400
-

banansol@xxxxxxxxxxxxx wrote:

On Apr 1, 9:06 pm, Eric Sosman <esos...@xxxxxxxxxxxxxxxxxxxxxxxx> wrote:

banan...@xxxxxxxxxxxxx wrote:

<snip>

A call to realloc() will return NULL on error and the original memory is left untouched, both when requesting a larger or a smaller size than the original, right? But a call to realloc() with size set to zero is equivalent to free(), which returns void. Does that mean that a call to realloc() can fail when shrinking memory except when shrinking it to zero in which case it will always succeed?

<snip>

But C99 has no such text, and makes no special case for size zero. All we're told is that realloc(...,0) either fails or it returns a pointer to an object of size zero. Nothing I can find in C99 forbids realloc(...,0) to fail, so presumably it can.

Thank you for your reply!
But I wonder, if realloc(...,0) fails in C99, how can I know that? How can I know if a returned NULL means that realloc() failed, or if it is the pointer to the new memory, empty, which is NULL. (That is what I got on my compiler anyway)

Re: when can realloc fail?

In C99 it only returns NULL when it fails. If it returns non-null and the size is 0 then it will return an address to a block of memory of zero size. The address *will not be NULL* and you will have to free it using free().

—

Ioan – Ciprian Tandau
tandau_at_freeshell_dot_org (hope it's not too late)
(... and that it still works...)

.