

Wrong vs. correct memory allocation

I have received lots of E-mails complaining something like:

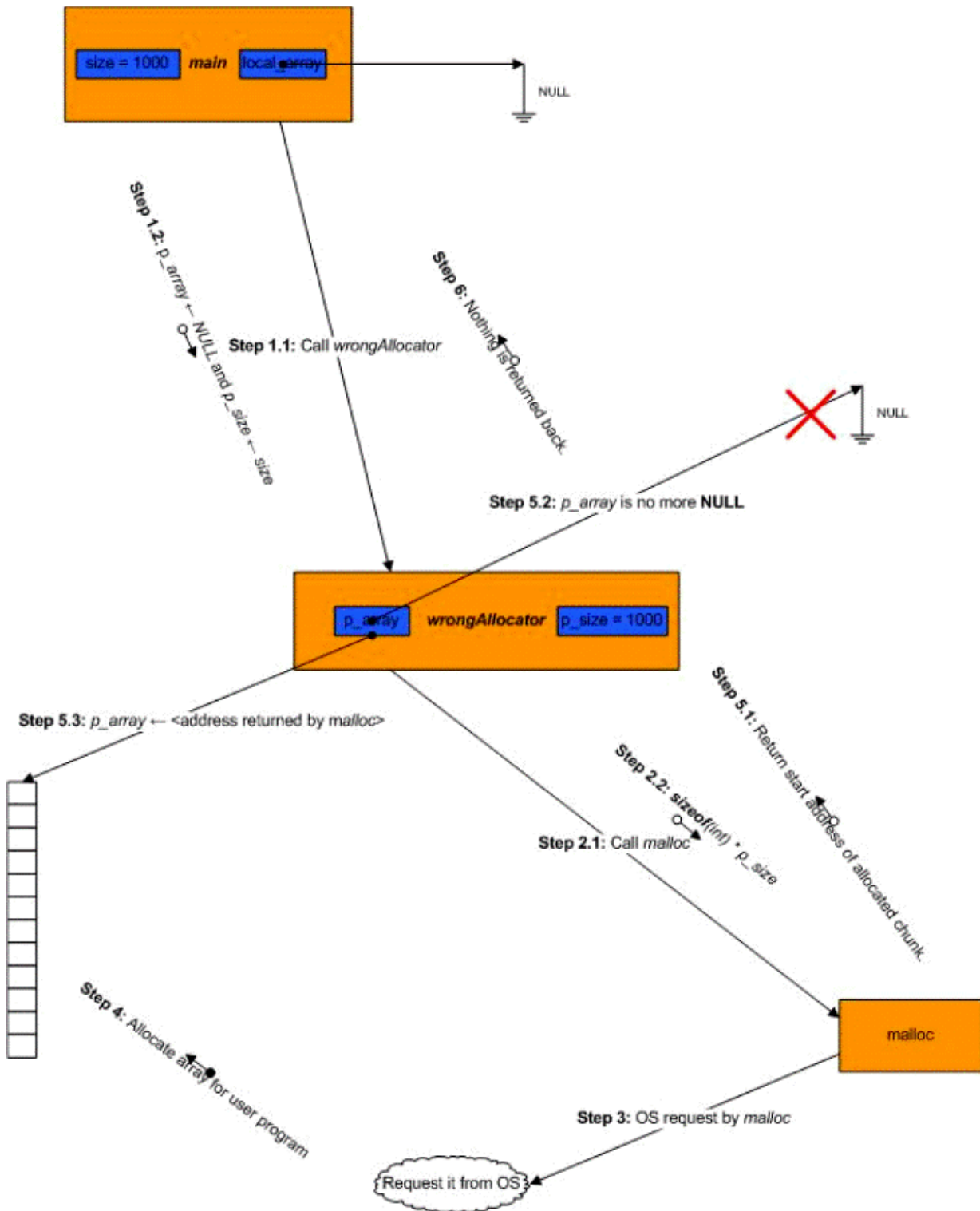
"I'm sure that I've successfully allocated some memory in some function but C suddenly frees this memory chunk after I returned to the caller function."

First of all, it is not possible for a programming language to behave non-deterministically. Moreover, while programming with C, you are the one doing the whole work. So, you are the one managing whole memory allocation/utilization as far as heap memory is concerned (*malloc/calloc/realloc/free* deals with heap).

The problem you are suffering from is totally based on misunderstanding of *call-by-value* and *call-by-reference* issues. I will trace over an example and try to explain what actually happens.

Wrong memory allocation

Initially **local_array** pointer is set to *NULL* and we send it to **wrongAllocation** function as formal parameter value (**p_array**). Within the function a *malloc* invocation is performed and we assume that *malloc* successfully allocates the required amount of memory from heap memory. The starting address of this chunk is set to **p_array** parameter and then function terminates. Now tell me the value of **local_array** pointer. *NULL!!!*. Let me illustrate it:

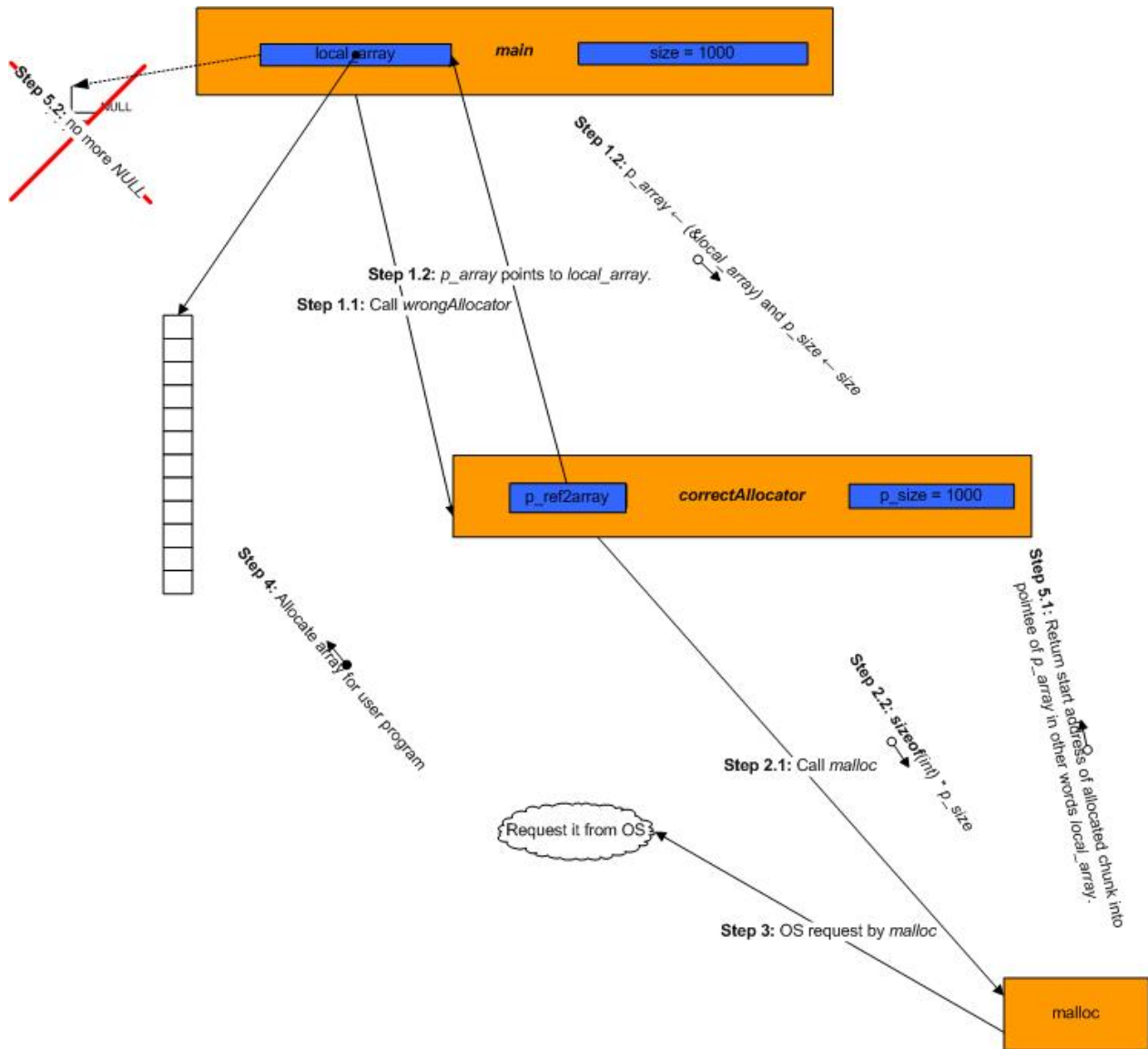


This is because we send `local_array` parameter to the function using *call-by-value* parameter passing method. This means no change with in the `wrongAllocation` function is reflected to the caller function. As a result, we lost the pointer pointing to the starting location (address) of allocated chunk (this means memory leak. In other words,

you will be never accessing this location any more). Moreover, remember that **local_array** parameter is still *NULL*. So anyone assumes memory allocation is succeeded will face with run-time error.

Correct memory allocation

Let's try the correct way of doing this:



This will work. Do you see why? That's because we send array pointer (**local_array**) via *call-by-reference* parameter passing method.

Conclusion

In C programming you need to answer those questions about dynamic memory allocation:

- Have I allocated memory appropriately? (**malloc** or so does not return *NULL*.)
- Are starting addresses of memory chunks allocated in some functions accessible by their caller? (Just like it is not in the first example. It is not possible by **main** function to access memory allocated in **wrongAllocator**).
- Have I freed memory chunks I have previously allocated? (You need to prevent your codes from memory leakage.)