

Experimental API

The experimental API is subject to change or removal without regard for backward compatibility.

The `allocm()`, `reallocm()`, `sallocm()`, and `dallocm()` functions all have a *flags* argument that can be used to specify options. The functions only check the options that are contextually relevant. Use bitwise or (`|`) operations to specify one or more of the following:

`ALLOCM_LG_ALIGN(la)`

Align the memory allocation to start at an address that is a multiple of (`1 << la`). This macro does not validate that *la* is within the valid range.

`ALLOCM_ALIGN(a)`

Align the memory allocation to start at an address that is a multiple of *a*, where *a* is a power of two. This macro does not validate that *a* is a power of 2.

`ALLOCM_ZERO`

Initialize newly allocated memory to contain zero bytes. In the growing reallocation case, the real size prior to reallocation defines the boundary between untouched bytes and those that are initialized to contain zero bytes. If this option is absent, newly allocated memory is uninitialized.

`ALLOCM_NO_MOVE`

For reallocation, fail rather than moving the object. This constraint can apply to both growth and shrinkage.

The `allocm()` function allocates at least *size* bytes of memory, sets **ptr* to the base address of the allocation, and sets **rsize* to the real size of the allocation if *rsize* is not `NULL`.

The `reallocm()` function resizes the allocation at **ptr* to be at least *size* bytes, sets **ptr* to the base address of the allocation if it moved, and sets **rsize* to the real size of the allocation if *rsize* is not `NULL`. If *extra* is non-zero, an attempt is made to resize the allocation to be at least *size + extra* bytes, though inability to allocate the extra byte(s) will not by itself result in failure. Behavior is undefined if *(size + extra > SIZE_T_MAX)*.

The `sallocm()` function sets **rsize* to the real size of the allocation.

The `dallocm()` function causes the memory referenced by *ptr* to be made available

The `dallocx()` function causes the memory referenced by *ptr* to be made available for future allocations.