

realloc

realloc

Suppose a program initially asks for n bytes on the heap, but later discovers that it needs more than n bytes. In this case, it is not acceptable to simply reference past the size of the initial **malloc** request. For example, if an array has 4 elements, which are indexed via **array[0]** to **array[3]**, it would *not* be acceptable to simply write into **array[4]** just because the program's space requirements for the array have changed. In locker-speak, this is the equivalent to realizing you need a 6th locker and taking locker 42, even though it was not given to you to use. (Locker 42 might be in use by someone else, or it might be given to another person at a later time.) The proper way to respond to this increased space needs is to use **realloc**.

```
void * realloc(void *ptr, size_t size);
```

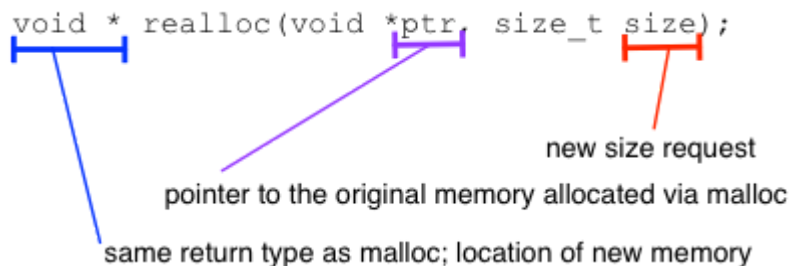


Diagram illustrating the signature of the **realloc** function:

- void ***: same return type as malloc; location of new memory
- *ptr**: pointer to the original memory allocated via malloc
- size_t size**: new size request

realloc effectively resizes a malloced region of memory. Its signature is shown in the figure above. The arguments to **realloc** are the pointer to the region in memory that you wish to resize and the new size you wish the region in memory to have. If successful, **realloc** will return a pointer to the new, larger location in the heap that is now at your disposal. If no area in the heap of the requested size is available, **realloc** returns **NULL**.

Keep in mind that the new location in memory does not need to be anywhere near the original location in the heap. In terms of our locker analogy, if you return to the window and say "I have the lockers starting at locker 37, but I need 6 lockers now," the worker may not be able to give you locker 42. Instead the worker may respond "Okay, your new starting locker is 12." Conveniently, the worker will move the contents of lockers 37–41 into lockers 12–16 for you.

Beyond **malloc**, **free**, and **realloc**, there are a few more standard functions available for memory allocation, such as **calloc** (which zeroes out the region in memory for you—by contrast, **malloc** does nothing to initialize the memory). The manpages for all of these functions are, as always, great reference.



Completed

