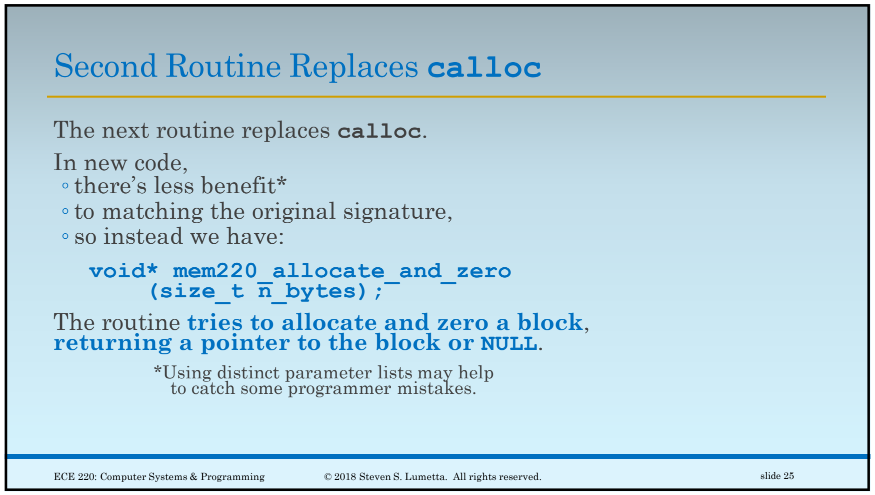


Second Routine Replaces **calloc**

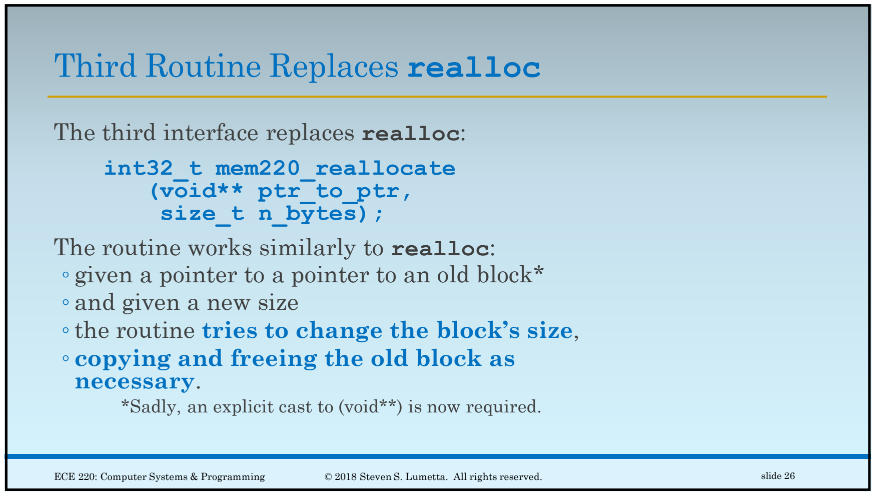
The next routine replaces `calloc`.

In new code,
- there's less benefit*
- to matching the original signature,
- so instead we have:

```
void* mem220_allocate_and_zero
        (size_t n_bytes);
```

The routine **tries to allocate and zero a block**, **returning a pointer to the block or `NULL`**.

*Using distinct parameter lists may help to catch some programmer mistakes.



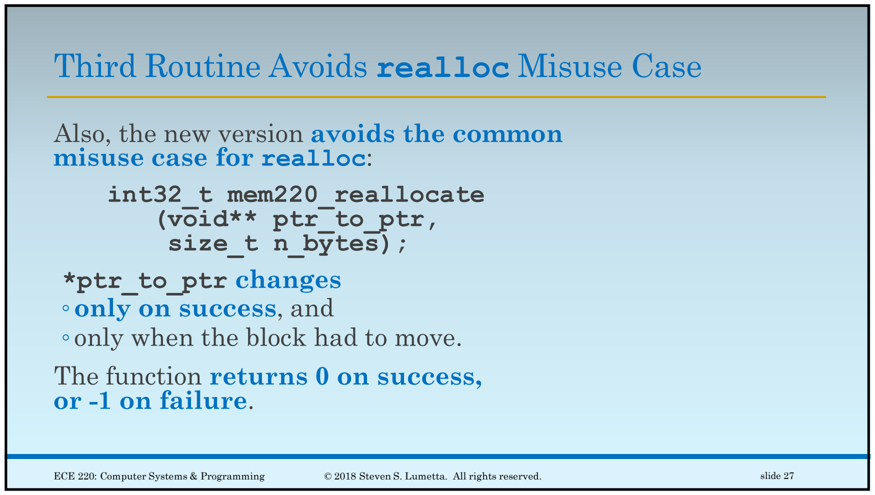
Third Routine Replaces **realloc**

The third interface replaces `realloc`:

```
int32_t mem220_reallocate
      (void** ptr_to_ptr,
       size_t n_bytes);
```

The routine works similarly to `realloc`:
- given a pointer to a pointer to an old block*
- and given a new size
- the routine **tries to change the block's size**,
- **copying and freeing the old block as necessary**.

*Sadly, an explicit cast to (void**) is now required.



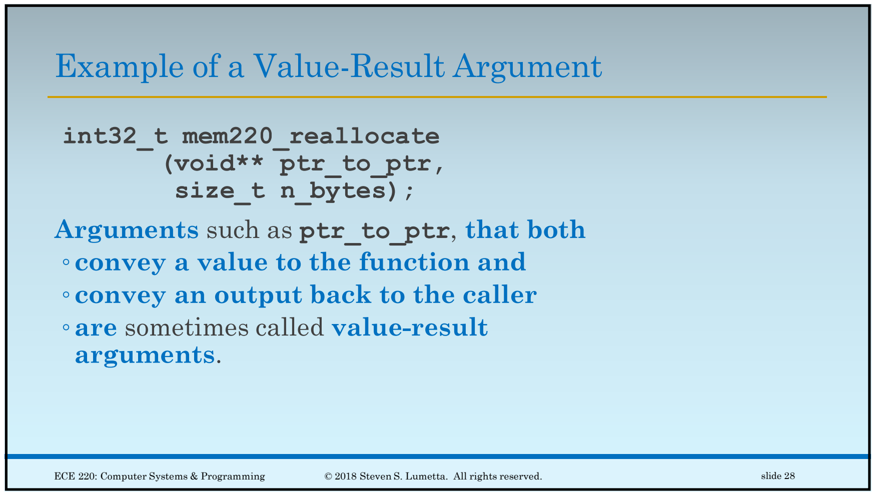
Third Routine Avoids **realloc** Misuse Case

Also, the new version **avoids the common misuse case for `realloc`**:

```
int32_t mem220_reallocate
      (void** ptr_to_ptr,
       size_t n_bytes);
```

`*ptr_to_ptr` **changes**
- **only on success**, and
- only when the block had to move.

The function **returns 0 on success, or -1 on failure**.



Example of a Value-Result Argument

```
int32_t mem220_reallocate
        (void** ptr_to_ptr,
         size_t n_bytes);
```

Arguments such as `ptr_to_ptr`, **that both**
- **convey a value to the function and**
- **convey an output back to the caller**
- **are** sometimes called **value-result arguments**.