Welcome CS170

Ryan Wenger

ryanwenger@ucsb.edu

Office Hours: TBD (see Canvas announcement)

Vote for OH times

Quick Aside: How to do well in this class

- Go
- To
- Lecture
- !!!
- That’s all!
- 9:30am is brutal but I promise it’s worth it
  - No matter how much you already know, you will learn *something*
  - Rich is incredibly receptive to questions during lectures (and on ~~Piazza~~Canvas)—ask him things if you need a way to stay awake!
    - Piazza is switching to a subscription model this year, so we’re using Canvas discussions instead

Review – what is malloc()?

- Poor man's version of C++'s `operator new`
  - Allocates "dynamic memory" from "The Heap"
- Every malloc() call **must** be paired with exactly 1 free() call—otherwise we get a memory leak
- Hopefully, none of this is new information

A closer look: “The Heap”

- Is it more than just the magical region of memory that malloc() and free() interface with?
  - Not really, actually (for our purposes).
- Like all other memory, the heap is just many 1’s and 0’s—a bunch of logically contiguous bytes; in some sense, it’s a (very large) array of `unsigned char`
  - Modern systems are naturally more complicated than this, but we are not trying to implement anything as complex as libc malloc()
- In Lab0, this “heap” is just a regular global variable: `unsigned char MyHeap[MAX_MALLOC_SIZE];`
  - Ironically, the buffer used by MyMalloc as its heap is not, in fact, stored on *The Heap*; it’s stored in your program’s .data or .bss segment

How does a memory allocator work?

- How does malloc() know what memory it's allowed to return?
  - Likewise, how does free() know the size of the memory chunk being free'd?
- Typical malloc() implementations use *records* placed at the start of every "chunk" of the heap to track size and used/free status
  - Whether allocated or not, these records *are* present in the bytes preceding a buffer returned by malloc()
  - Try malloc'ing some memory on a *Nix system then write zeros to buf[-1], buf[-2], etc, and watch your program implode
- Crucially, this means that, in order to service any allocation request, the allocator (malloc) requires a chunk of memory at least the size of the request **plus the size of a record**.

Misc. tips

- Rich's writeup at https://sites.cs.ucsb.edu/~rich/class/cs170/labs/lab1.malloc/index.html goes into more detail and includes helpful diagrams, so reference that.
- Prepare to get lost in a rabbit hole if you try to reference "real" malloc implementations—they are interesting, but *way* more complicated.
- Test your code
- Test your code more
- Test your tests too

Vote for OH times