Writing Malloc In C0

Luke Erbsen

1 Introduction

C0 is a memory safe subset of C. It is used as a teaching tool for Carnegie Mellon's Principals of Imperative Programming course. Memory safety, however, takes all of the fun out of programming. If your mission critical code can't fail because of a use after free error what is even the point of coding?

Currently vibecoders and their AI hallucinating slop have lead to many programming errors. We need to go back to the good days of memory errors. Anyone can vibecode AI slop, it takes a real programmer to crash valgrind.

2 Limitations and workarounds

Since C0 is a small subset if C it does not have all of the quirks of C. Unfortunately this makes implementing malloc quite annoying.

2.1 Globals

C0 doesn't support global variables. This is quite annoying. It prevents both global constants and having global pointers. As a workaround global constants can be made by having a function that returns a constant value. Global pointer must be a pointer that the client of this library passes around every time they call any of the functions.

2.2 Typecasting

C0 doesn't support any type casting. There are no void* pointers, no arbitrary changing the signed-ness of ints (there are only unsigned ints). In a traditional malloc implementation you treat the heap like an array of bytes, and you individually manipulate blocks on a byte by byte basis. Because you can't arbitrarily cast between pointer types whatever type you treat your heap as, is the only type that can be allocated in the heap. Integers in C0 are 32 bits, the largest type in C0. Because of this the heap is an array of integers, to allow the most flexiable storage option.

2.3 Very Few Types

Unlike c's extensive types (both signed and unsigned) C0 only support signed types. Additionally, the largest non-pointer type is 32 bits, unlike c which supports 64 bit types. In a traditional malloc implementation you treat integers like pointers to form an implicit linked list. Because of the lack of typecasting and no 64 bit integer types, there is no linked list. This means everytime the user wants to allocate a new block of memory, the entire list must be searched through.

2.4 No sbrk

C0 does not allow you to directly interface with the heap. This means that the initial size of the heap is constant. There is also no realloc feature built into C0. This means to expand the size of the heap, a second block of memory, not attached to the first, would need to be created and all data copied over. Thus the heap would function like an unbounded array. However, this library is not designed to be useful, so the programmer really should be more careful about the inital amount of memory they choose to allocate.

2.5 Address Of

In c you can use the & character to get the address of data. This is helpful for returning a pointer to the start of malloced memory. C0 does not have this feature. As a workaround data is stored separately that contains the address of the pointer relative to the start of the heap.

3 Fun Behavior that is Supported

Nothing is byte aligned. You can technically get the size of a block that is allocated, however, that would make this library slightly useful, so it is not provided.