

A black t-shirt with a crew neck and short sleeves, centered against a white background. The text "free the mallocs" is printed in a white, monospaced, typewriter-style font across the chest.

free the mallocs

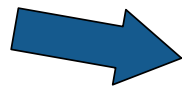
TODO

- Assignment 2 assigned due October 7

Overview for today

- Dynamic memory allocation
- Debugging

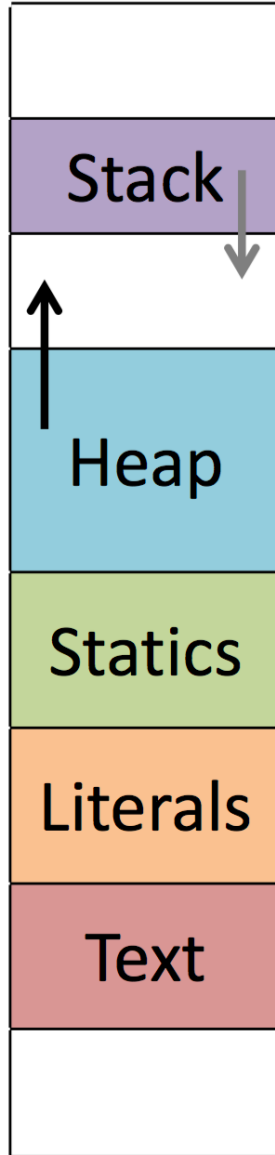
"Top" of stack
(lowest address)



2^N-1

Addresses

0



Stack

Function context, local variables
(**compiler-managed**)

Heap

Persistent data structures
(**programmer-managed**)

Statics

Global variables

Literals

String literals

Text

Program code (instructions)

Dynamic memory allocation

- `void *malloc(size_t size);`
- `void free(void *ptr);`

zips.c

- We will malloc() a zip code.
- I.e., an array of ints

zips.c

- We will malloc() an array zip codes.
- I.e., a 2d array of ints

Uninitialized memory

- *Uninitialized read error*
- Technically undefined behavior
 - Immediate crash
 - Crash later
 - Weird, hard to debug behavior
 - Et cetera
- Often reads whatever was there previously

NULL pointers

- Pointer sentinel value (represented as all zero bits on most machines)
- `#define ((void *) 0)`
- E.g., can be returned by `malloc()` if there is insufficient memory
- Can also be used in APIs to omit optional values

How best to allocate arrays?

- Automatic allocation to stack:
- `int zip[5];`
- Dynamic allocation to heap:
- `int *zip = malloc(sizeof(int) * 5);`

largearray.c

How best to allocate arrays?

- When does an automatically allocated array get deallocated?

How best to allocate arrays?

- When does an automatically allocated array get deallocated?
- What if we want the data to survive the function returning?

Problems with automatic allocation

- Cannot handle large arrays (unsafe for variable or user-controlled input)
- Cannot survive the function returning

Use stack when...

- The size of the array is a constant
- AND
- The size of the array small
- AND
- The data does not need to survive the function returning

Use heap when...

- The size of the array is variable
- OR
- The size of the array large
- OR
- The data needs to survive the lifetime of the function

Typical stack sizes

- Linux x86: 8mb
- MacOS: 8mb
- Windows: 1mb

Allocation Exercise

Debugging

- What do we do when our code has a bug?
- What have we done?

Debugging

- What do we do when our code has a bug?
- What have we done?
- We will learn some dedicated debugging tools:
 - gdb (GNU debugger)
 - valgrind
- Other languages have debugging tools too, but C needs them more than others

debugdemo.c

TODO

- Assignment 2 assigned due October 7