

# Memory, the Heap, and malloc

---

CS 2130: Computer Systems and Organization 1

November 9, 2022

# Announcements

- Homework 8 due next Wednesday at 11pm



# Header Files

C header files: `.h` files

- Written in C, so look like C
- Only put header information in them
  - Function headers
  - Macros
  - `typedefs`
  - `struct` definitions
- Essentially: information for the **type checker** that does not produce any actual binary
- `#include` the header files in our `.c` files

# Big Picture

## Header files

- Things that tell the type checker how to work
- Do not generate any actual binary

## C files

- Function definitions and implementation
- Include the header files

# Including Headers

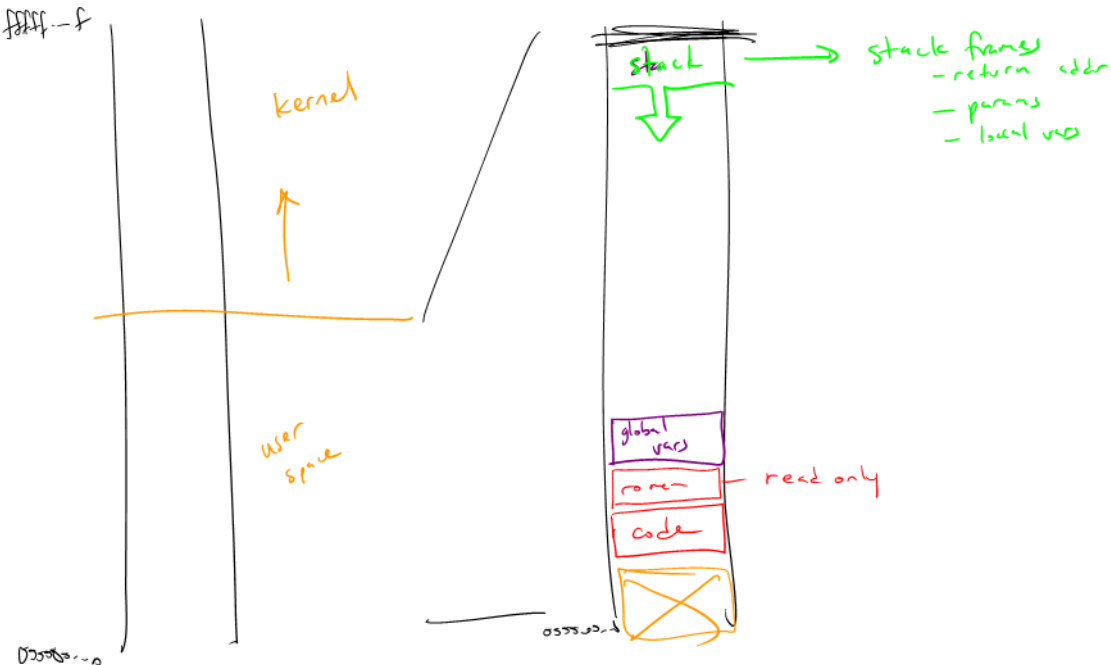
```
#include "myfile.h"
```

- Quotes: look for a file where I'm writing code
- Our header files

```
#include <string.h>
```

- Angle brackets: look in the standard place for includes
- Code that came with the compiler
- Likely in `/usr/include`

# Memory



```
int x = 3;  
int y[100000];
```

```
int *foo() {  
    int z[10000];
```

```
    return z; - escape pointer
```

```
}
```



**The heap:** unorganized memory for our data

- Most code we write will use the heap
- *Not a heap data structure...*

# The Heap: Requesting Memory

```
void *malloc(size_t size);
```

- Ask for **size** bytes of memory
- Returns a (**void \***) pointer to the first byte
- It does not know what we will use the space for!
- Does not erase (or zero) the memory it returns

# malloc Example

```
typedef struct student_s {
    const char *name;
    int credits;
} student;

student *enroll(const char *name, int transfer_credits) {
    student *ans = (student *)malloc(sizeof(student));
    ans->name = name;
    ans->credits = transfer_credits;
    return ans;
}
```

# The Heap: Freeing Memory

Freeing memory: `free`

```
void free(void *ptr);
```

- Accepts a pointer returned by `malloc`
- Marks that memory as no longer in use, available to use later
- You should `free()` memory to avoid *memory leaks*