

# C, Memory

CS 2130: Computer Systems and Organization 1

April 8, 2026

# Announcements

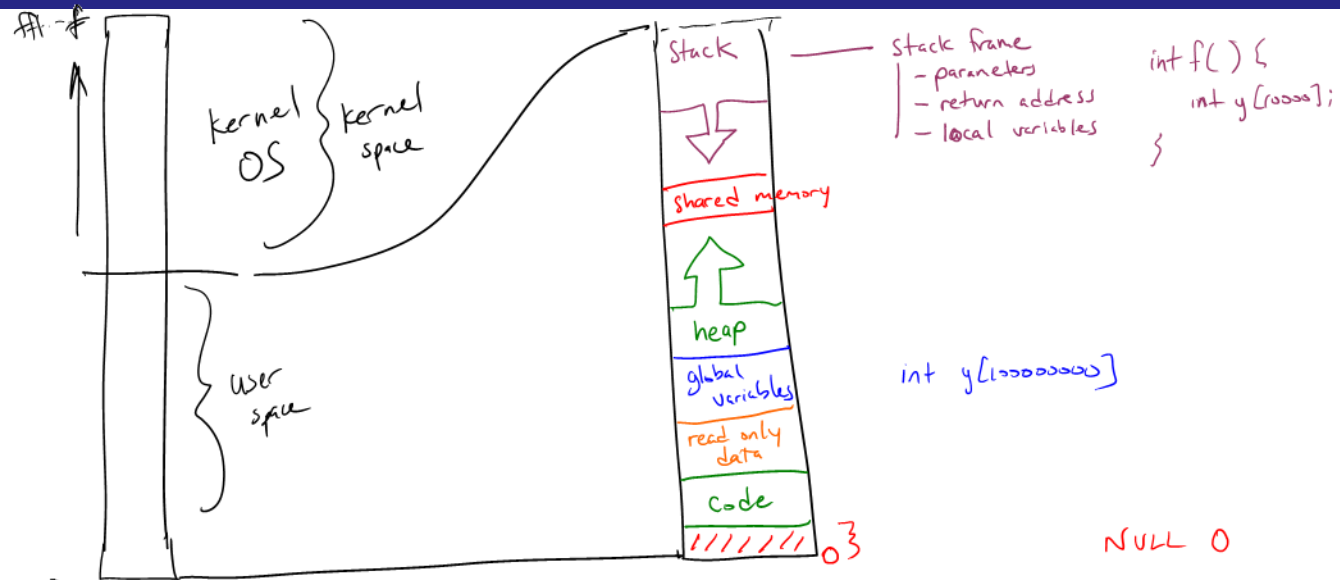
- Homework 8 due Monday on Gradescope

# Revisiting C Examples

- header example
- `string.h`
- variadic functions

# Memory

## Segments



# An Interesting Stack Example

```
int *makeArray() {
    int answer[5];
    return answer;
}

void setTo(int *array, int length, int value) {
    for(int i=0; i<length; i+=1)
        array[i] = value;
}

int main(int argc, const char *argv[]) {
    int *a1 = makeArray();
    setTo(a1, 5, -2);
    return 0;
}
```

# The Heap

**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

# 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*