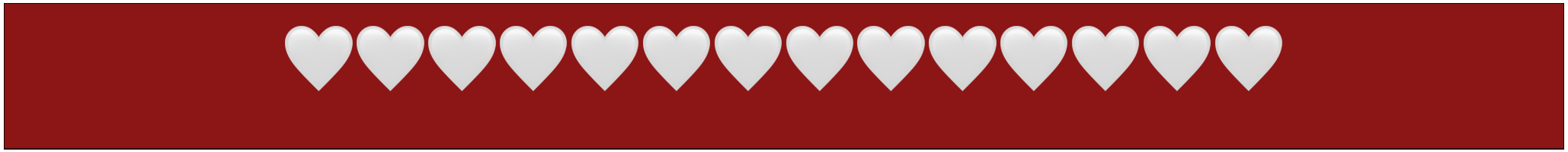
CS107 Lecture 26
Wrap-Up / What's Next?



We've covered a lot in just 10 weeks. Let’s wax nostalgic.

CS107's Topics and Big Questions

1. **Bits and Bytes**: How can a computer represent **`int`** values? or **`float`**s?
2. **Characters and C Strings**: How can a computer represent and manipulate more complex data, like text?
3. **Pointers, Stack and Heap Memory**: How can we effectively manage all forms of memory in our programs?
4. **Generics**: How can we leverage our understanding of memory and data representation to write code that works with **any** data type?
5. **Assembly**: How does a computer compile and execute C programs to assembly code? What does assembly code look like?
6. **Heap Allocators**: How do core memory allocation functions like **`malloc`** and **`free`** work? Are the built-in versions always good enough?

Baby's First Programs

```
int main(int argc, char *argv[]) {
   bool x = argc > 2 && argv[argc - 1][0] != 'A';
   if (x) {
      printf("Hello, world!\n");
   } else if (argc > 5) {
      printf("Greetings, traveler!\n");
   } else {
      printf("Salut, monsieur!\n");
   }
   printf("Adios!\n");
   return 0;
}
```

```
int main(int argc, char *argv[]) {
   printf("This program got %d argument(s).\n", argc);
   for (size_t i = 0; i < argc; i++) {
      printf("Argument %zu: %s\n", i, argv[i]);
   }
   return 0;
}
```

```
unsigned int absolute_value_bitwise(int value) {
   int mask = value >> (sizeof(value) * CHAR_BIT - 1);
   return (value ^ mask) - mask;
}
```

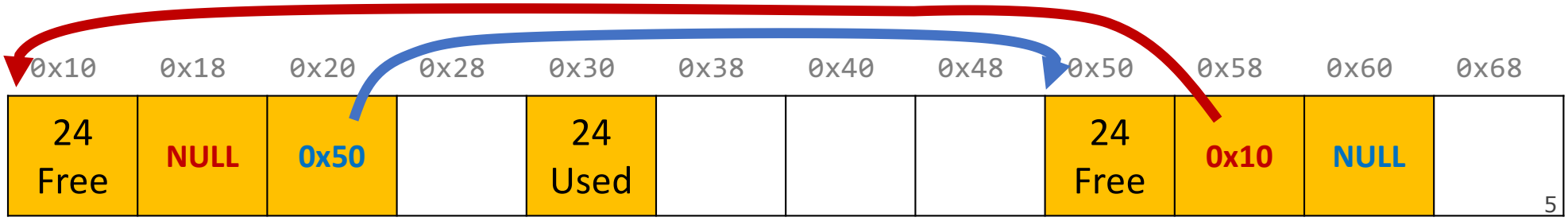
```
void reverse(char s[]) {
   if (s[0] == '\0') return;
   size_t lh = 0, rh = strlen(s) - 1;
   while (lh < rh) {
      char temp = s[lh];
      s[lh] = s[rh];
      s[rh] = temp;
      lh++;
      rh--;
   }
}
```

```
void rotate(void *front, void *separator, void *end) {
   assert(front <= separator && separator <= end);
   size_t width = (char *)end - (char *)front;
   size_t prefix_width = (char *)separator - (char *)front;
   size_t suffix_width = width - prefix_width;
   if (prefix_width == 0 || suffix_width == 0) return;
   char temp[prefix_width];
   memcpy(temp, front, prefix_width);
   memmove(front, separator, suffix_width);
   memcpy((char *)end - prefix_width, temp, prefix_width);
}
```

Baby's Grown Up: Where Did The Time Go?

The idea **extends our implicit allocator** but stores pointers to the **next** and **previous** free blocks using the first 16 bytes of the free block's payload.

- In response to each **`malloc`** call, we:
  - **search our list of free blocks**—that is, the **explicit free list**—to find an appropriately sized one,
  - **update its header** to note the block is now allocated,
  - **splice** the now allocated block **out of the free list**, and potentially
  - **create a node** out of any substantial excess payload and **thread it back into the free list**.
- In response to each **`free`** call, we:
  - **update its header** to note the block is now free, and
  - **thread it into the free list** so it's discoverable by **`malloc`**



CS107 Learning Goals: Achieved!

What do we hope you get out of the course? We'd love for students to:

attain **fluency** with

- ✅ • pointers and memory, and how to use them effectively in C
- ✅ • an executable's address space and runtime behavior

develop **competency** with

- ✅ • the translation between C code and assembly
- ✅ • implementing programs that respect the limits of computer arithmetic
- ✅ • the ability to identify bottlenecks and improve runtime performance
- ✅ • the ability to navigate your own Unix development environment
- ✅ • ethical frameworks to consider when designing and implementing software

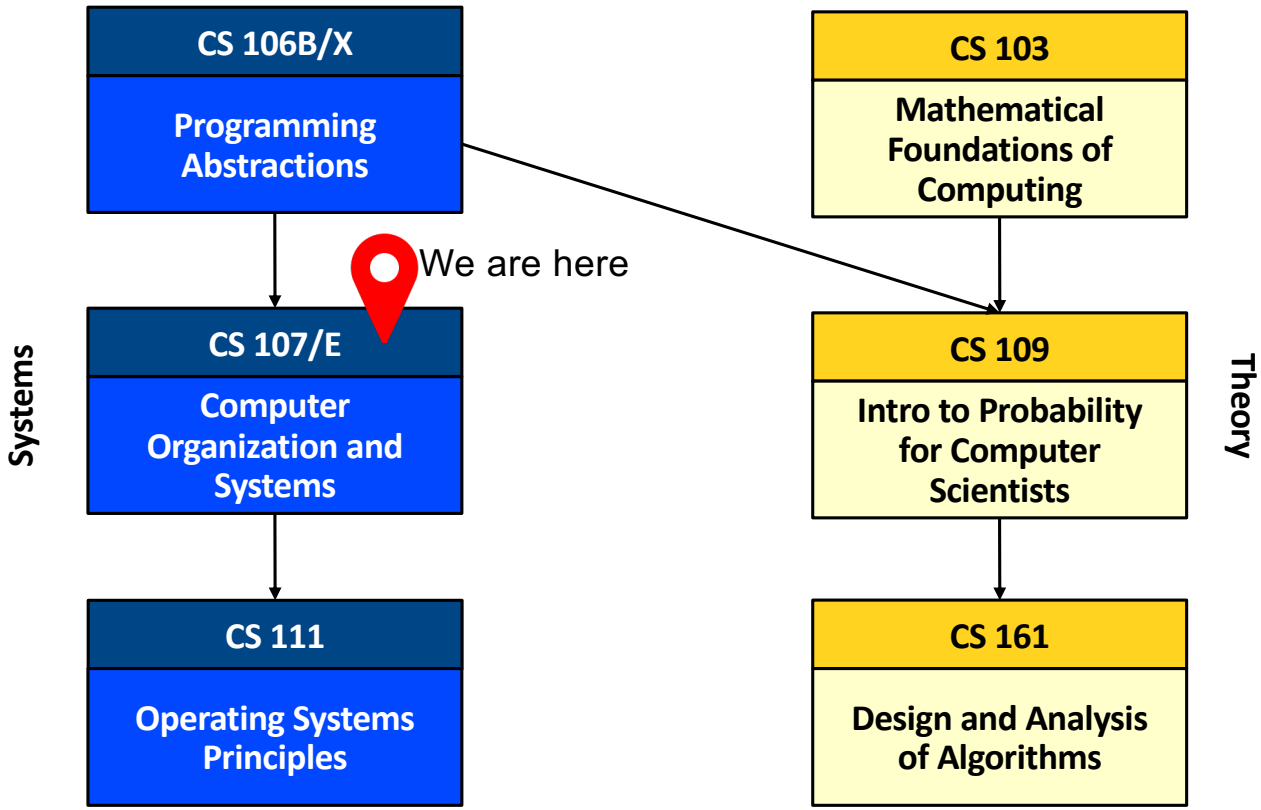
gain **exposure** to

- ✅ • the basics of computer architecture
- ✅ • compilers and assemblers and how they work

Sebastian C

https://www.youtube.com/watch?v=G7LJC9vJluU

Where Are We? Where To Go?



CS 111: Operating Systems Principles

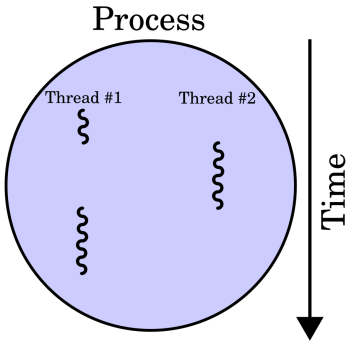
- How can programs **perform multiple tasks concurrently** and **share resources between those tasks**?
- Why does **every process think it has access to all memory addresses** if it needs them?
- How can we **design and implement a filesystem** to store persistent data?
- How can we implement **core operating system services** like **processes**, **threads**, **filesystems**, and **virtual memory**?


Troccoli


Mazieres


Rosenblum


Ousterhout



Looking to the Future: Other Courses

- **CS112**: Operating Systems Projects
- **CS212**: Operating Systems
- **CS143:** Compilers
- **CS144:** Networking
- **CS145:** Databases
- **CS149:** Parallel Programming
- **CS152:** Trust and Safety Engineering
- **CS155:** Computer and Network Security
- **CS181:** Computers, Ethics, and Public Policy
- **CS182:** Ethics, Public Policy, and Technological Change
- **CS221:** Artificial Intelligence
- **CS246:** Mining Massive Datasets
- **EE108:** Digital Systems Design
- **EE180:** Digital Systems Architecture

Thank you!