CIS241

System-Level Programming and Utilities

C - Memory and String Functions

Erik Fredericks, frederer@gvsu.edu

Fall 2025

> **Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers**

New include for strings!

#include <string.h>

Clearing memory

Zero-ing out memory:

```
int* data = (int*)malloc(sizeof(int) * 32);
for(int i = 0; i < 32; i++){
    data[i] = 0
}
```

Alternative:

- `memset(void* s, int c, size_t n);`
- Fills `n` bytes of `s` with byte `c`

Does this work?

- `memset(data, 1, 32 * sizeof(int));`

Memory functions

Copy an array:

```
int* data = (int*)malloc(sizeof(int) * 32);
int* copy = (int*)malloc(sizeof(int) * 32);
for(int i = 0; i < 32; i++){
    copy[i] = data[i];
}
```

Or:

```
memcpy(void* dest, void* src, size_t n);
```

Copy *n* bytes:

```
memcpy(copy, data, 32 * sizeof(int));
```

Another alternative for copying an array

```
memmove(void* dest, void* src, size_t n);
```

Copy *n* bytes, allowing for overlapping buffers:

```
memmove(copy, data, 32 * sizeof(int));
```

Strings!

How are they different?

- `\0` - null terminator!

String functions we'd like to have?

Concatenation

```
strcat(char *s1, char* s2);
```

Appends copy of `s2` to `s1`

- What happens if `s1` isn't big enough?
- We are writing into arbitrary memory…

```
strncat(char* s1, char* s2, size_t n);
```

- Same, but will only copy *n* chars + terminator

Concatenation

Can also use string formatting

- `sprintf(char *s1, char* format_str, ...);`
- Same idea and formatting as `printf`
- Stores result in `s1`, adds terminator
- Returns new length of `s1`

```
sprintf(s, "x = %d", 5);
```

Copying

`strcpy(char* dest, char* src);`

- Copies `src` to `dest`, including '`\0`', too.
- Make sure `dest` has enough space!

`strncpy(char* dest, char* src, size_t n);`

- Copy up to *n* chars
- Will add `'\0'`s to fill if `src` is too short
- Otherwise no terminators

Length

```
strlen(char* s);
```

- Returns length of string
- How does this work?
  - Returns number of characters before terminator!
  - (must be null-terminated!)

Comparison

```
int strcmp(char* s1, char* s2);
```

- Lexicographically compares strings
  - Return 0 if strings are the same
  - Returns negative num `if s1 < s2`
  - Returns positive num `if s1 > s2`

```
int strncmp(char* s1, char* s2, size_t n);
```

- Same, but limits to *n* chars (or to `\0`)

Searching

```
char* strchr(char *s, int c);
```

- Searches for first instance of char `c` in string
- Returns pointer to first instance
- Returns `NULL` if `c` not found

```
char* strrchr(char *s, int c);
```

- Returns last instance of `c` in `s` (or `NULL` if none)

Searching

```
char* strstr(char* s1, char* s2);
```

- Searches for substring `s2` in `s1`.
- Returns pointer to first instance, or `NULL` if not found