CPSC 259: Data Structures and Algorithms for Electrical Engineers

Strings

Learning Goal

- Become familiar with Strings in C

fgets and sscanf

- The fgets function:
  - Reads a line from the specified stream and stores it into the string pointed to by str.
  - It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.
  - On success, returns str, else returns a null pointer.

```
char *fgets(char *str, int n, FILE *stream)
```

- The sscanf function:
  - reads formatted input from a string.
  - On success, returns the number of arguments returned, else 0 or EOF

```
int sscanf(const char *str, const char *format, ...)
```

See fgets_and_sscanf.c

Strings

- What is a Cstring?
  - An array (or string) of char that terminates in a null character
  - Null character = ‘\0’ (single quotes!)
- Different ways to create strings
  - `char an_array[6] = {'H', 'e', 'l', 'l', 'o', '\0'};`
  - `char string[SOMESIZE] = "A string of char";`
    - When we do this, we automatically get a null char at the end
  - `char * another_string = "A string of characters";`
    - When we do this, we automatically get a null char at the end.

String and C

- C provides a group of functions for string processing
- Declared in the header file string.h

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

- Calculating length

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

  – * size_t is an unsigned data type defined by several C/C++ standards

```
char string[] = "Hello there";
int length;
length = strlen(string);
printf(“string '%s' has %d letters\n", string, length);
```

The string 'Hello there' has 11 letters

String and C

- Comparing

```
Integers
int a = 6;
int b = 7;
if (a == b) { … }
```

```
Characters

char a = ‘a’;
char b = ‘b’;
if (a == b) { … }
```

- To compare strings in C we can use:

```
int strcmp (const char *str1, const char *str2);
int strncmp (const char *str1, const char *str2, size_t num);
```

```
char string1[] = "Hello";
char string2[] = "Hello there";
int length;

length = strlen(string1);
if ( strncmp(string1, string2, length) == 0) {
	printf("The first %d letters of %s and %s are the same\n", length, string1, string2);
} else {
	printf("Not the same\n");
}
```

String and C

- Searching
  - How can we check if a string contains another string?

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

  - Locates the first occurrence in the string pointed to by s1 of the sequence of bytes (excluding the terminating null byte) in the string pointed to by s2

```
char string1[] = "Hello";
char string2[] = "123 Hello there";
char * result = NULL;

result = strstr(string2, string1);
printf("%s\n", result);

result = strstr(string2, "Zoinks");
if (result == NULL) {
  printf("Not found\n");
}
```

```
Hello there
Not found
```

String and C

- Concatenating
  - "Hello" + " there" = "Hello there"

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

- Appends not more than n bytes from the string pointed to by s2 to the end of the string pointed to by s1. The initial byte of s2 overwrites the null byte at the end of s1. A terminating null byte is always appended to the result.
- The strncat() function returns s1.

```
char * empty_string;
char a_long_string[128] = "These ";
strcat (a_long_string,"strings ");
strcat (a_long_string,"are ");
empty_string = strcat(a_long_string, "concatenated!");
printf("%s\n", empty_string);
```

These strings are concatenated!

String and C

- Copying

```
char * strncpy(char * s1, const char * s2, size_t n);
```

  - Copies not more than n bytes from the string pointed to by s2 to the string pointed to by s1
  - Returns s1.

```
char a_str[] = "Take the test.";
int length = strlen(a_str);
char * other_str = (char *) malloc(length + 1); // Why +1?
strcpy(other_str, a_str);
a_str[0] = 'M';
printf("a_str = %s\nother_str = %s\n", a_str, other_str);
```

a_str = Make the test.
other_str = Take the test.

Learning Goal Revisited

- Become familiar with Strings in C