

EECS 22: Advanced C Programming

Lecture 25

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 25: Overview

- Course Administration
 - Reminder: Final course evaluation
- String Operations
 - Using pointers
- Standard C Library
 - Functions provided in `string.h`, `stdlib.h`
- Math Library
 - Functions provided in `math.h`

Course Administration

- Final Course Evaluation
 - Open until end of 10th week (Sunday night)
 - Nov. 22, 2017, through Dec. 10, 2017, 11:45pm
 - Online via EEE Evaluation application
- Mandatory Evaluation of Course and Instructor
 - Voluntary
 - Anonymous
 - Very valuable
- Please spend 5 minutes for this survey!
 - Your feedback is appreciated!

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

3

String Operations

- String Operations using Pointers
 - Example: String length

```
int Length(char *s)
{
    int l = 0;
    char *p = s;

    while(*p != 0)
    { p++;
      l++;
    }
    return l;
}
```

```
char s1[] = "ABC";
char s2[] = "Hello World!";

printf("Length of %s is %d\n",
       s1, Length(&s1[0]));
printf("Length of %s is %d\n",
       s2, Length(&s2[0]));
```

```
Length of ABC is 3
Length of Hello World! is 12
```

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

4

String Operations

- String Operations using Pointers

- Example: String length

```
int Length(char *s)
{
    int l = 0;
    char *p = s;

    while(*p != 0)
    { p++;
      l++;
    }
    return l;
}
```

```
char s1[] = "ABC";
char s2[] = "Hello World!";

printf("Length of %s is %d\n",
      s1, Length(&s1[0]));
printf("Length of %s is %d\n",
      s2, Length(s2));
```

```
Length of ABC is 3
Length of Hello World! is 12
```

- Array and pointer types are equivalent

- `s2` is an array, but can be passed as a pointer argument
- Character array `s2` is same as character pointer `&s2[0]`

String Operations

- String Operations using Pointers

- Example: String length

```
int Length(char *s)
{
    int l = 0;
    char *p = s;

    while(*p != 0)
    { p++;
      l++;
    }
    return l;
}
```

```
char s1[] = "ABC";
char *s2 = "Hello World!";

printf("Length of %s is %d\n",
      s1, Length(s1));
printf("Length of %s is %d\n",
      s2, Length(s2));
```

```
Length of ABC is 3
Length of Hello World! is 12
```

- Array and pointer types are equivalent

- `s1` is an array of characters, `s2` is a pointer to character
- Both `s1` and `s2` can be passed to character pointer `s`

String Operations

- String Operations using Pointers

- Example: String length

```
int Length(char s[])
{
    int l = 0;
    char *p = s;

    while(*p != 0)
    { p++;
      l++;
    }
    return l;
}
```

```
char s1[] = "ABC";
char *s2 = "Hello World!";

printf("Length of %s is %d\n",
       s1, Length(s1));
printf("Length of %s is %d\n",
       s2, Length(s2));
```

```
Length of ABC is 3
Length of Hello World! is 12
```

- Array and pointer types are equivalent

- `s1` is an array of characters, `s2` is a pointer to character
- Both `s1` and `s2` can be passed to character array `s`

String Operations

- String Operations using Pointers

- Example: String copy

```
void Copy(
    char *Dst,
    char *Src)
{
    do{
        *Dst = *Src;
        Dst++;
    } while(*Src++);
}
```

```
char s1[] = "ABC";
char s2[] = "Hello World!";

printf("s1 is %s, s2 is %s\n",
       s1, s2);

Copy(s2, s1);
printf("s1 is %s, s2 is %s\n",
       s1, s2);
```

```
s1 is ABC, s2 is Hello World!
s1 is ABC, s2 is ABC
```

- Passing pointers as arguments to functions

- Function can modify caller data by pointer dereferencing
- **Passing pointers = Pass by reference!**

String Operations

- String Operations using Pointers

- Example: String copy

```
void Copy(
    char *Dst,
    const char *Src)
{
    do{
        *Dst = *Src;
        Dst++;
    } while(*Src++);
}
```

```
char s1[] = "ABC";
char s2[] = "Hello World!";

printf("s1 is %s, s2 is %s\n",
        s1, s2);

Copy(s2, s1);
printf("s1 is %s, s2 is %s\n",
        s1, s2);
```

```
s1 is ABC, s2 is Hello World!
s1 is ABC, s2 is ABC
```

- Passing pointers as arguments to functions

- Function can modify caller data by pointer dereferencing
- Type qualifier **const**:
Modification by pointer dereferencing *not* allowed!

String Operations

- String Operations using Pointers

- Example: String copy

```
void Copy(
    const char *Dst,
    const char *Src)
{
    do{
        *Dst = *Src;
        Dst++;
    } while(*Src++);
}
```

```
char s1[] = "ABC";
char s2[] = "Hello World!";

printf("s1 is %s, s2 is %s\n",
        s1, s2);

Copy(s2, s1);
printf("s1 is %s, s2 is %s\n",
        s1, s2);
```

```
s1 is ABC, s2 is Hello World!
s1 is ABC, s2 is ABC
```

Error!
Write access to
const data!

- Passing pointers as arguments to functions

- Function can modify caller data by pointer dereferencing
- Type qualifier **const**:
Modification by pointer dereferencing *not* allowed!

Standard Library

- Standard C library
 - standard library supplied with every C compiler
 - predefined standard functions
 - e.g. `printf()`, `scanf()`, etc.
- C library header files
 - input/output function declarations `#include <stdio.h>`
 - standard function declarations `#include <stdlib.h>`
 - string function declarations `#include <string.h>`
 - others
- C library linker file
 - contains standard function definitions (pre-compiled)
 - library file `libc.a`
 - compiler links against the standard library by default (no need to supply extra options)

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

11

Standard Library

- Functions declared in `string.h` (part 1/2)
 - `typedef unsigned int size_t;`
 - type definition for length of strings
 - `size_t strlen(const char *s);`
 - returns the length of string `s`
 - `int strcmp(const char *s1, const char *s2);`
 - alphabetically compares string `s1` with string `s2`
 - returns -1 / 0 / 1 for less-than / equal-to / greater-than
 - `int strncmp(const char *s1, const char *s2, size_t n);`
 - same as previous, but compares maximal `n` characters
 - `int strcasecmp(const char *s1, const char *s2);`
 - `int strncasecmp(const char *s1, const char *s2, size_t n);`
 - same as string comparisons above, but case-insensitive

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

12

Standard Library

- Functions declared in `string.h` (part 2/2)
 - `char *strcpy(char *s1, const char *s2);`
 - copies string `s2` into string `s1`
 - `char *strncpy(char *s1, const char *s2, size_t n);`
 - copies maximal `n` characters of string `s2` into string `s1`
 - `char *strcat(char *s1, const char *s2);`
 - concatenates string `s2` to string `s1`
 - `char *strncat(char *s1, const char *s2, size_t n);`
 - concatenates maximal `n` characters of string `s2` to string `s1`
 - `char *strchr(const char *s, int c);`
 - returns a pointer to the first character `c` in string `s`, or `NULL` if not found
 - `char *strrchr(const char *s, int c);`
 - returns a pointer to the last character `c` in string `s`, or `NULL` if not found
 - `char *strstr(const char *s1, const char *s2);`
 - returns a pointer to the first appearance of `s2` in string `s1` (or `NULL`)

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

13

Standard Library

- Functions declared in `stdlib.h` (selected subset)
 - `int abs(int x);`
 - `long int labs(long int x);`
 - return the absolute value of a (long) integer `x`
 - `int rand(void);`
 - return a random value in the range 0 - `RAND_MAX`
 - `RAND_MAX` is a constant integer (e.g. 32767)
 - `void srand(unsigned int seed);`
 - initialize the random number generator with value `seed`
 - `void exit(int result);`
 - exit the program with return value `result`
 - `void abort(void);`
 - abort the program (with an error result)

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

14

Standard Library

- Standard Math Library
 - standard library supplied with every C compiler
 - predefined mathematical functions
 - e.g. $\cos(x)$, \sqrt{x} , etc.
- Math library header file
 - contains math function declarations
 - `#include <math.h>`
- Math library linker file
 - contains math function definitions (pre-compiled)
 - library file `libm.a`
 - compiler needs to link against the math library
 - use option `-llibraryname`
 - Example: `gcc MathProgram.c -o MathProgram -lm`

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

15

Standard Library

- Functions declared in `math.h` (part 1/2)
 - `double sqrt(double x);` \sqrt{x}
 - `double pow(double x, double y);` x^y
 - `double exp(double x);` e^x
 - `double log(double x);` $\log(x)$
 - `double log10(double x);` $\log_{10}(x)$
 - `double ceil(double x);` $\lceil x \rceil$
 - `double floor(double x);` $\lfloor x \rfloor$
 - `double fabs(double x);` $|x|$
 - `double fmod(double x, double y);` $x \bmod y$

EECS22: Advanced C Programming, Lecture 25

(c) 2017 R. Doemer

16

Standard Library

- Functions declared in `math.h` (part 2/2)

- <code>double cos(double x);</code>	<i>cos(x)</i>
- <code>double sin(double x);</code>	<i>sin(x)</i>
- <code>double tan(double x);</code>	<i>tan(x)</i>
- <code>double acos(double x);</code>	<i>acos(x)</i>
- <code>double asin(double x);</code>	<i>asin(x)</i>
- <code>double atan(double x);</code>	<i>atan(x)</i>
- <code>double cosh(double x);</code>	<i>cosh(x)</i>
- <code>double sinh(double x);</code>	<i>sinh(x)</i>
- <code>double tanh(double x);</code>	<i>tanh(x)</i>