Advanced C Programming

Real Time Programming

Contents

- typedef’s
- Preprocessor conditionals
- Pointer Variables
- String
- Pointer Arithmetic

Good Coding Practices

- Signed (good) vs Unsigned (bad) Math
  - for physical calculations
- Use Braces

```
{
    Always
}
```

- Simple Readable Code
  - Concept of "Self Documenting Code"
  - Code as if your grandmother is reading it
- Never use Recursion
  - (watch your stack)
- Treat Warnings as Errors

*Disclaimer: Not all code in this presentation follows these practices due to space limitations

Typedef's

Using Naturally Named Data Types

Why Typedef?

- You use variable with logical names, why not use data types with logical names?
- Is an “int” 8-bits or 16-bits? What’s a “long”? Better question: why memorize it?
- Most integer data types are platform dependent!!!
- typedef’s make your code more portable.

typedef.h Example

```
typedef unsigned char  u8;
typedef signed char    s8;
typedef unsigned short u16;
typedef signed short   s16;
typedef unsigned long  u32;
typedef signed long    s32;
```

In your code:

```
unsigned char variable;
```

Is replaced with:

```
u8 variable;
```

Preprocessor conditionals

#ifdef macro

only includes the subsequent code if macro was #defined.

#ifndef macro

only includes the subsequent code if macro was not #defined

if expression

only includes the subsequent code if expression is true

Pointer Variables

A **pointer** is a variable that stores the **memory address** of another variable.

Pointer Variables

- Explanation char *is a character pointer type. p is called a character pointer variable. stores the memory address of a character (the first character (’H’) of the string "Hello")
- Example (A Character Pointer) Pointers, Arrays, and Strings int main(void) { char s[6] = "Hello"; char *p; p = s; printf("%s n", s); printf("%s n", p); return 0; }

String

A String in C programming is a sequence of characters terminated with a null character ‘\0’. The C String is stored as an array of characters. The difference between a character array and a C string is that the string in C is terminated with a unique character ‘\0’.

C String Declaration Syntax

Declaring a string in C is as simple as declaring a one-dimensional array. Below is the basic syntax for declaring a string.

Char string_name[size];

Copying Strings

- A String is an array of characters one character after the other in memory.
- Strings need to be copied character by character loop that stops when the end of string is reached.
- Example Pointers, Arrays, and Strings int main(void) { char b[8], a[6] = "Hello"; int i = 0; do { b[i] = a[i]; } while (a[i++] != ’ 0’) printf("%s n", b); return 0; }.
- String a gets copied to b character by character Integer i counts up the current index into the array ’0’denotes the end of the string needs to be copied before finishing the loop

Integer Data Types: Valid Ranges

- u8 : 0 - 255
- s8 : -128 - 127
- u16 : 0 - 65535
- s16 : -32768 - 32767
- u32 : 0 - 4.294967295e9
- s32 : -2.147483648e9 - 2.147483647e9

Note: ranges given are decimal.

Pointer Arithmetic

- Pointers store memory addresses just numbers telling the processor which memory cell to access.
- Adding 1 to a pointer makes it point to the next memory location.
- Subtracting 1 from a pointer makes it point to the previous memory location
- Subtracting two pointers from each other shows how much space is between the memory locations pointed to by the pointers.
- Pointers “know” the sizes of the variables they point to.
- adding to an int pointer will probably result in a different address than adding to a char pointer