Embedded Programming: C resumes

Nis Sarup

31. januar 2011

1 Language Basics

1.1 Characteristics of C

- Officially published in 1978
- Source code portability
- operates "close to the machine"
- Efficiency
- Compiler very compact
- C Library provides functions not in the core
- Library (mostly) written in portable C

1.2 The Structure of C Programs

- Statements, Blocks, Functions
- Starts with main()
- Prior declaration needed for compiler
- Definition of a function also declaration
- No nested functions

1.3 Source Files

- Internal structure:
  - Preprocessor Directives
  - Global Declarations
  - Function definitions
- Suffix for source files: .c

- Header files can contain commonly used Preprocessor Directives, suffix: .h
- Header files can be included in source files with #include
- Can i turn include other files
- White space is unimportant, except in Preprocessor Directives

1.4 Comments

- /* Block comments */
- // endline comments
- Comments does not work inside String Literals: "I am /* not a comment*/"
- Commenting out parts of a program often used.

1.5 Character Sets

- Two sets according to environment:
  - Translation environment (compilation): source character set
  - Execution Environment (running program): execution character set

1.6 Identifiers

- Names of variables, functions, macros, structures etc defined in a C program.
- 37 reserved keywords in C
- Format:
  - a-z, A-Z
  - _
  - 0-9, but not as the first character
  - Universal Characters from other languages
- __func__ will evaluate to the name of the current function

1.7 Identifier Scope

- File Scope: Inside the translation unit
- Block Scope: Often the same as function scope, but sometimes smaller
- Function Prototype Scope: parameter names in function prototypes
- Function Scope: Inside a function

1.8 How the C Compiler Works

- Compiler compiles source into:
  - Translation units
  - Object files
  - And then into the Executable file

2 Types

- Object refers a location in memory
- Named objects are called variables
- sizeof(type) will yield the storage size of the object or type in bytes

2.1 Typology

- Basic type
  - Integers
  - Floating point
- Enumerated types
- The type void
- Derived types:
  - Pointer
  - Array
  - Structure
  - Union
  - Function

2.2 Integer Types

Type	**Min.**	**Max.**
char	same as below	same as below
unsigned char	0	255
signed char	-128	127
int	-32,768 or -2,147,483,648	32,767 or 2,147,483,647
unsigned int	0	65,535 or 2,147,483,647
short	-32,768	32,767
unsigned short	0	65,535
long	-2,147,483,648	2,147,483,647
unsigned long	0	4,294,967,295
long long	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long	0	18,446,744,073,709,551,615

2.3 Floating-Point Types

- Represents nonintegers with decimal points in any position

Type	Range	Min. positive value	Precision
float	$\pm 3.4 \cdot 10^{38}$	$1.2 \cdot 10^{-38}$	6 digits
double	$\pm 1.7 \cdot 10^{308}$	$2.3 \cdot 10^{-308}$	15 digits
long double	$\pm 1.1 \cdot 10^{4932}$	$3.4 \cdot 10^{4932}$	19 digi

2.4 Enumerated Types

- enum [identifier] { enumerator-list };
- enum color { black, red, green, yellow, blue, white=7, gray };
- A list of a created types possible values
- Can also be used to define constants instead of using #define:
  - enum { OFF, ON, STOP = 0, GO = 1, CLOSED = 0, OPEN = 1 };

2.5 The Type void

- Cannot be used for variables or constants
- A function with no return values is of the type *void*
- If a functions prototype is declared with *void* as parameter the compiler would issue an error if that function is ever called with parameters
- A void expression has no return value
- A pointer of type *void*$*$ represents the address of an object but not its type