**[[iBoard 电子学堂][第二卷 C程序设计语言 ]第一篇 C语言简介](http://www.cnblogs.com/xiaomagee/archive/2012/03/25/2417098.html)**

**一、C语言发展史**

C语言是一种程序设计语言，由于它既有高级语言的特性，又有低级语言的特性，所以它既能编写基于操作系统的大型应用程序，也能编写直接操作硬件的程序，也就是我们称之为“裸奔”的程序。

C语言的发明是一件伟大的事。

它的原型是 ALGOL 60语言。1963年，剑桥大学将 ALGOL 60语言发展成为 CPL（Combined Programming Language）语言。1967年，剑桥大学的Matin Richards 对 CPL 语言进行了简化，于是产生了 BCPL（Basic Combined Programming Language）语言。

1970年，贝尔实验室的Kenneth Lane Thompson将 BCPL 语言进行修改，将他称之为“B语言”，并用B语言写出了第一 UNIX 操作系统。

1973年，美国贝尔实验室的Dennis MacAlistair Ritchie（伟大的DMR）在B语言的基础上，设计出了一种新的语言，他取了BCPL 的第二个字母，命名为“C语言），到这里C语言终于诞生了。

C语言是于 UNIX 操作系统一同成长起来的，1977年，为了推广 UNIX 操作系统，DMR同志发表了不依赖于具体机器系统的C语言编译文本《可移植性的C语言编译程序》。1978年，Brian W.Kernighian和Dennis MacAlistair Ritchie（DMR）（如图一所示）出版了可谓C语言圣经的一本书《The C Programming Language》。从那以后，C语言越来越流行，可谓盛行几十年。

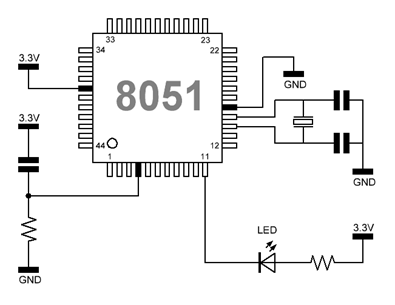
[](http://images.cnblogs.com/cnblogs_com/xiaomagee/201203/201203252210585622.png)

图一 伟大的 K&R

**二、第一个C语言程序**

市面上C语言教材比比皆是，但大部分都是面向个人计算机的。本教程以嵌入式应用为方向（面向8051、ARM等微控制器），来介绍C语言的一些基本特性。关于一些个人计算机、或者面向计算机操作系统等方面的特点（如文件系统，微机的输入输出等），我们不做研究。

我们先以8051为硬件平台，来设计第一个 C语言程序。嵌入式C语言编程是基于硬件的，要求程序员首先要了解我们的硬件平台组成，所以，我们首先要构造一个硬件平台。第一个C语言程序是在如基于8051的最小系统上完成的，其原理示意图如图二所示。

[](http://images.cnblogs.com/cnblogs_com/xiaomagee/201203/201203252211005981.png)

图二 8051单片机最小系统示意图

我们首先介绍下图二所示的最小系统，它包含如下几部分：

n **8051单片机：**这是一个TQFP封装的贴片8051单片机，共有44个引脚，以圆圈为索引点的地方为第一脚，然后逆时针旋转，直到数到倒数第一脚（也就是第44脚）。

n **供电系统：**电源就像电子系统的血液，只有供电后，电子系统才会“活“起来，本最小系统中的单片机采用 3.3V供电，如图二所示。

n **时钟电路：**通过外部晶体振荡器与匹配电容，与单片机内部的振荡电路连接，上电后晶体起振，然后为系统提供执行节拍。微处理器外部晶体一般在几MHz到几十MHz，本文采用 11.0592MHz，11.0592MHz虽然不规则，但它的确是一个常用的晶振频率。

n **复位电路：**单片机内部有很多寄存器，上电后有可能是不确定状态，我们必须通过上电复位，来完成所有寄存器的初始化。若内部有程序，则从第一条语句开始执行。

从图二中我们可以看到，除了最小系统外，我们外接了一个发光二级管。电源通过限流电阻，连接发光二级管（LED）至单片机的第11脚。**限流电阻**，顾名思义就是“阻碍电流流动“用的电阻，一般发光二级管我们只需要通过几个mA到十几个 mA的电流，就已经足够亮了；过大的电流会损坏二极管，所以我们需要增加一个电阻，来”限流“，限流电阻一般取几百欧姆到几千欧姆不等。通过查芯片手册我们可以知道，第11脚为单片机的P3.5口，我们通过以下代码，来使得LED闪烁。

[](http://images.cnblogs.com/cnblogs_com/xiaomagee/201203/201203252211019470.png)

代码一LED闪烁程序，是一段基于 8051单片机的最基本的程序。代码中，最左侧的行号仅仅是为了方便描述，实际代码中不需要键入行号。

第一行，#include <reg52.h> 表示包含8051单片机通用的寄存器定义，以#include <...> 这种格式的写法，是C语言包含头文件的固定模式。被包含的文件必须在编译器系统路径或者用户工程目录里，一般地，以 < > 包含的文件，表示编译器系统路径，以 “ ”表示的，为用户工程路径，如 #include “my\_header.h”。

C语言的注释规则：//（两条ASCII码格式的斜杠在一起）后的文字是C语言的注释，它只为了增加系统的可读性，不参与任何编译操作。C语言的注释包含两种，第一种为两条斜杠，即 // 后，直到本行结束，注释也跟着结束。第二种为 /\* … \*/ 内的字符或文字，以斜杠+星号开始，以星号+斜杠结束，这两个特征体之内的所有文字均为注释（即使包含换行）。优良的代码一般包含必要的注释，但是注释过多也会影响到代码的可读性和美观性，代码一的注释就显得繁琐，但是我们为了给大家讲解，不得已这样做。实际的工程中，并不需要每行都注释。

第二行为一个空行，我们编写代码时，通常以功能为单位，插入若干空行，以便增加代码的可读性。

第三行 sbit LED = P3^5; ，它其实并不是ANSI C的一种语法，是 KEILC编译器对8051微处理器的一种扩展，这条代码在桌面计算机上就编译不通过。sbit 关键字只能在函数体外定义8051单片机的可位寻址的某位，也可以通过 ^ 运算符，知名SFR寄存器中的某位。在这里，我们可以理解成通过 sbit，我们把 LED 这个关键字与P3口的第5位即P3.5捆绑在一起。

第五行为main 程序的主题。一个完整的 C语言工程，必须包含有 main函数，它是整个函数的入口，main函数前面的修饰词 void 说明这个函数不返回值，main函数后面小括号里的 void表明此函数不需要传递参数。

第六行为左大括号，在这里标志着一个函数的开始，大括号必须以配对的形式出现，第六行的大括号与地15行的右大括号为一对，他表示他们之间所有的代码都归属 main 函数所拥有。 C语言的大括号题表明了大括号中间代码的归属属性，如地9行后的大括号和地14行的大括号之间的函数，他们归属于 while(1)循环语句，只要 while(1)成立，他们就循环执行。

第七行为变量声明，int 是C语言的关键字，表示后面跟随的变量为整数，是单词integer 的缩写。由于此变量声明在函数体 main内，所以此变量只能在 main函数内调用，所以它也叫“局部变量“。C语言中，变量必须先声明后使用，一般地，我们在函数体的开始声明本函数所用的局部变量。

第九行与第十四行之间为while循环语句，while后面紧跟着的小括号内为条件表达式，条件表达式紧跟着循环体，当循环体大于一条语句时，我们需要用大括号把从前后把循环内容括起来。当条件表达式成立时（逻辑真，true），我们从头到尾执行一遍循环体，然后再测试while后面的条件表达式，如果成立则继续执行，如果不成立（条件表达式结果为假，false）则跳出循环体，往下顺序执行程序。这里我们需要让LED 一直闪烁下去，所以我们用 while(1)这种恒为真的条件表达式，来实现LED永远闪烁下去，直到断电。

第十行和第十二行为 for 循环语句。for语句后紧跟小括号，括号内分为三部分，他们之间用分号（；）隔开。第一部分为初始化部分，即i = 0;，这里我们把变量 i 赋值零；第二部分是控制循环的条件部分，首先CPU对条件部分求值，如果为真，将执行循环体一次，然后执行第三部分i ++ ，这里表示变量 i 累加一， 然后再测试条件部分是否为真，如果为真则再一次执行循环体，如果为假则跳出for循环，执行后面的语句。这里的循环体为空语句，我们利用空的 for 循环来让处理器原地踏步，目的是为了拖延时间（微处理器的速度始终比人眼睛快得多，如果不拖延时间，人类的眼睛根本分辨不出来LED闪烁）。

第十一行 LED = 1;  和第十三行 LED = 0;  为单片机I/O口赋值语句，等于一表示拉高IO口，使得 IO 输出高电平（电压近似等于系统供电电压）；等于零表示拉低 IO口，使得IO输出低电平（电压近似等于系统地电压），不断的让IO口拉高拉低，拉高和拉低过程中通过插入延时，所以我们就能看到LED一亮一灭的闪烁。

这里我们已经逐行讲解完了最基本的C语言程序，大家可以参考 8051卷的相关章节，把此代码通过 KEILC51 集成开发环境编译并通过下载器下载到 iBoard板载的8051 单片机中执行。

从代码一种我们可以看到，C语言是一种面向过程的语言；它需要编译成机器可以识别的机器指令后才能执行，也就是通常说的编译型语言（相对立的为解释性语言，如BASIC）。通过代码我们亦能感觉得到，C语言天生是操作硬件的利器，他可以直接操作硬件资源，这也是C语言在嵌入式系统中霸主地位的关键所在。

**三、C语言标准**

C语言成型过程中，经历了多次进化，完善过程中也存在很多分支，为了力求表达的准确性并尽量避免歧义，1983年，美国国家标准协会（American National Standards Institute，ANSI）组成了一个委员会，专门为了建立 C 语言的一套标准，经过漫长的、坎坷的、长达6年的努力，他们终于在 1989年完成了这套C标准，这个标准被称作 “ANSI C“，也叫 C89。值得欣慰的是，他们6年多的努力没有白费，直到现在为止，C89 也是众多C编译器所支持的最成熟的标准。

ANSIC 标准确立后，C语言标准在很长一段时间内没有改动。反而C++语言的标准建立过程中，带动了C语言新标准的出现，1994年，通过修正了C89标准的一些细节问题并增加了一些新内容后，一个新的一个C语言标准产生了，这个标准 1999年被采用，所以也叫 C99。

直到现在，C89 和 C99 一直是两个影响力较大的标准，绝大多数编译器支持C89 标准，部分编译器支持 C99 标准。嵌入式系统编程中，根据 CPU 和编译器的不同，各个编译器基本上会对C语言进行一些特定的扩充，以便满足嵌入式编程的需求，如上一节中 sbit 关键字，这也是我们初学者需要注意的地方。

iBoard 电子学堂教学代码力求通用，在严格遵照编译器语法的同时，也利用了C99 标准的一些优良的特性。基于STM32F103VC处理器的代码里某些部分包含了 C99 标准的语法，使用过程中需要在编译器配置里增加对 C99 语法的支持。