嵌入式C语言高手炼成之内存操作篇

嵌入式ARM 2019-11-08

数据指针

在嵌入式系统的编程中，常常要求在特定的内存单元读写内容，汇编有对应的MOV指令，而除C/C++以外的其它编程语言基本没有直接访问绝对地址的能力。在嵌入式系统的实际调试中，多借助C语言指针所具有的对绝对地址单元内容的读写能力。以指针直接操作内存多发生在如下几种情况：

(1) 某I/O芯片被定位在CPU的存储空间而非I/O空间，而且寄存器对应于某特定地址；

(2) 两个CPU之间以双端口RAM通信，CPU需要在双端口RAM的特定单元（称为mail box）书写内容以在对方CPU产生中断；

(3) 读取在ROM或FLASH的特定单元所烧录的汉字和英文字模。

譬如：

```
1  unsigned char *p = (unsigned char *)0xF000FF00;
```

```
2  *p="11";
```

以上程序的意义为在绝对地址0xF0000+0xFF00(80186使用16位段地址和16位偏移地址)写入11。

在使用绝对地址指针时，要注意指针自增自减操作的结果取决于指针指向的数据类别。上例中p++后的结果是p= 0xF000FF01，若p指向int，即：

```
1  int *p = (int *)0xF000FF00;
```

p++(或++p)的结果等同于：p = p+sizeof(int)，而p-(或-p)的结果是p = p-sizeof(int)。

记住：CPU以字节为单位编址，而C语言指针以指向的数据类型长度作自增和自减。理解这一点对于以指针直接操作内存是相当重要的。

函数指针

首先要理解以下三个问题：

（1）C语言中函数名直接对应于函数生成的指令代码在内存中的地址，因此函数名可以直接赋给指向函数的指针；

（2）调用函数实际上等同于"调转指令＋参数传递处理＋回归位置入栈"，本质上最核心的操作是将函数生成的目标代码的首地址赋给CPU的PC寄存器；

（3）因为函数调用的本质是跳转到某一个地址单元的code去执行，所以可以"调用"一

个根本就不存在的函数实体，晕？请往下看：

请拿出你可以获得的任何一本大学《微型计算机原理》教材，书中讲到，186 CPU启动后跳转至绝对地址0xFFFF0（对应C语言指针是0xF000FFF0，0xF000为段地址，0xFFF0为段内偏移）执行，请看下面的代码：

```
typedef void (*lpFunction) ( ); /* 定义一个无参数、无返回类型的 */
/* 函数指针类型 */
lpFunction lpReset = (lpFunction)0xF000FFF0; /* 定义一个函数指针，指向*/
/* CPU启动后所执行第一条指令的位置 */
lpReset(); /* 调用函数 */
```

在以上的程序中，我们根本没有看到任何一个函数实体，但是我们却执行了这样的函数调用：lpReset()，它实际上起到了"软重启"的作用，跳转到CPU启动后第一条要执行的指令的位置。

记住：函数无它，唯指令集合耳；你可以调用一个没有函数体的函数，本质上只是换一个地址开始执行指令！

数组**vs.**动态申请

在嵌入式系统中动态内存申请存在比一般系统编程时更严格的要求，这是因为嵌入式系统的内存空间往往是十分有限的，不经意的内存泄露会很快导致系统的崩溃。

所以一定要保证你的malloc和free成对出现，如果你写出这样的一段程序：

```
char * function(void)
{
```

```
  char *p;
  p = (char *)malloc(…);
  if(p==NULL)
    …;
    … /* 一系列针对p的操作 */
  return p;
}
在某处调用function()，用完function中动态申请的内存后将其free，如下：
char *q = function();
…
free(q);
```

上述代码明显是不合理的，因为违反了malloc和free成对出现的原则，即"谁申请，就由谁释放"原则。不满足这个原则，会导致代码的耦合度增大，因为用户在调用function函数时需要知道其内部细节！

正确的做法是在调用处申请内存，并传入function函数，如下：

```
char *p="malloc"(…);
if(p==NULL)
…;
function(p);
…
free(p);
p="NULL";
而函数function则接收参数p，如下：
void function(char *p)
{
  … /* 一系列针对p的操作 */
}
```

基本上，动态申请内存方式可以用较大的数组替换。对于编程新手，笔者推荐你尽量采用数组！嵌入式系统可以以博大的胸襟接收瑕疵，而无法"海纳"错误。毕竟，以最笨的方式苦练神功的郭靖胜过机智聪明的杨康。

给出原则：

（**1**）尽可能的选用数组，数组不能越界访问（真理越过一步就是谬误，数组越过界限就光荣地成全了一个混乱的嵌入式系统）；

（**2**）如果使用动态申请，则申请后一定要判断是否申请成功了，并且malloc和free应成对出现！

const在C++语言中则包含了更丰富的含义，而在C语言中仅意味着："只能读的普通变量"，可以称其为"不能改变的变量"（这个说法似乎很拗口，但却最准确的表达了C语言中const的本质），在编译阶段需要的常数仍然只能以#define宏定义！故在C语言中如下程序是非法的：

关键字**const**

const意味着"只读"。区别如下代码的功能非常重要，也是老生长叹，如果你还不知道它们的区别，而且已经在程序界摸爬滚打多年，那只能说这是一个悲哀：

```
const int a;
int const a;
const int *a;
int * const a;
int const * a const;
const int SIZE = 10;
char a[SIZE]; /* 非法：编译阶段不能用到变量 */
```

关键字**volatile**

volatile变量可能用于如下几种情况：

(1) 并行设备的硬件寄存器（如：状态寄存器，例中的代码属于此类）；

(2) 一个中断服务子程序中会访问到的非自动变量(也就是全局变量)；

(3) 多线程应用中被几个任务共享的变量。

-END-

推荐阅读

【01】深度：震惊世间的惊人代码（附完整代码）

【02】编译器如何将高级语言转化成汇编语言的?

【03】C语言在嵌入式系统编程时的注意事项

【04】由C语言编写的C编译器是怎样来的?

【05】还没搞懂C语言指针？最详细的干货讲解

【06】C语言结构体（struct）最全的讲解

【07】为什么在C语言中，goto这么不受待见?

@纳税人：以电子发票报销入账归档，要注意这些事项

国家税务总局

“女儿好还是儿媳好？”妈妈和婆婆的回答，截然不同

清欢读书会