**宏定义中的#和##**

关于记号粘贴操作符(token paste operator): ##  
  
  
  
  
1. 简单的说，“##”是一种分隔连接方式，它的作用是先分隔，然后进行强制连接。  
  
  
  
    其中，分隔的作用类似于空格。我们知道在普通的宏定义中，预处理器一般把空格  
  
    解释成分段标志，对于每一段和前面比较，相同的就被替换。但是这样做的结果是，  
  
    被替换段之间存在一些空格。如果我们不希望出现这些空格，就可以通过添加一些  
  
    ##来替代空格。  
  
  
  
    另外一些分隔标志是，包括操作符，比如 +, -, \*, /, [,], ...，所以尽管下面的  
  
    宏定义没有空格，但是依然表达有意义的定义： define add(a, b)   a+b  
  
  
  
    而其强制连接的作用是，去掉和前面的字符串之间的空格，而把两者连接起来。  
  
  
  
  
  
2. 举列 -- 试比较下述几个宏定义的区别  
  
  
  
    #define A1(name, type)   type name\_##type##\_type 或  
  
    #define A2(name, type)   type name##\_##type##\_type  
  
  
  
    A1(a1, int);   /\* 等价于: int name\_int\_type; \*/  
  
    A2(a1, int);   /\* 等价于: int a1\_int\_type;    \*/  
  
  
  
    解释：  
  
         1) 在第一个宏定义中，"name"和第一个"\_"之间，以及第2个"\_"和第二个  
  
    "type"之间没有被分隔，所以预处理器会把name\_##type##\_type解释成3段：  
  
    “name\_”、“type”、以及“\_type”，这中间只有“type”是在宏前面出现过  
  
     的，所以它可以被宏替换。  
  
  
  
         2) 而在第二个宏定义中，“name”和第一个“\_”之间也被分隔了，所以  
  
    预处理器会把name##\_##type##\_type解释成4段：“name”、“\_”、“type”  
  
    以及“\_type”，这其间，就有两个可以被宏替换了。  
  
  
  
         3) A1和A2的定义也可以如下：  
  
            #define A1(name, type)   type name\_   ##type ##\_type    
  
                                       <##前面随意加上一些空格>  
  
            #define A2(name, type)   type name ##\_ ##type ##\_type  
  
  
  
     结果是## 会把前面的空格去掉完成强连接，得到和上面结果相同的宏定义  
  
  
  
3. 其他相关 -- 单独的一个 #  
  
  
  
    至于单独一个#，则表示 对这个变量替换后，再加双引号引起来。比如  
  
  
  
       #define   \_\_stringify\_1(x)    #x  
  
那么  
  
       \_\_stringify\_1(linux)    <==>   "linux"  
  
  
  
所以，对于MODULE\_DEVICE\_TABLE  
  
  
  
      1) #define MODULE\_DEVICE\_TABLE(type,name)                          
  
              MODULE\_GENERIC\_TABLE(type##\_device,name)  
  
      2) #define MODULE\_GENERIC\_TABLE(gtype,name)                        
  
              extern const struct gtype##\_id \_\_mod\_##gtype##\_table       
  
              \_\_attribute\_\_ ((unused, alias(\_\_stringify(name))))  
  
  
  
得到    
  
       MODULE\_DEVICE\_TABLE(usb, products)    
  
                              /\*notes: struct usb\_device\_id products; \*/  
  
<==> MODULE\_GENERIC\_TABLE(usb\_device,products)  
  
<==> extern const struct usb\_device\_id \_\_mod\_usb\_device\_table       
  
              \_\_attribute\_\_ ((unused, alias("products")))     
  
  
  
注意到alias attribute需要一个双引号，所以在这里使用了\_\_stringify(name)来  
  
给name加上双引号。另外，还注意到一个外部变量"\_\_mod\_usb\_device\_table"被alias  
  
到了本驱动专用的由用户自定义的变量products<usb\_device\_id类型>。这个外部变量  
  
是如何使用的，更多的信息请参看《probe()过程分析》。