## 跟涛哥一起学嵌入式 11：一个非常有意思的宏

| 文档说明 | 作者 | 日期 |
| --- | --- | --- |
| 来自微信公众号：宅学部落(armLinuxfun) | wit | 2018.11.16 |
| 嵌入式视频教程淘宝店：https://wanglitao.taobao.com/ | | |
| 联系微信：brotau(宅学部落) | | |

QQ群 (宅学部落)有位学员问了一个很奇怪的宏，觉得很有意思，特拿来分享，它的定义如下:

```
#include<stdio.h>

#define AA(){ \
    unsigned long state;\
    state = 1;\
    printf("%d\r\n", state);
#define BB() state = 0;\
    printf("%d\r\n",state);\
}

int main(void)
{
//  AA();
//  BB();
    AA();BB();
    return 0;
}
```



我们知道，宏定义其实就是为了方便，给一串代码字符串定义一个别名。有时候字符串过于复杂，我们可以分多行书写，然后使用逻辑连接符"\"连接起来，表示一个完整的字符串。但是分析上面的宏定义，你会发现它分别定义了2个宏，但是呢，又使用了一对大括号括起来，很有欺骗性：看起来很像语句表达式，但是呢，有没有小括号括起来，是不是很奇怪？

调用的时候，使用方法更是奇怪，如果我们单独使用 AA() 或 BB() 调用，你会发现编译根本通不过，这是为什么呢？我们可以使用命令对上面的程序作预处理展开：

```
$ gcc -E main.c
```

预处理器对宏定义展开后，你会发现，会报语法错误，但是我们通过AA();BB();这种调用方式呢，就可以避免语法错误，可以顺利编译通过并运行。

这两个有意思的宏，要成对出现，才能避免程序编译错误，如果你只使用了其中一个，程序就编译通不过，彻底歇菜了。这是谁写的代码啊？为什么要这么写？其实正是这对宏的有意思之处，我们可以稍作封装：

```c
#include<stdio.h>

#define SYSTEM_LOCK(){ \
    unsigned long state;\
    state = 1;\
    printf("%d\r\n", state);
#define SYSTEM_UNLOCK() state = 0;\
    printf("%d\r\n",state);\
}

int main(void)
{
    SYSTEM_LOCK();
    //critical section operation
    SYSTEM_UNLOCK();
    return 0;
}
```

通过上面的修改，你会发现这对宏变成了一对加锁加锁功能，可以实现原子操作的临界区功能。而且更巧妙的是这对宏利用了C语法编译检查，强制程序员成对使用，否则就会报语法编译错误。因此，这种强制成对使用，也就避免了加锁解锁不成对出现时引起的死锁问题。

C语言博大精深，任何一段代码仔细分析，仔细推敲，都有很多编程技巧和精华在里面，感觉该学员提出的一个好问题，也欢迎更多的学员遇到问题踊跃提问、勤于思考，大家一起学习、努力、进步。