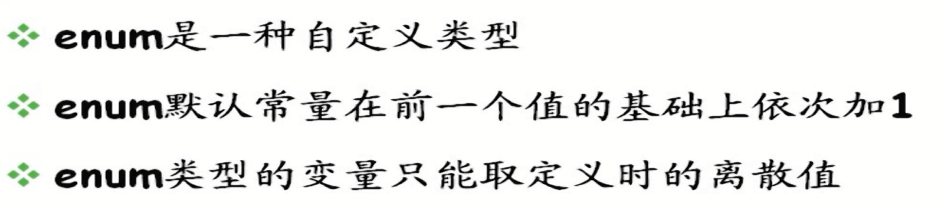
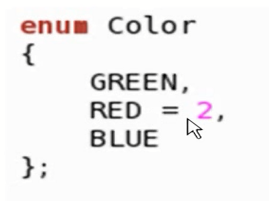


enum 自定义的类型 ---- 地位和struct和union地位一样

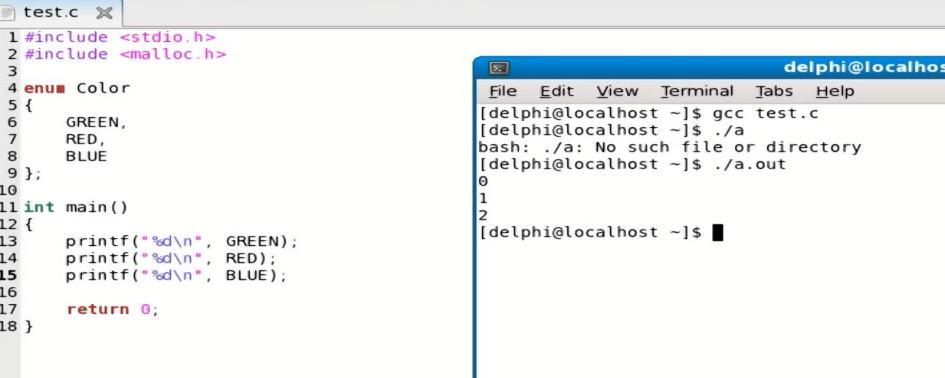
enum 是用来定义常量 是真真正正 是实实在在的常量 不是const那样的

enum只能定义离散的值 不能定义枚举类型为浮点数



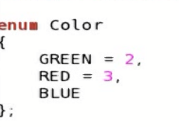
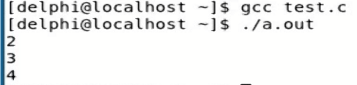


GREEN和BLUE 没有值 默认的第一个值就是0 这样GREEN没有任何值 默认取值为0 但是 BLUE会在前一个上面自动加1



任意枚举类型 不给值 就会给0

可以显式复制

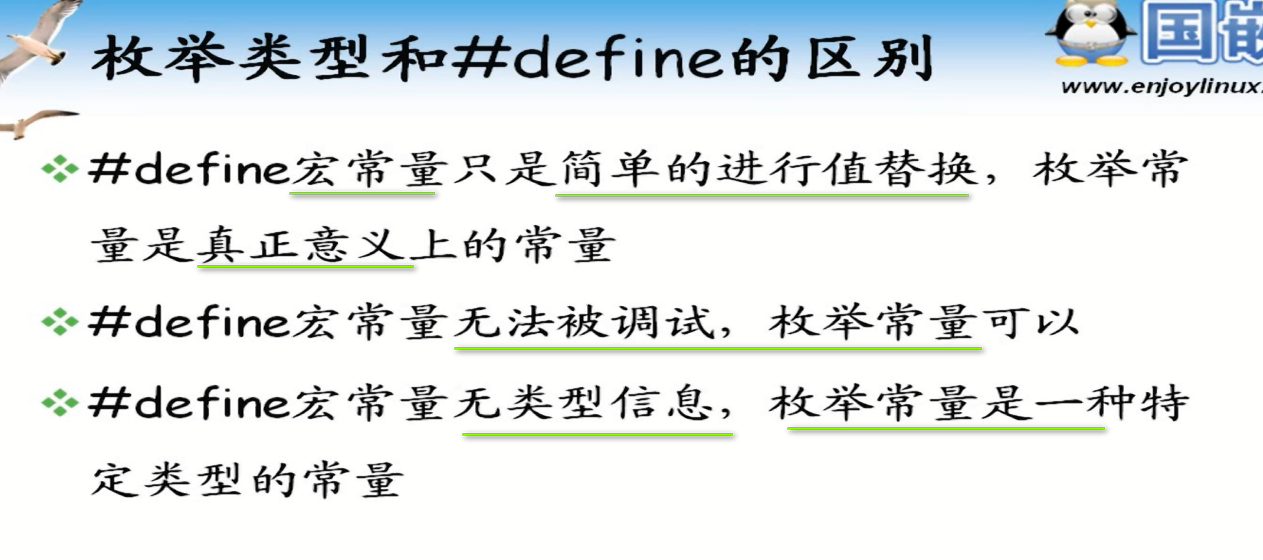
 

枚举常量 是真正意义的常量 ---- 可以试试



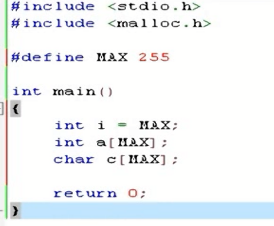


有人问 我这个枚举定义的常量 和define定义的宏 有什么区别？



枚举常量进入了符号表 那么gdb就可以查找到原来常量定义的名字

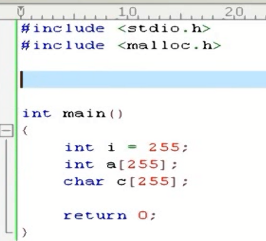
模拟define宏常量

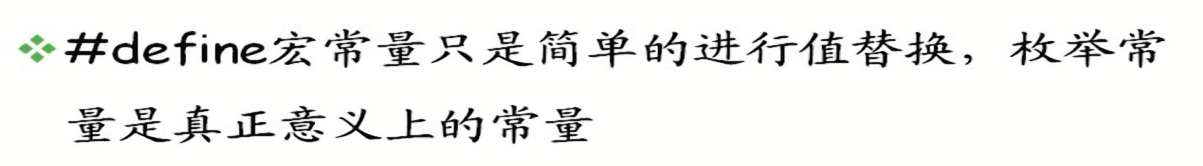


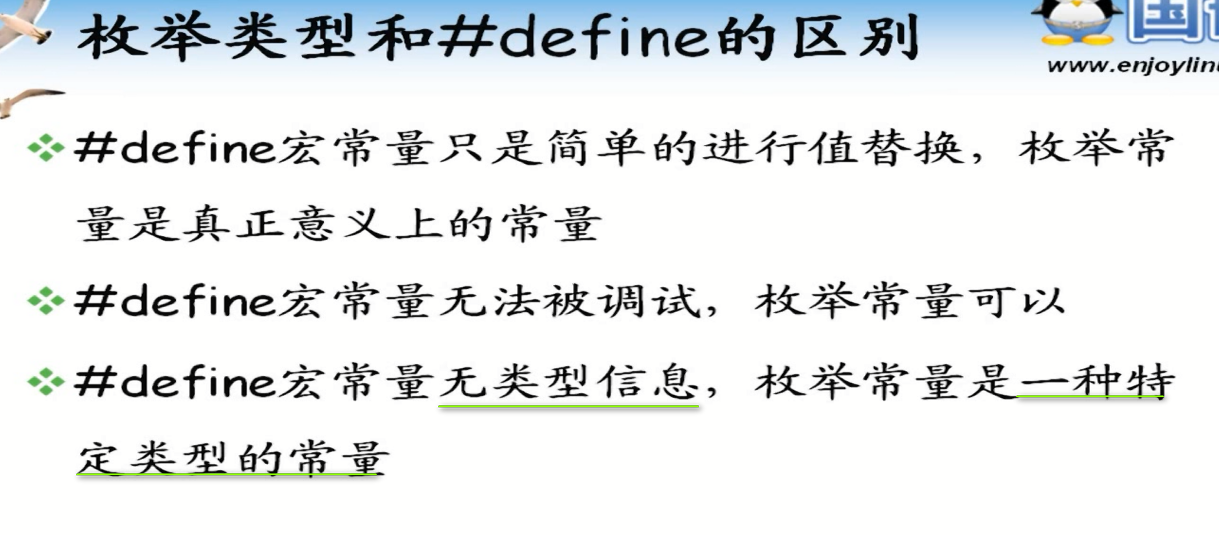
编译之前预处理器已经把MAX这个宏替换成255

然后进行编译

最终替换完 再把#define MAX 255这句给删除

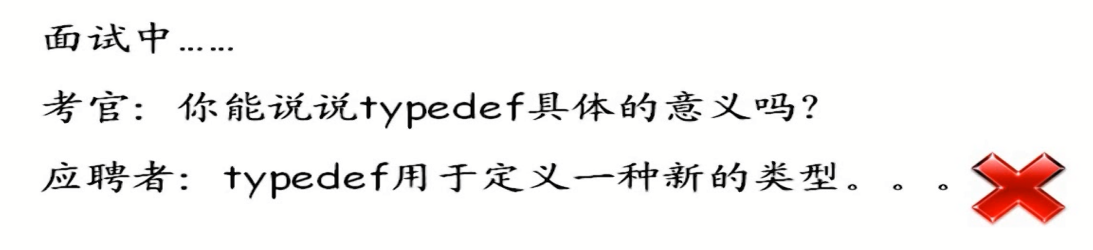
 真正编译的时候 MAX就没有了

所以  所以MAX不进符号表 GDB无法进行调试



**建议真正的常量使用enum进行定义**

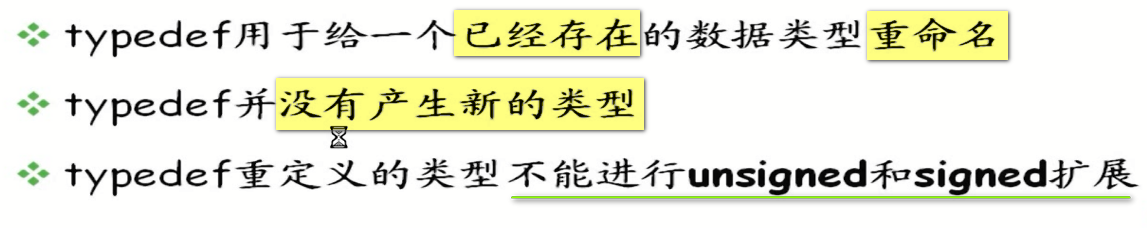
Typedef



考官一听 就知道不是真真正正了解C语言

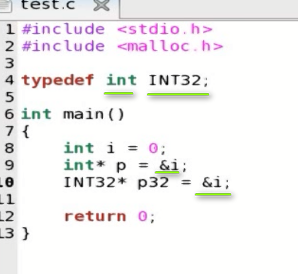
**typedef让人迷惑 ----- C语言中 仅仅是给一个已经存在的数据类型进行重命名 而不是进行定义新类型的操作 ===== typedef就是type rename的意思**

因此typedef没有产生新的类型 ------ 同样的数据类型有了一个新的名字



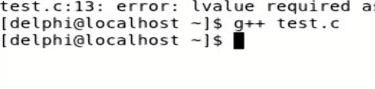
typedef不能用unsigned和signed扩展

通过指针验证typedef的数据类型是一个 没有新的

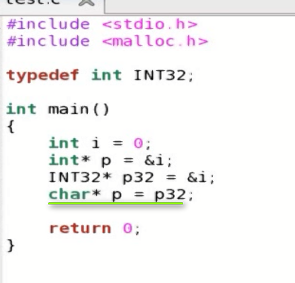


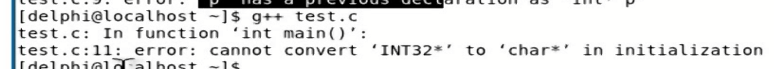
用强类型的编译器进行编译 不用gcc 用g++

G++是C++编译器 对类型更加严格



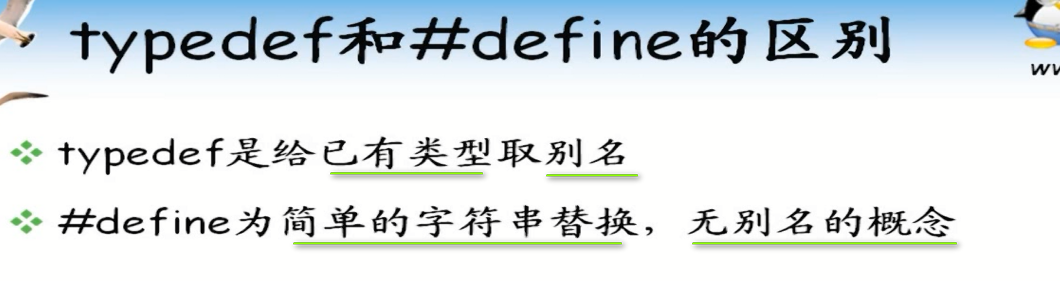
编译通过

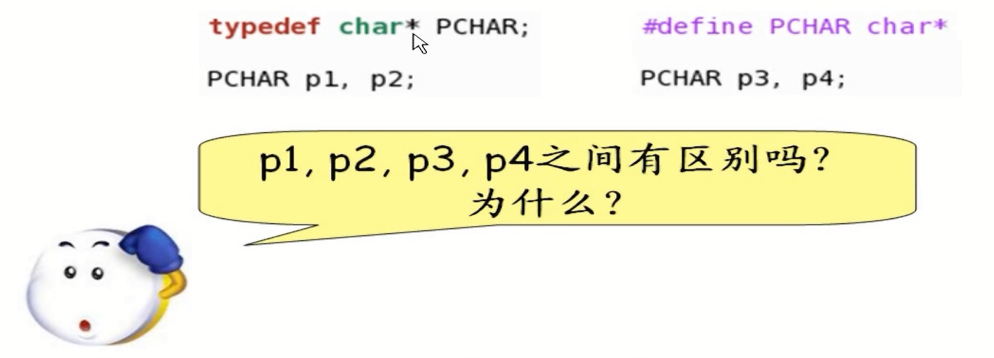




只有一种可能 也就是int 和INT32是同一种类型 仅是不同的别名

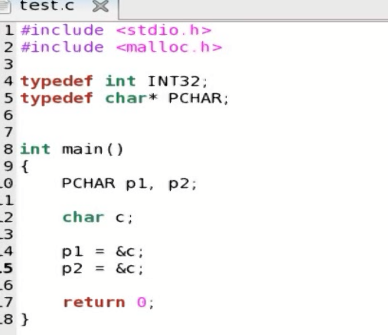
typedef是重命名的操作 不是重定义的操作



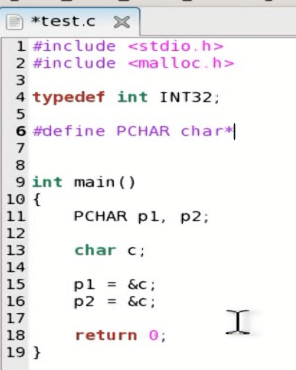


P1 p2是指向字符串的指针

后面define的PCHAR就会替换成char\* ---- char\* p3, p4; ---- 注意 p3是指向字符的指针 但是p4不是！

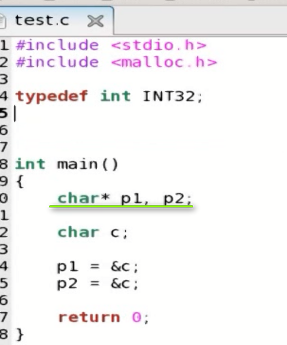
 编译通过

然后 把这个注释掉 使用宏定义

不能把char\*转换成char

实际上替换的时候 ---- 按照编译器的方法

 p2前面没有指针 所以 p2是一个char

但是typedef里面都是PCHAR ---- 所以 PCHAR p1, p2 ---- 说明p1和p2都是PCHAR类型 ---- 不是类似于define的简单替换