# 链接属性（external、internal、none）

当多个源文件链接在一起，我们如何处理相同名字的标识符？假设多个源文件均含有变量a，那么它的值到底采用哪个源文件定义的值呢？这就涉及到标识符的链接属性。

**对象**

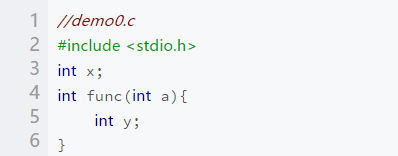
链接属性针对标识符而言，对象通常是变量、函数名、形参。

**分类**

链接属性（linkage）分为三种——外部（external）、内部（internal）、无（none）

**外部（external）**

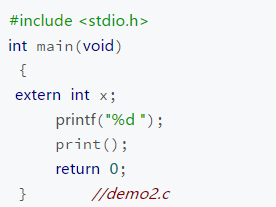
1、判别依据



在demo0.c中，x、func具有external的链接属性，简而言之，并非声明于代码块内的变量，缺省情况下具有external链接属性，（函数名也不包含于任何代码块）。

2、extern关键字



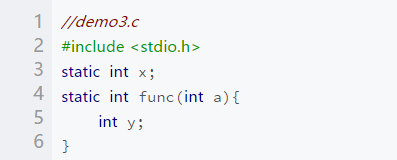


在文件A中具有external链接属性的变量（或函数名）具有全局可见性，对于变量，在声明前加extern表示引用文件A中的对应变量，函数则可以直接使用。

例如demo1.c中x和print具有external链接属性，在demo2.c中，extern int x表示这里的x就是demo1.c中的x，这里的x链接属性就是external，print就是demo1.c中的print 。

**内部（internal）**

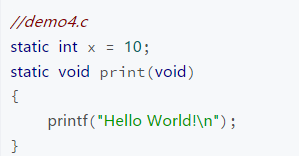
1、判断依据

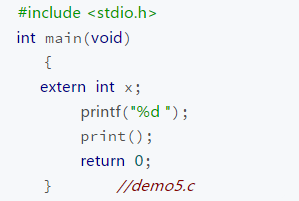


缺省情况下，除了上述说明的external属性以外，其余均为none。因此internal属性出现在static修饰之后。

demo3.c中,如果删除static，则和demo0.c相同，即x和func在原先属于external，在用static关键字修饰之后则有了internal属性，即只能够在demo3.c一个文件中访问，其他文件不能访问。

2、static关键字





上述两个程序不能成功链接。

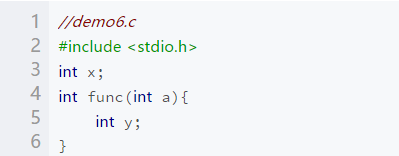
当具有external属性的变量（或函数名）用static关键字修饰之后，就具有了internal链接属性，其他文件不能访问它们。

因此，demo5.c中extern int x; 和 print(); 是错误的，因为x和print不再是external的链接属性了。

不要用static关键字修饰none属性的标识符，对于链接属性，static只能修改external属性的标识符，若修饰none属性的，改变的是存储类型而非链接属性。

**无（none）**

判断依据



在demo6.c中，a和y具有none的链接属性，简而言之， 函数形式参数和代码块内声明的变量在缺省情况下具有none链接属性 。

none链接属性标识符仅允许作用域内访问。

**Tips**

1. 链接属性的改变不会改变作用域（scope）。
2. 链接不是 #include，不能直接 #include”demox.c” 来链接不同的源文件，否则会得出不同结论。链接方式多样，可以用gcc命令，gcc -o filename file1.c file2.c 。上述程序范例很相似，可以通过寻找不同处来理解链接属性。
3. 链接属性不仅意味着能不能访问变量，也意味着能不能在文件中定义同名变量或函数名。