 面向对象的另一个特性是运行时识别。当然，如果你的系统设计的足够完美的话，也用不到什么运行时识别，但是有时使用一下运行是识别，能够是设计简化不少。言归正传，本文就是在前文的基础上再研究一下，如何使用 C 来进行运行时识别。 我们知道，C 里的虚函数实际上不是与对象绑定，而是与类绑定的。也就是说，一个类一个虚函数表，而不是一个对象一个虚函数表。所以，如果一个类的虚函数较多时，像前文那样定义接口： struct IStream { void (*write)(IStream* pStream, char* pstr). void(*read)(IStream* pStream, char buf, ssize_t size). void (*flush)(IStream* pStream). … }. 就会很浪费，因为每个对象都有 sizeof(struct IStream) 这么大。但是我们知道，虚函数表是与类绑定的，而不是与对象绑定的，那么我们可以改进这个设计。 struct IStreamClass { void (*write)(IStream* pStream, char* pstr). void(*read)(IStream* pStream, char buf, ssize_t size). void (*flush)(IStream* pStream). … }. 基类的定义改为： struct IStream{ { struct IStreamClass* vtbl. } 如果需要实现一个 FileStream 的话，可以定义一个具体类。 struct FileStream { struct IStream base. FILE* f. } FileStreamClass 可以使用一个全局变量来定义： extern struct IStreamClass class_FileStream. 因为这个虚函数表是与类绑定的，所以有了它，我们还可以增加运行时识别的功能！ int isFileStream(struct IStream* pStream){ return pStream- 