

在C语言里进行面向对象设计:模拟运行时识别计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/645/2021_2022__E5_9C_A8C

[_E8_AF_AD_E8_A8_80_c97_645853.htm](https://www.100test.com/kao_ti2020/645/2021_2022__E5_9C_A8C_E8_AF_AD_E8_A8_80_c97_645853.htm) 面向对象的另一个特性是运行时识别。当然，如果你的系统设计得足够完美的话，也用不到什么运行时识别，但是有时使用一下运行时识别，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 IStreamClass { 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->vtbl == class_FileStream.vtbl; }
```

```
return pStream->vtbl == class_FileStream.vtbl; }
```

```
return pStream->vtbl == class_FileStream.vtbl; }
```

```
return pStream->vtbl == class_FileStream.vtbl; }
```

```
return pStream->vtbl == class_FileStream.vtbl; }
```

```
return pStream->vtbl == class_FileStream.vtbl; }
```

载。详细请访问 www.100test.com