会得到这张表。

.line：原始C源程序中的行号和.text节中机器指令之间的映射。只有以-g选项调用编译器驱动程序时，才会得到这张表。

.strtab：一个字符串表，其内容包括.symtab和.debug节中的符号表，以及节头部中的节名字。字符串表就是以null结尾的字符串的序列。

旁注　为什么未初始化的数据称为.bss

用术语.bss来表示未初始化的数据是很普遍的。它起始于IBM 704汇编语言(大约在1957年)中“块存储开始(Block Storage Start)”指令的首字母缩写，并沿用至今。一种记住.data和.bss节之间区别的简单方法是把“bss”看成是“更好地节省空间(Better Save Space)”的缩写。

7.5　符号和符号表

每个可重定位目标模块m都有一个符号表，它包含m定义和引用的符号的信息。在链接器的上下文中，有三种不同的符号：

- 由模块m定义并能被其他模块引用的全局符号。全局链接器符号对应于非静态的C函数和全局变量。
- 由其他模块定义并被模块m引用的全局符号。这些符号称为外部符号，对应于在其他模块中定义的非静态C函数和全局变量。
- 只被模块m定义和引用的局部符号。它们对应于带static属性的C函数和全局变量。这些符号在模块m中任何位置都可见，但是不能被其他模块引用。

认识到本地链接器符号和本地程序变量不同是很重要的。.symtab中的符号表不包含对应于本地非静态程序变量的任何符号。这些符号在运行时在栈中被管理，链接器对此类符号不感兴趣。

有趣的是，定义为带有C static属性的本地过程变量是不在栈中管理的。相反，编译器在.data或.bss中为每个定义分配空间，并在符号表中创建一个有唯一名字的本地链接器符号。比如，假设在同一模块中的两个函数各自定义了一个静态局部变量x：

```
1   int f()
2   {
3       static int x = 0;
4       return x;
5   }
6
7   int g()
8   {
9       static int x = 1;
10      return x;
11  }
```

在这种情况中，编译器向汇编器输出两个不同名字的局部链接器符号。比如，它可以用x.1表示函数f中的定义，而用x.2表示函数g中的定义。

给C语言初学者　利用static属性隐藏变量和函数名字

C程序员使用static属性隐藏模块内部的变量和函数声明，就像你在Java和C++