回车符结束，其他系统则用换行符结束。有些系统还同时采用回车符和换行符（如MS-DOS）。文件中各行的长度不一定相同。

ASCII文件的最大优势是可以显示和打印，还可以用任何文本编辑器进行编辑。再者，如果很多程序都以ASCII文件作为输入和输出，就很容易把一个程序的输出作为另一个程序的输入，如shell管道一样。（用管道实现进程间通信并非更容易，但若以一种公认的标准（如ASCII码）来表示，则更易于理解一些。）

其他与ASCII文件不同的是二进制文件。打印出来的二进制文件是无法理解的、充满混乱字符的一张表。通常，二进制文件有一定的内部结构，使用该文件的程序才了解这种结构。

如图4-3a是一个简单的可执行二进制文件，它取自某个版本的UNIX。尽管这个文件只是一个字节序列，但只有文件的格式正确时，操作系统才会执行这个文件。这个文件有五个段：文件头、正文、数据、重定位位及符号表。文件头以所谓的魔数（magic number）开始，表明该文件是一个可执行的文件（防止非这种格式的文件偶然运行）。魔数后面是文件中各段的长度、执行的起始地址和一些标志位。程序本身的正文和数据在文件头后面。这些被装入内存，并使用重定位位重新定位。符号表则用于调试。

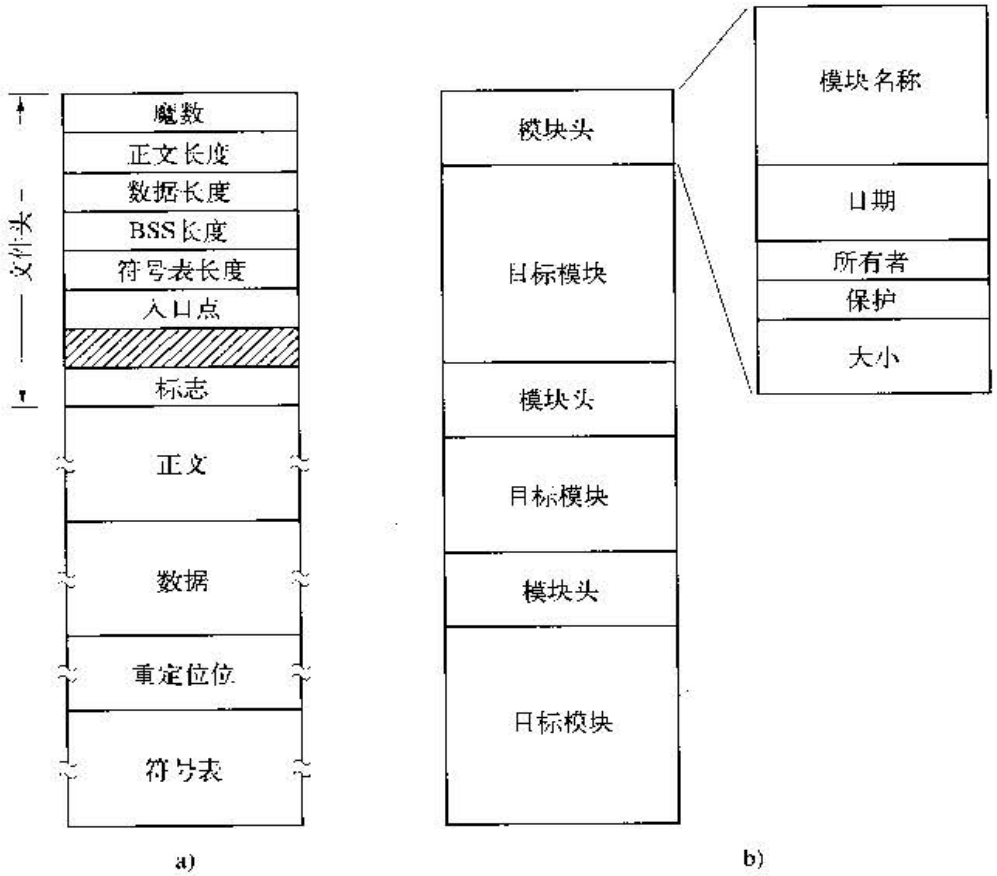

图4-3 a)一个可执行文件；b) 一个存档文件

二进制文件的第二个例子是UNIX的存档文件，它由已编译但没有连接的库过程（模块）集合而成。每个文件以模块头开始，其中记录了名称、创建日期、所有者、保护码和文件大小。该模块头与可执行文件一样，也都是二进制数字，打印输出它们毫无意义。

所有操作系统必须能够识别它们自己的可执行文件的文件类型，其中有些操作系统还可识别更多的信息。一种老式的TOPS-20操作系统（用于DECsystem20计算机）甚至可检查可执行文件的创建时间，然后，它可以找到相应的源文件，看它在二进制文件生成后是否被修改过。如果修改过，操作系统自动重新编译这个文件。在UNIX中，就是在shell中嵌入make程序。这时操作系统要求用户必须采用固定的文件扩展名，从而确定哪个源程序生成哪个二进制文件。

如果用户执行了系统设计者没有考虑到的某种操作，这种强制类型的文件有可能会引起麻烦。比如在一个系统中，程序输出文件的扩展名是.dat（数据文件），若用户写一个格式化程序，读入.c（C程序）