1 流的概念

“流是磁盘或其它外围设备中存储的数据的源点或终点。”。这是在《C程序设计语言》上的原话。

据我的理解，流主要是指一种逻辑上的概念，它提供或存储数据。产生数据的叫输入流，消耗数据的叫输出流。至于怎么产生，又怎么消耗，这是一种物理上的实现，根据每种设备有所不同，但C语言中对它们一视同仁，以一个“流”字来概括它们的特征。作为流的使用者来说，不需要关心太多的细节。流的实现保证了它具有它所声明的特性。C语言中对流除了分为I/O流之外，还分为文本流与二进制流。文本流的特点是流由文本行组成，每一行有0个或多个字符并以'\n'字符结束，即它是有一定意义的，以某种字符集的字符组成的一个序列。一个文本流，读入与写出时可能会对其内容作更改，因为它是有一定意义的，系统可以识别并在适当时候解释，比如在输出文本流中碰到'\b'时，系统的操作是将输入流中的前一个字符删除，在终端上显示就是在它前面输出的这个字符被删除了；二进制流则完成是由一些“生”的，未经处理的数据组成的，C语言将它们看成由0与1组成的序列来读与写，所以它们的特性是同一系统中把同一二进制流读入与写出，其内容没有任何变化。

2 文件的概念

每个操作系统为了存储的管理，都会提供文件系统，文件系统由接口与实现组成。接口定义了文件，目录的属性与操作。实现与具体的物理设备有关。用户不必要关注具体是怎么实现的，只要了解了接口就可以操作文件了。

文件也是一个逻辑上的概念，它定义了一个逻辑上的存储对象是什么样子的，如它的结构，属性等。每个操作系统对文件的定义都有差别，有的简单，有的复杂，这里讨论Unix的文件定义。

Unix将文件看成是字节序列。从这里可以看出，Unix对文件的定义是相当原始与简单的，这个定义没有规定文件的任何结构（除了文件必须是一个字节的序列之外）。这个定义给了用户极大的灵活性。操作系统在读一个文件时，它也不知道文件本身是什么意思，它将文件读出来交给应用程序，文件的结构由应用程序自己维护，也就是说，只有应用程序才知道这个文件是什么意思。不过，Unix系统中的很多程序都是将文件看作是文本的，所以很多应用程序通过文件很容易共享数据。

3 流与文件的关系

这两个都是逻辑上的概念，都是对I/O设备的一种高级抽象。它们往往指那些物理上的设备。所以它们经常互换着使用。但它们还是有区别的，主要的区别是它们各自的侧重点不同。流是动态的，更偏重于操作的过程，将数据看成是一种正在朝向个方向运动的对象，输入或输出；文件是静态的，更偏重于操作的对象本身，将数据看成是操作的对象或结果，文件在其它系统里本来就只有存储对象的意思，Unix扩展了这个概念。不过，我们一般都不加区别地使用它们，尤其是在Unix系统与C语言环境中。

4 文件的结束标记

学过C的都知道，文件的结束标记是EOF(-1)。实际上它是文本文件的结束标记，也即文本流的结束标记。

前面说过，Unix甚至对简单如文本的结构也不能识别，所以，它不知道一个文本文件的结束标记是EOF。Unix是通过记录文件的长度来知道一个文件是否结束的。我们要知道这一点。