当前大小字段指出了当前的文件大小。在一些老式大型机操作系统中创建文件时，要给出文件的最大长度，以便操作系统事先按最大长度留出存储空间。工作站和和个人计算机中的操作系统则聪明多了，不需要这一点提示。

4.1.6 文件操作

使用文件的目的是存储信息并方便以后的检索。对于存储和检索，不同系统提供了不同的操作。以下是与文件有关的最常用的一些系统调用：

1) create。创建不包含任何数据的文件。该调用的目的是表示文件即将建立，并设置文件的一些属性。

2) delete。当不再需要某个文件时，必须删除该文件以释放磁盘空间。任何文件系统总有一个系统调用用来删除文件。

3) open。在使用文件之前，必须先打开文件。open调用的目的是：把文件属性和磁盘地址表装入内存，便于后续调用的快速存取。

4) close。存取结束后，不再需要文件属性和磁盘地址，这时应该关闭文件以释放内部表空间。很多系统限制进程打开文件的个数，以鼓励用户关闭不再使用的文件。磁盘以块为单位写入，关闭文件时，写入该文件的最后一块，即使这个块还没有满。

5) read。在文件中读取数据。一般地，读出数据来自文件的当前位置。调用者必须指明需要读取多少数据，并且提供存放这些数据的缓冲区。

6) write。向文件写数据，写操作一般也是从文件当前位置开始。如果当前位置是文件末尾，文件长度增加。如果当前位置在文件中间，则现有数据被覆盖，并且永远丢失。

7) append。此调用是write的限制形式，它只能在文件末尾添加数据。若系统只提供最小系统调用集合，则通常没有append。很多系统对同一操作提供了多种实现方法，这些系统中有时有append调用。

8) seek。对于随机存取文件，要指定从何处开始取数据，通常的方法是用seek系统调用把当前位置指针指向文件中特定位置。seek调用结束后，就可以从该位置开始读写数据了。

9) get attributes。进程运行常需要读取文件属性。例如，UNIX中make程序通常用于管理由多个源文件组成的软件开发项目。在调用make时，检查全部源文件和目标文件的修改时间，实现最小编译，使得全部文件都为最新版本。为达到此目的，需要查找文件的某一些属性，特别是修改时间。

10) set attributes。某些属性是可由用户设置的，在文件创建之后，用户还可以通过系统调用set attributes来修改它们。保护模式信息是一个显著的例子，大多数标志也属于此类属性。

11) rename。用户常常要改变已有文件的名字，rename系统调用用于这一目的。严格地说，设置这个系统调用不是十分必要的，因为可以先把文件复制到一个新文件名的文件中，然后删除原来的文件。

4.1.7 使用文件系统调用的一个示例程序

本节会考察一个简单的UNIX程序，它把文件从源文件处复制到目标文件处。程序清单如图4-5所示。该程序的功能很简单，甚至没有考虑出错报告处理，但它给出了有关文件的系统调用是怎样工作的一般思路。

例如，通过下面的命令行可以调用程序copyfile：

```
copyfile abc xyz
```

把文件abc复制到xyz。如果xyz已经存在，abc会覆盖它。否则，就创建它。程序调用必须提供两个参数，它们都是合法的文件名。第一个是源文件，第二个是输出文件。

在程序的开头是四个#include语句，它们把大量的定义和函数原型包含在这个程序。为了使程序遵守相应的国际标准，这些是需要的，无须作进一步的讨论。接下来一行是main函数的原型，这是ANSI C所必需的，但对我们的目的而言，它也不是重点。

接下来的第一个#define语句是一个宏定义，它把BUF_SIZE字符串定义为一个宏，其数值为4096。程序会读写若干个有4096个字节的块。类似地，给常数一个名称而且用这一名称代替常数是一种良好的编程习惯。这样的习惯不仅使程序易读，而且使程序易于维护。第二个#define 语句决定谁可以访问输出文件。

主程序名为main，它有两个参数：argc和argv。当调用这个程序时，操作系统提供这两个参数。第一个参数表示在调用该程序的命令行中包含多少个字符串，包括该程序名。它应该是3。第二个参数是指向程序参数的指针数组。在上面的示例程序中，这一数组的元素应该包含指向下列值的指针：