```
fd = creat("abc", mode);
```

创建了一个名为abc的文件，并根据mode设置文件的保护位。这些保护位决定了用户访问文件的权限及方式。在下文将会具体讨论。

creat系统调用不仅创建了一个新文件，还以写的方式打开了这个文件。为了使以后的系统调用能够访问这个文件，creat成功时返回一个非负整数，这个非负整数叫做文件描述符，也就是例子中的fd。如果creat作用在一个已经存在的文件上，那么该文件的文件长度会被截短为0，它的内容会被丢弃。通过设置合适的参数，open系统调用也能创建文件。

现在我们继续讨论图10-27列出的主要的文件系统调用。为了读或写一个已经存在的文件，必须使用open系统调用打开这个文件。它的参数是要打开文件的文件名以及打开方式：只读、只写或两者。此外，也可以指定不同的选项。和creat一样，open返回一个文件描述符，可用来进行读写。然后可以使用close系统调用来关闭文件，它使得文件描述符可以被后来的creat或open使用。creat和open系统调用总是返回未被使用的最小数值的文件描述符。

当一个程序以标准方式运行时，文件描述符0、1、2已经分别用于标准输入、标准输出和标准错误。通过这种方式，一个过滤器，比如sort程序，可以从文件描述符0读取输入，输出到文件描述符1，而不需要关心这些文件是什么。这种机制能够有效是因为shell在程序启动之前就设置好了它们的值。

毫无疑问，最常使用的文件系统调用是read和write。它们每个都有三个参数：文件描述符（标明要读写的文件）、缓冲区地址（给出数据存放的位置或者读取数据的位置），长度（给出要传输的数据的字节数）。这些就是全部了。这种设计非常简单，一个典型的调用方法是：

```
n = read(fd, buffer, nbytes);
```

系统调用	描　述
fd = creat(name,mode)	创建新文件的一种方法
fd = open(file, how,…)	打开文件读、写或者读写
s = close(fd)	关闭一个已经打开的文件
n = read(fd, buffer, nbytes)	从文件中读取数据到一个缓冲区
n = write(fd, buffer, nbytes)	把数据从缓冲区写到文件
position = lseek(fd, offset, whence)	移动文件指针
s = stat(name, &buf)	获取一个文件的状态信息
s = fstat(fd, &buf)	获取一个文件的状态信息
s = pipe(&fd[0])	创建一个管道
s = fcntl(fd, cmd,…)	文件加锁及其他操作

图10-27　跟文件相关的一些系统调用。如果发生错误，那么返回值s是-1；fd是一个文件描述符，position是文件偏移。参数的含义是很清楚的

虽然几乎所有程序都是顺序读写文件的，但是一些程序需要能够从文件的任何位置随机地读写文件。每个文件都有一个指针指向文件当前的读写位置。当顺序地读写文件时，这个指针指向将要读写的字节。如果文件位置指针最初指向4096，在读取了1024个字节后，它会自动地指向第5120个字节。lseek系统调用可以改变位置指针的值，所以之后的read和write可以从文件的任何位置开始读写，甚至是超出文件的结尾。这个系统调用叫做lseek，是为了避免与seek冲突，其中后者以前在16位计算机上用于查找，现在已经不使用了。

lseek有三个参数：第一个是文件描述符，第二个是文件读写位置，第三个表明读写位置是相对于文件开头、当前位置还是文件尾。lseek的返回值是当读写位置改变后的绝对位置。有点讽刺的是，lseek是惟一一个从不会引起实际的磁盘寻道的文件系统调用，因为它所做的只是修改了内存中的一个值（文件读写位置）。

对于每个文件，Linux记录了它的文件类型（普通文件、目录、特殊文件）、大小、最后一次修改时间和其他信息。程序可以使用stat系统调用来查看这些信息，stat的第一个参数是文件名，第二个参数是