/usr/ast/mailbox表示根目录中有子目录usr，而usr中又有子目录ast，文件mailbox就在子目录ast下。绝对路径名一定从根目录开始，且是惟一的。在UNIX中，路径各部分之间用“/”分隔。在Windows中，分隔符是“\”。在MULTICS中是“>”。这样在这三个系统中同样的路径名按如下形式写成：

Windows	\usr\ast\mailbox
UNIX	/usr/ast/mailbox
MULTICS	>usr>ast>mailbox

不管采用哪种分隔符，如果路径名的第一个字符是分隔符，则这个路径就是绝对路径。

另一种指定文件名的方法是使用相对路径名（relative path name）。它常和工作目录（working directory）（也称作当前目录（current directory））一起使用。用户可以指定一个目录作为当前工作目录。这时，所有的不从根目录开始的路径名都是相对于工作目录的。例如，如果当前的工作目录是/usr/ast，则绝对路径名为/usr/ast/mailbox的文件可以直接用mailbox来引用。也就是说，如果工作目录是/usr/ast，则UNIX命令

```
cp /usr/ast/mailbox /usr/ast/mailbox.bak
```

和命令

```
cp mailbox mailbox.bak
```

具有相同的含义。相对路径往往更方便，而它实现的功能和绝对路径完全相同。

一些程序需要存取某个特定文件，而不论当前目录是什么。这时，应该采用绝对路径名。比如，一个检查拼写的程序要读文件/usr/lib/dictionary，因为它不可能事先知道当前目录，所以就采用完整的绝对路径名。不论当前的工作目录是什么，绝对路径名总能正常工作。

当然，若这个检查拼写的程序要从目录/usr/lib中读很多文件，可以用另一种方法，即执行一个系统调用把该程序的工作目录切换到/usr/lib，然后只需用dictionary作为open的第一个参数。通过显式地改变工作目录，可以知道该程序在目录树中的确切位置，进而可以采用相对路径名。

每个进程都有自己的工作目录，这样在进程改变工作目录并退出后，其他进程不会受到影响，文件系统中也不会有改变的痕迹。对进程而言，切换工作目录是安全的，所以只要需要，就可以改变当前工作目录。但是，如果改变了库过程的工作目录，并且工作完毕之后没有修改回去，则其他程序有可能无法正常运行，因为它们关于当前目录的假设已经失效。所以库过程很少改变工作目录，若非改不可，必定要在返回之前改回到原有的工作目录。

支持层次目录结构的大多数操作系统在每个目录中有两个特殊的目录项“.”和“..”，常读作“dot”和“dotdot”。dot指当前目录，dotdot指其父目录（在根目录中例外，根目录中它指向自己）。要了解怎样使用它们，请考虑图4-8中的UNIX目录树。一个进程的工作目录是/usr/ast，它可采用“..”沿树向上。例如，可用命令

```
cp ../lib/dictionary .
```

把文件usr/lib/dictionary复制到自己的目录下。第一个路径告诉系统上溯（到usr目录），然后向下到lib目录，找到dictionary文件。

第二个参数（.）指定当前目录。当cp命令用目录名（包括“.”）作为最后一个参数时，则把全部的文件复制到该目录中。当然，对于上述复制，键入

```
cp /usr/lib/dictionary .
```

是更常用的方法。用户这里采用“.”可以避免键入两次dictionary。无论如何，键入

```
cp /usr/lib/dictionary dictionary
```

也可正常工作，就像键入

```
cp /usr/lib/dictionary /usr/ast/dictionary
```

一样。所有这些命令都完成同样的工作。