这是有原因的。系统调用可能堵塞调用者，避免它继续执行。例如，如果试图读键盘，但是并没有任何键入，那么调用者就必须被阻塞。在这种情形下，操作系统会查看是否有其他可以运行的进程。稍后，当需要的输入出现时，进程会提醒系统注意，然后步骤9～步骤11会接着进行。

下面几节中，我们将考察一些常用的POSIX系统调用，或者用更专业的说法，考察进行这些系统调用的库过程。POSIX大约有100个过程调用，它们中最重要的过程调用列在图1-18中。为方便起见，它们被分成4类。我们用文字简要地叙述其作用。

从广义上看，由这些调用所提供的服务确定了多数操作系统应该具有的功能，而在个人计算机上，资源管理功能是较弱的（至少与多用户的大型机相比较是这样）。所包含的服务有创建与终止进程，创建、删除、读出和写入文件，目录管理以及完成输入输出。

**进程管理**

| 调 用 | 说 明 |
|---|---|
| pid = fork() | 创建与父进程相同的子进程 |
| pid = waitpid(pid, &statloc,options) | 等待一个子进程终止 |
| s = execve(name, argv, environp) | 替换一个进程的核心映像 |
| exit(status) | 中止进程执行并返回状态 |

**文件管理**

| 调 用 | 说 明 |
|---|---|
| fd = open(file, how, ...) | 打开一个文件供读、写或两者 |
| s = close(fd) | 关闭一个打开的文件 |
| n = read(fd, buffer, nbytes) | 把数据从一个文件读到缓冲区中 |
| n = write(fd, buffer, nbytes) | 把数据从缓冲区写到一个文件中 |
| position = lseek(fd, offset, whence) | 移动文件指针 |
| s = stat(name, &buf) | 取得文件的状态信息 |

**目录和文件系统管理**

| 调 用 | 说 明 |
|---|---|
| s = mkdir(name, mode) | 创建一个新目录 |
| s = rmdir(name) | 删去一个空目录 |
| s = link(name1, name2) | 创建一个新目录项name2，并指向name 1 |
| s = unlink(name) | 删去一个目录项 |
| s = mount(special, name, flag) | 安装一个文件系统 |
| s = umount(special) | 卸载一个文件系统 |

**杂 项**

| 调 用 | 说 明 |
|---|---|
| s = chdir(dirname) | 改变工作目录 |
| s = chmod(name, mode) | 修改一个文件的保护位 |
| s = kill(pid, signal) | 发送信号给一个进程 |
| seconds =time(&seconds) | 自1970年1月1日起的流逝时间 |

图1-18 一些重要的POSIX系统调用。若出错则返回代码s为−1。返回代码如下：pid是进程的id，fd是文件描述符，*n*是字节数，position是在文件中的偏移量，而seconds是流逝时间。参数在表中解释

有必要指出，将POSIX 过程映射到系统调用并不是一对一的。POSIX标准定义了构造系统所必须