# 文件编程

与UNIX一样，Linux环境中的文件具有特别重要的意义，因为它们的操作系统服务和设备提供了一个简单而统一的接口。在linux中一切皆文件。这就意味着，通常程序可以像使用文件那样使用磁盘文件、串行口、打印机和其他设备。

在输入输出操作中，直接使用底层系统调用的问题使它们的效率非常低。为什么呢？

系统调用会影响系统的性能。与函数调用相比，系统调用的开销要大些，因为在执行系统调用时，Linux必须从用户代码切换到内核代码运行，然后再返回用户代码。减少这种开销的一个好方法是，在程序中尽量减少系统调用的次数，并且让每次系统调用完成尽可能多的工作。

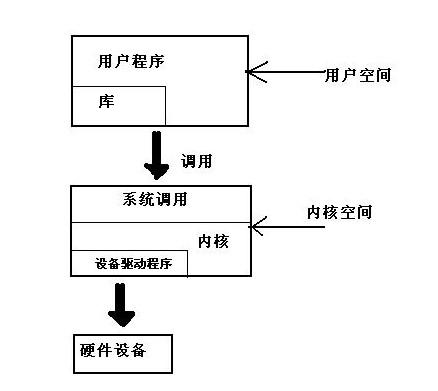
例如：每次读写大量的数据而不是每次仅读写一个字符。

  硬件会对底层系统调用一次所能读写的数据块做出一定的限制。例如：磁带机通常的写操作数据块的长度为10k，所以如果卸的数据量不是10k的整数倍，磁带机还是会以10k为单位卷绕磁带，这就在磁带上留下了空隙。

为了给设备和磁盘文件提供更高层的接口，与Unix一样，Linux发行版提供了一系列的标准函数库。它们是一些由函数构成的集合，你可以把它们包括在自己的程序中去处理一些与设备和文件有关的问题。提供输出缓冲功能的标准IO库就是一个这样的例子。你可以高效地写任意长度的数据块，库函数则在数据满足数据块长度要求时安排执行底层系统调用。这就极大的降低了系统调用的负面影响。

库函数会有一个与之相对应的标准头文件，例如：与标准IO库对应的头文件stdio.h。

图：显示了Linux系统中各种文件函数与用户、设备驱动程序、内核和硬件之间的关系。



每个运行中的程序被称为进程（process），它有一些与之相关联的文件描述符。这是一些小值整数，你可以通过它们访问打开的文件或设备。有多少文件描述符可用取决于系统的配置情况。当开始运行程序时，它一般会有三个已经打开的文件描述符。它们是：

0：标准输入

1：标准输出

2：标准出错

**文件描述符**

在Linux系统中，所有打开的文件都对应一个文件描述符。文件描述符的本质是一个非负整数 。当打开一个文件时，该整数由系统来分配 。文件描述符的范围是是0 - OPEN\_MAX 。早期的的UNIX 版本OPEN\_MAX =19，即允许每个进程同时打开20个文件，现在很多系统则将其增加至1024。

## Linux中文件编程可以使用两种方法:

#### Linux系统调用

#### C语言库函数

前者依赖于Linux系统，后者与操作系统是独立的，在任何操作系统下，使用C语言操作文件的方法都是一致的。

系统调用open ()  ,read(),write(),lseek();

库函数fopen(), fread(),fwrite(),flseek();

系统调用 – 创建

Int creat(const char \*filename,mode\_t mode)

filename:要创建的文件名（包含路径，缺省为当前路径）

mode:创建模式

常见创建模式：

S\_IRUSR 可读

S\_IWUSR 可写

S\_IXUSR 可执行

S\_IRWXU 可读、可写、可执行

除了可以使用上述宏以外，还可以直接是用数字表示文件的访问权限：

可执行-> 1

可写 -> 2

可读 ->4

系统调用 –打开

int open(const char \*pathname,int flags)

int open(const char \*pathname,int flags,mode\_t mode)

系统调用 –关闭

int close(int fd)

fd:文件描述符

系统调用 –读

int read(int fd,const void \*buf,size\_t length)

功能：从文件描述符fd所指定的文件中读取length个字节到buf所指定的缓冲区中，返回值为实际读取的字节数。

系统调用 –写

int write(int fd,const void \*buf,size\_t length)

系统调用 –定位

int lseek(int fd,offset\_t offect,int whence)

功能：将文件读写指针相对whence移动offset个字节。操作成功时，返回文件指针相对于文件头的位置。

whence可使用下述值：

SEEK\_SET：相对文件开头。

SEEK\_CUR：相对文件读写指针的当前位置。

SEEK\_END：相对文件末尾

offset可取负值，表示向前移动。

系统调用 –访问判断

int access(const char \*pathname,int mode)

pathname:文件名称

mode:要判断的访问权限。

返回值：当我们测试成功时，函数返回0，否则如果一个条件不符时，返回-1.

库函数—创建和打开

FILE \*fopen(const char \*filename,const char \*mode)

LINUX不区分二进制文件和文本文件。

库函数 –读

size-t fread(void \*ptr,size\_t size,size\_t n,FILE \*stream)

库函数 –写

size\_t fwrie(const void \*ptr,size\_t size,size\_t n,FILE \*stream)

库函数 –读字符

int fgetc(FILE \*stream)

库函数 –写字符

int fputc(int c,FILE \*stream)

库函数

fscanf (FILE \*stream,char \*format[,argument])

从一个流中格式化输入

库函数 –格式化写

int fprintf(FILE \*stream,char \*format[,argument,…])

格式化输出到一个流中

库函数 –定位

int fseek(FILE \*stream,long offset,int whence)

库函数 –路径获取

char \*getcwd(char \*buffer,size\_t size)

库函数 –创建目录

#include<sys/stat.h>

int mkdir(char \* dir,int mode)