## 实验十：文件系统编程 suof@lzu.edu.cn

返回[目录](#_目录)

### 实验目的：

学习和掌握使用系统调用操作文件和目录的属性、内容的方法

### 预备知识：

1. 系统调用
   1. 文件操作

1.1.1 open（打开文件）

表头文件 #include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

定义函数 int open( const char \* pathname, int flags);

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

函数说明 参数pathname 指向欲打开的文件路径字符串。下列是参数flags 所能使用的标志:

O\_RDONLY 以只读方式打开文件

O\_WRONLY 以只写方式打开文件

O\_RDWR 以可读写方式打开文件。

上述三种旗标是互斥的，也就是不可同时使用，但可与下列的旗标利用OR(|)运算符组合。

O\_CREAT 若欲打开的文件不存在则自动建立该文件。

O\_EXCL 如果O\_CREAT 也被设置，此指令会去检查文件是否存在。文件若不存在则建立该文件，否则将导致打开文件错误。此外，若O\_CREAT与O\_EXCL同时设置，并且欲打开的文件为符号连接，则会打开文件失败。

O\_NOCTTY 如果欲打开的文件为终端机设备时，则不会将该终端机当成进程控制终端机。

O\_TRUNC 若文件存在并且以可写的方式打开时，此旗标会令文件长度清为0，而原来存于该文件的资料也会消失。

O\_APPEND 当读写文件时会从文件尾开始移动，也就是所写入的数据会以附加的方式加入到文件后面。

O\_NONBLOCK 以不可阻断的方式打开文件，也就是无论有无数据读取或等待，都会立即返回进程之中。

O\_NDELAY 同O\_NONBLOCK。

O\_SYNC 以同步的方式打开文件。

O\_NOFOLLOW 如果参数pathname 所指的文件为一符号连接，则会令打开文件失败。

O\_DIRECTORY 如果参数pathname 所指的文件并非为一目录，则会令打开文件失败。

参数mode 则有下列数种组合，只有在建立新文件时才会生效，此外真正建文件时的权限会受到umask值所影响，因此该文件权限应该为（mode-umaks）。

S\_IRWXU 00700 权限，代表该文件所有者具有可读、可写及可执行的权限。

S\_IRUSR 或S\_IREAD，00400权限，代表该文件所有者具有可读取的权限。

S\_IWUSR 或S\_IWRITE，00200 权限，代表该文件所有者具有可写入的权限。

S\_IXUSR 或S\_IEXEC，00100 权限，代表该文件所有者具有可执行的权限。

S\_IRWXG 00070权限，代表该文件用户组具有可读、可写及可执行的权限。

S\_IRGRP 00040 权限，代表该文件用户组具有可读的权限。

S\_IWGRP 00020权限，代表该文件用户组具有可写入的权限。

S\_IXGRP 00010 权限，代表该文件用户组具有可执行的权限。

S\_IRWXO 00007权限，代表其他用户具有可读、可写及可执行的权限。

S\_IROTH 00004 权限，代表其他用户具有可读的权限

S\_IWOTH 00002权限，代表其他用户具有可写入的权限。

S\_IXOTH 00001 权限，代表其他用户具有可执行的权限。

返回值 若所有欲核查的权限都通过了检查则返回0 值，表示成功，只要有一个权限被禁止则返回-1。

错误代码 EEXIST 参数pathname 所指的文件已存在，却使用了O\_CREAT和O\_EXCL旗标。

EACCESS 参数pathname所指的文件不符合所要求测试的权限。

EROFS 欲测试写入权限的文件存在于只读文件系统内。

EFAULT 参数pathname指针超出可存取内存空间。

EINVAL 参数mode 不正确。

ENAMETOOLONG 参数pathname太长。

ENOTDIR 参数pathname不是目录。

ENOMEM 核心内存不足。

ELOOP 参数pathname有过多符号连接问题。

EIO I/O 存取错误。

1.1.2 close(见管道通信)

1.1.3 read(见管道通信)

1.1.4 write(见管道通信)

1.1.5 lseek（移动文件的读写位置）

表头文件 #include<sys/types.h>

#include<unistd.h>

定义函数 off\_t lseek(int fildes,off\_t offset ,int whence);

函数说明 每一个已打开的文件都有一个读写位置，当打开文件时通常其读写位置是指向文件开头，若是以附加的方式打开文件(如O\_APPEND)，则读写位置会指向文件尾。当read()或write()时，读写位置会随之增加，lseek()便是用来控制该文件的读写位置。参数fildes 为已打开的文件描述词，参数offset 为根据参数whence来移动读写位置的位移数。

参数 whence为下列其中一种:

SEEK\_SET 参数offset即为新的读写位置。

SEEK\_CUR 以目前的读写位置往后增加offset个位移量。

SEEK\_END 将读写位置指向文件尾后再增加offset个位移量。

当whence 值为SEEK\_CUR 或SEEK\_END时，参数offet允许负值的出现。

下列是较特别的使用方式:

1) 欲将读写位置移到文件开头时:lseek（int fildes,0,SEEK\_SET）；

2) 欲将读写位置移到文件尾时:lseek（int fildes，0,SEEK\_END）；

3) 想要取得目前文件位置时:lseek（int fildes，0,SEEK\_CUR）；

返回值 当调用成功时则返回目前的读写位置，也就是距离文件开头多少个字节。若有错误则返回-1，errno 会存放错误代码。

附加说明 Linux系统不允许lseek（）对tty装置作用，此项动作会令lseek（）返回ESPIPE。

1.1.6 fstat（由文件描述词取得文件状态）

表头文件 #include<sys/stat.h>

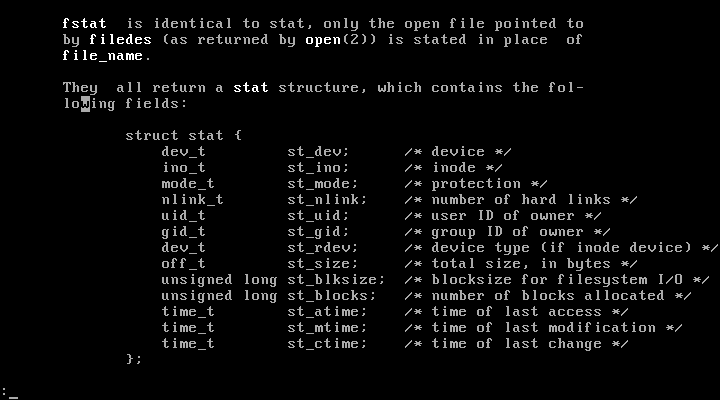
#include<unistd.h>

定义函数 int fstat(int fildes,struct stat \*buf);

函数说明 fstat()用来将参数fildes所指的文件状态，复制到参数buf所指的结构中(struct stat)。Fstat()与stat()作用完全相同，不同处在于传入的参数为已打开的文件描述词。

返回值 执行成功则返回0，失败返回-1，错误代码存于errno。

struct stat：



列数种情况

S\_IFMT 0170000 文件类型的位掩码

S\_IFSOCK 0140000 scoket

S\_IFLNK 0120000 符号连接

S\_IFREG 0100000 一般文件

S\_IFBLK 0060000 区块装置

S\_IFDIR 0040000 目录

S\_IFCHR 0020000 字符装置

S\_IFIFO 0010000 先进先出

S\_ISUID 04000 文件的（set user-id on execution）位

S\_ISGID 02000 文件的（set group-id on execution）位

S\_ISVTX 01000 文件的sticky位

S\_IRUSR（S\_IREAD） 00400 文件所有者具可读取权限

S\_IWUSR（S\_IWRITE）00200 文件所有者具可写入权限

S\_IXUSR（S\_IEXEC） 00100 文件所有者具可执行权限

S\_IRGRP 00040 用户组具可读取权限

S\_IWGRP 00020 用户组具可写入权限

S\_IXGRP 00010 用户组具可执行权限

S\_IROTH 00004 其他用户具可读取权限

S\_IWOTH 00002 其他用户具可写入权限

S\_IXOTH 00001 其他用户具可执行权限

上述的文件类型在POSIX 中定义了检查这些类型的宏定义

S\_ISLNK （st\_mode） 判断是否为符号连接

S\_ISREG （st\_mode） 是否为一般文件

S\_ISDIR （st\_mode）是否为目录

S\_ISCHR （st\_mode）是否为字符装置文件

S\_ISBLK （s3e） 是否为先进先出

S\_ISSOCK （st\_mode） 是否为socket

范例

#include<sys/stat.h>

#include<unistd.h>

#include<fcntk.h>

main()

{

struct stat buf;

int fd;

fd = open (“/etc/passwd”,O\_RDONLY);

fstat(fd,&buf);

printf(“/etc/passwd file size +%d\n “,buf.st\_size);

}

* 1. 文件权限操作

1.2.1 chown（改变文件的所有者）

表头文件 #include<sys/types.h>

#include<unistd.h>

定义函数 int chown(const char \* path, uid\_t owner,gid\_t group);

函数说明 chown()会将参数path指定文件的所有者变更为参数owner代表的用户，而将该文件的组变更为参数group组。如果参数owner或group为-1，对应的所有者或组不会有所改变。root与文件所有者皆可改变文件组，但所有者必须是参数group组的成员。当root用chown()改变文件所有者或组时，该文件若具有S\_ISUID或S\_ISGID权限，则会清除此权限位，此外如果具有S\_ISGID权限但不具S\_IXGRP位，则该文件会被强制锁定，文件模式会保留。

返回值 成功则返回0，失败返回-1，错误原因存于errno。

错误代码 参考chmod（）。

范例 /\* 将/etc/passwd 的所有者和组都设为root \*/

#include<sys/types.h>

#include<unistd.h>

main()

{

chown(“/etc/passwd”,0,0);

}

1.2.2 chmod（改变文件的权限）

表头文件 #include<sys/types.h>

#include<sys/stat.h>

定义函数 int chmod(const char \* path,mode\_t mode);

函数说明 chmod()会依参数mode 权限来更改参数path 指定文件的权限。

参数 mode 有下列数种组合

S\_ISUID 04000 文件的（set user-id on execution）位

S\_ISGID 02000 文件的（set group-id on execution）位

S\_ISVTX 01000 文件的sticky位

S\_IRUSR（S\_IREAD） 00400 文件所有者具可读取权限

S\_IWUSR（S\_IWRITE）00200 文件所有者具可写入权限

S\_IXUSR（S\_IEXEC） 00100 文件所有者具可执行权限

S\_IRGRP 00040 用户组具可读取权限

S\_IWGRP 00020 用户组具可写入权限

S\_IXGRP 00010 用户组具可执行权限

S\_IROTH 00004 其他用户具可读取权限

S\_IWOTH 00002 其他用户具可写入权限

S\_IXOTH 00001 其他用户具可执行权限

只有该文件的所有者或有效用户识别码为0，才可以修改该文件权限。基于系统安全，如果欲将数据写入一执行文件，而该执行文件具有S\_ISUID 或S\_ISGID 权限，则这两个位会被清除。如果一目录具有S\_ISUID 位权限，表示在此目录下只有该文件的所有者或root可以删除该文件。

返回值 权限改变成功返回0，失败返回-1，错误原因存于errno。

错误代码 EPERM 进程的有效用户识别码与欲修改权限的文件拥有者不同，而且也不具root权限。

EACCESS 参数path所指定的文件无法存取。

EROFS 欲写入权限的文件存在于只读文件系统内。

EFAULT 参数path指针超出可存取内存空间。

EINVAL 参数mode不正确

ENAMETOOLONG 参数path太长

ENOENT 指定的文件不存在

ENOTDIR 参数path路径并非一目录

ENOMEM 核心内存不足

ELOOP 参数path有过多符号连接问题。

EIO I/O 存取错误

范例 /\* 将/etc/passwd 文件权限设成S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IROTH \*/

#include<sys/types.h>

#include<sys/stat.h>

main()

{

chmod(“/etc/passwd”,S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IROTH);

}

* 1. 目录操作

1.3.1 getcwd（取得当前的工作目录）

表头文件 #include<unistd.h>

定义函数 char \* getcwd(char \* buf,size\_t size);

函数说明 getcwd()会将当前的工作目录绝对路径复制到参数buf所指的内存空间，参数size为buf的空间大小。在调用此函数时，buf所指的内存空间要足够大，若工作目录绝对路径的字符串长度超过参数size大小，则回值NULL，errno的值则为ERANGE。倘若参数buf为NULL，getcwd()会依参数size的大小自动配置内存(使用malloc())，如果参数size也为0，则getcwd()会依工作目录绝对路径的字符串程度来决定所配置的内存大小，进程可以在使用完此字符串后利用free()来释放此空间。

返回值 执行成功则将结果复制到参数buf所指的内存空间，或是返回自动配置的字符串指针。失败返回NULL，错误代码存于errno。

范例 #include<unistd.h>

main()

{

char buf[80];

getcwd(buf,sizeof(buf));

printf(“current working directory : %s\n”,buf);

}

1.3.2 chdir（改变当前的工作目录）

表头文件 #include<unistd.h>

定义函数 int chdir(const char \* path);

函数说明 chdir()用来将当前的工作目录改变成以参数path所指的目录。

返回值 执行成功则返回0，失败返回-1，errno为错误代码。

范例 #include<unistd.h>

main()

{

chdir(“/tmp”);

printf(“current working directory: %s\n”,getcwd(NULL,NULL));

}

1.3.3 opendir（打开目录）

表头文件 #include<sys/types.h>

#include<dirent.h>

定义函数 DIR \* opendir(const char \* name);

函数说明 opendir()用来打开参数name指定的目录，并返回DIR\*形态的目录流，和open()类似，接下来对目录的读取和搜索都要使用此返回值。

返回值 成功则返回DIR\* 型态的目录流，打开失败则返回NULL。

错误代码 EACCESS 权限不足

EMFILE 已达到进程可同时打开的文件数上限。

ENFILE 已达到系统可同时打开的文件数上限。

ENOTDIR 参数name非真正的目录

ENOENT 参数name 指定的目录不存在，或是参数name 为一空字符串。

ENOMEM 核心内存不足。

1.3.4 readdir（读取目录）

表头文件 #include<sys/types.h>

#include<dirent.h>

定义函数 struct dirent \* readdir(DIR \* dir);

函数说明 readdir()返回参数dir目录流的下个目录进入点。

结构dirent定义如下

struct dirent

{

ino\_t d\_ino; /\*此目录进入点的inode \*/

ff\_t d\_off; /\*目录文件开头至此目录进入点的位移\*/

signed short int d\_reclen; /\*文件名长度\*/

har d\_name[NAME\_MAX+1]; /\*文件名\*/

};

返回值 成功则返回下个目录进入点。有错误发生或读取到目录文件尾则返回NULL。

附加说明 EBADF参数dir为无效的目录流。

范例 #include<sys/types.h>

#include<dirent.h>

#include<unistd.h>

main()

{

DIR \* dir;

struct dirent \* ptr;

int i;

dir =opendir(“/etc/rc.d”);

while((ptr = readdir(dir))!=NULL)

{

printf(“d\_name: %s\n”,ptr->d\_name);

}

closedir(dir);

}

1.3.5 closedir（关闭目录）

表头文件 #include<sys/types.h>

#include<dirent.h>

定义函数 int closedir(DIR \*dir);

函数说明 closedir()关闭参数dir所指的目录流。

返回值 关闭成功则返回0，失败返回-1，错误原因存于errno 中。

错误代码 EBADF 参数dir为无效的目录流

1.3.6 mkdir(创建目录)

表头文件 #include<fcntl.h>

#include<unistd.h>

#include<sys/stat.h>

#include<sys/types.h>

定义函数 int mkdir (const char \*pathname, mode\_t mode);

函数说明 以mode(mode&~umask)为权限建立pathname指定的目录。

返回值 创建成功则返回0，失败返回-1，错误原因存于errno 中。

1.3.7 rmdir(删除目录)

表头文件 #include<fcntl.h>

#include<unistd.h>

#include<sys/stat.h>

#include<sys/types.h>

定义函数 int rmdir (const char \*pathname);

函数说明 删除pathname指定的目录

返回值 成功则返回0，失败返回-1，错误原因存于errno 中。

1. 标准库函数（略）

### 实验内容:

编写一个程序，实现将一个目录的所有内容复制到另一个目录的功能。要求：

源文件（目录）和目标文件（目录）的属主、权限等信息保持一致；

每复制一个文件（目录），在屏幕提示相应信息；

当遇到符号链接文件时，显示该文件为链接文件，不复制。

### 实验报告：

实现模拟cp的程序并写出报告，报告的要求同前。