**操作系统课程设计实验报告**

实验名称： 复制文件

姓名/学号： 刘鑫/1120181208

1. **实验目的**

掌握修改文件属性的方法，学会使用常用的API操作文件属性，理解平时操作系统复制文件背后具体的操作步骤

1. **实验内容**

完成一个目录复制命令mycp，包括目录下的文件和子目录。

Linux: creat，read，write等系统调用，要求支持软链接

Windows: CreateFile(), ReadFile(), WriteFile(), CloseHandle()等函数

特别注意复制后，不仅读写权限一致，而且时间属性也一致。

1. **实验环境**

Windows：

Windows10

处理器 Inter core [i5-8265u@1.60GHz](mailto:i5-8265u@1.60GHz)

内存 8.00GB

系统类型 64位操作系统，基于x64的处理器

Linux：

虚拟机软件：VMware workstation 15

虚拟机操作系统：Ubuntu20.04

虚拟机内存：4GB

虚拟机硬盘容量:60GB

1. **程序设计与实现**

**【Windows部分】：**

程序主要分为三个部分：main函数，cpdir函数，cpfile函数，整体采用递归的方法来实现文件的复制

**一、main函数部分：**

先判断运行时给的参数数量是否正确，然后寻找原文件，创建目标目录，最后调用cpdir函数来复制文件到目标目录中，主要涉及到的函数和结构体如下：

1. WIN32\_FIND\_DATA结构体

用于存储文件全部属性信息的结构体，可以通过FindFirstFile（）函数根据当前的文件存放路径查找该文件来把待操作文件的相关属性读取到WIN32\_FIND\_DATA结构中去

1. FindFirstFile()系统函数

函数功能描述:查找指定目录的第一个文件或目录并返回它的句柄

函数原型：

HANDLE FindFirstFile(  
                                LPCTSTR lpFileName,                       // 文件名  
                                LPWIN32\_FIND\_DATA lpFindFileData   // 用于存储文件 信息的结构体  
                              );

1. CreateDirectory()系统函数

函数功能描述：创建一个新目录，返回值为long，非零表示创建成功，零表示创建失败

函数原型：

BOOL CreateDirectory(  
  LPCTSTR lpPathName, //目标目录名  
  LPSECURITY\_ATTRIBUTES lpSecurityAttributes//忽略，一般设置为NULL  
);

1. cpdir()用户自定义函数

函数功能描述：用于递归复制文件目录，内部调用cpfile()函数来对文件进行复制，并实现修改文件和目录时间权限等属性的操作。

函数原型：

void cpdir(

LPCTSTR f1, //原文件的文件名

LPCTSTR f2 //目标文件的文件名

)

**二、cpdir部分**

主要步骤**：**

1. 新建WIN32\_FIND\_DATA结构体，用于存储文件的属性信息
2. 拼接目标文件和原文件的文件路径，获得当前递归层下的文件或目录名
3. 调用FindFirstFile函数查找当前递归层的原文件和已经创建的目标文件，并将对应的文件信息保存在WIN32\_FIND\_DATA结构体中
4. 调用FindNextFile函数寻找当前目录下，下一个需要复制的文件，返回文件句柄并将文件信息存储到WIN32\_FIND\_DATA结构体中
5. 通过WIN32\_FIND\_DATA结构体中的cFileName获取需要被复制的文件的名字，与当前目录拼接成完整文件名
6. 判断⑤中获得的文件名是否是当前目录名或者父目录名，如是，结束函数，否则继续执行函数
7. 若⑤中的文件名不是目录，则直接调用cpfile进行复制并修改文件属性
8. 若⑤中的文件名既不是文件，也不是当前目录和父目录，则先递归调用cpdir，再修改当前创建的文件夹的时间属性

主要涉及到的函数和结构体：

1. WIN32\_FIND\_DATA结构体
2. FindFirstFile系统函数
3. FindNextFile系统函数

函数功能描述：主要用于遍历目录与文件，判断当前目录是否有下一个文件或目录，若有，则返回TRUE，并将查找到的文件或目录信息装载到WIN32\_FIND\_DATA结构体中，若没有，则返回FALSE

函数原型：

BOOLFindNextFile(

HANDLE hFindFile, //搜索的文件名

LPWIN32\_FIND\_DATA lpFindFileData //WIN32\_FIND\_DATA结构体

);

1. CreateFile系统函数

函数功能描述：可打开或创建文件或者I/O设备，并返回可访问的句柄

函数原型：

HANDLE CreateFile(  
 LPCTSTR lpFileName,    // 指向文件名的指针   
 DWORD dwDesiredAccess,    // 访问模式（写 / 读）   
 DWORD dwShareMode,    // 共享模式   
 LPSECURITY\_ATTRIBUTES lpSecurityAttributes, // 指向安全属性的指针   
 DWORD dwCreationDisposition,   // 如何创建   
 DWORD dwFlagsAndAttributes,   // 文件属性   
 HANDLE hTemplateFile    // 用于复制文件句柄   
);

参数：

主要介绍两个参数

1. dwCreationDisposition

取值：

OPEN\_EXISTING | OPEN\_ALWAYS

文件必须已经存在。由设备提出要求 | 文件不存在则创建它

1. wFlagsAndAttributes

取值：FILE\_ATTRIBUTE\_NORMAL，表示默认属性

1. cpfile用户自定义函数

函数功能描述：实现对文件的复制已经文件时间权限等熟悉值的修改

函数原型：

void cpfile(

LPCTSTR f1. //原文件的文件名

LPCTSTR f2 //目标文件的文件名

)

**三、cpfile部分：**

主要步骤：

①新建WIN32\_FIND\_DATA结构体，用于保存原文件的文件信息

②暂时修改原文件的文件属性为FILE\_ATTRIBUTE\_NORMAL

③调用以可读可写的方式打开原文件，同时以可读可写的方式创建并打开目标文件，暂时设置目标文件的文件属性为FILE\_ATTRIBUTE\_NORMAL

④调用SetFileTime函数修改目标文件时间

⑤通过WIN32\_FIND\_DATA中的信息获取原文件大小并创建初始化缓冲区

⑥创建文件读写指针

⑦调用ReadFile函数将原文件数据写入到缓冲区

⑧调用WriteFile函数将缓冲区的内容写入到目标文件

1. 调用SetFileAttributes函数修改原文件和目标文件属性为初始值
2. 关闭所有文件句柄，结束

主要涉及到的函数和数据结构

1. WIN32\_FIND\_DATA结构体
2. FindFirstFile系统函数
3. SetFileAttributes系统函数

函数功能描述：设置文件属性

函数原型：

BOOL SetFileAttributesA(

LPCSTR lpFileName, //文件名

DWORD dwFileAttributes //属性值

);

1. CreateFile系统函数
2. SetFileTime系统函数

函数功能描述：设置文件的创建、访问以及上次修改时间

函数原型：

BOOL SetFileTime(

HANDLE hFile, //文件句柄

const FILETIME \*lpCreationTime,//文件创建时间

const FILETIME \*lpLastAccessTime,//上次文件访问时间

const FILETIME \*lpLastWriteTime//最近文件被修改时间

);

1. buffer缓冲区
2. DWORD读写指针
3. ReadFile系统函数

函数功能表述：从文件指针指向的位置开始将数据读出到一个文件中，调用成功返回非0，调用不成功返回0

函数原型：

BOOL ReadFile(

    HANDLE hFile, //文件的句柄

    LPVOID lpBuffer,//用于保存读入数据的一个缓冲区

    DWORD nNumberOfBytesToRead, //要读入的字节数

    LPDWORD lpNumberOfBytesRead, //指向实际读取字//节数的指针

    LPOVERLAPPED lpOverlapped

    //如文件打开时指定了FILE\_FLAG\_OVERLAPPED，那么必须，用这个参数引用一个特殊的结构。

    //该结构定义了一次异步读取操作。否则，应将这个参数设为NULL

);

1. WriteFile系统函数

函数功能描述：将数据写入一个文件或者I/O设备，调用成功返回非0，调用不成功返回0

函数原型：

BOOL WriteFile(

HANDLE hFile,//文件句柄

LPCVOID lpBuffer,//数据缓存区指针

DWORD nNumberOfBytesToWrite,//你要写的字节数

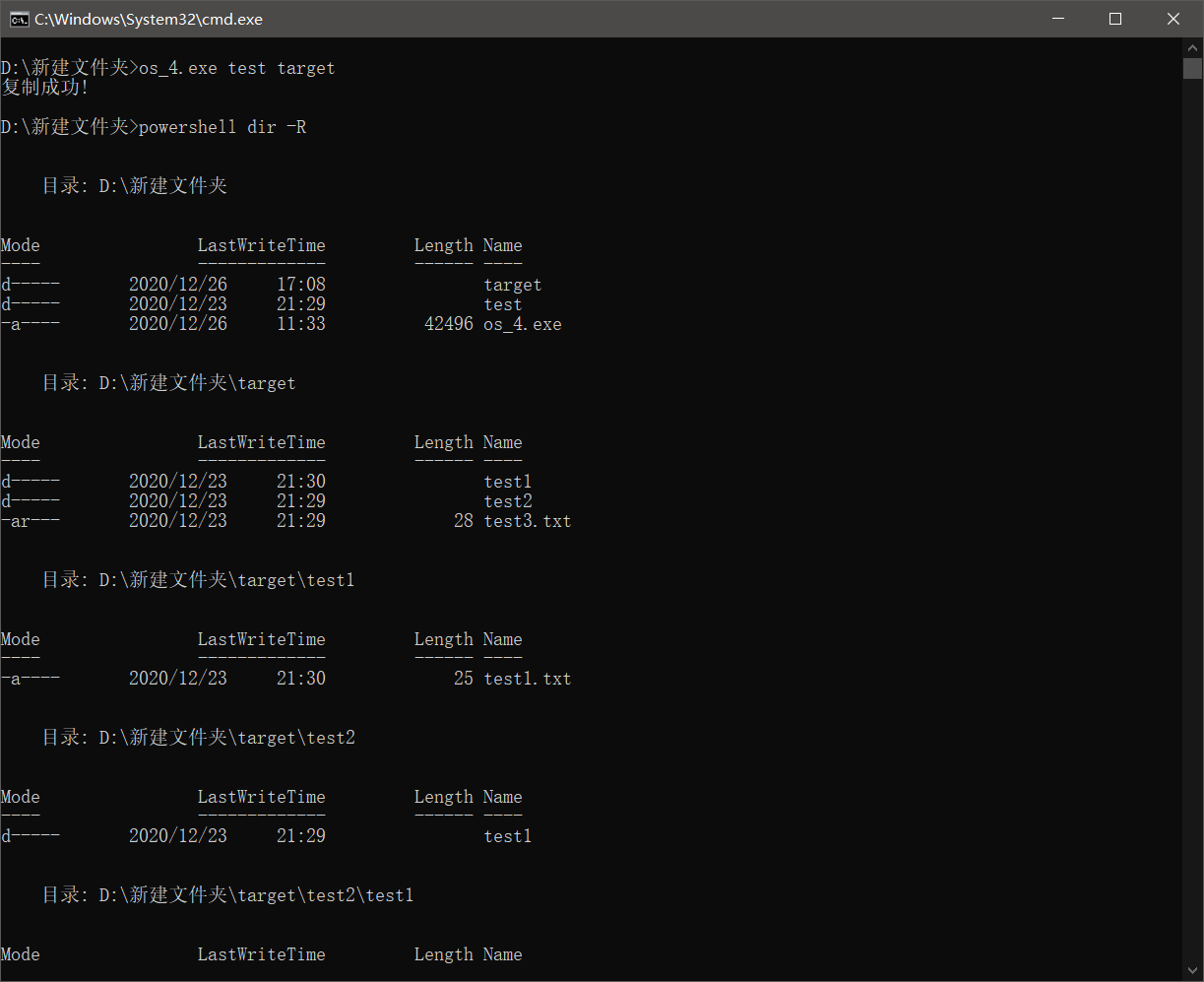
LPDWORD lpNumberOfBytesWritten,//用于保存实际写入字节数的存储区域的指针

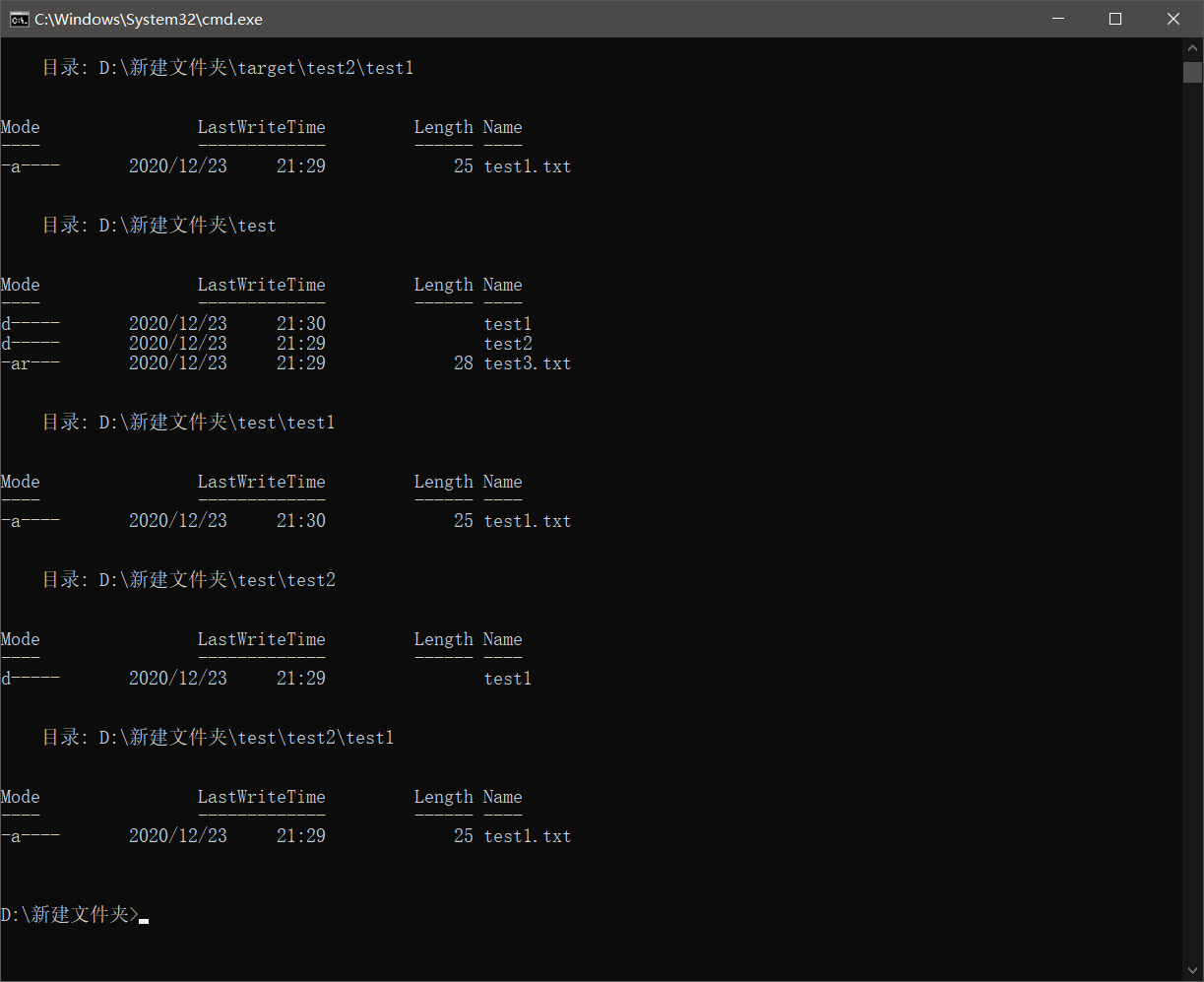
LPOVERLAPPED lpOverlapped//OVERLAPPED结构体指针

);

1. CloseHandle系统函数

程序运行结果：





Windows部分实验成功！

**【Linux部分】：**

程序整体上分为三个部分：main函数部分，myCopyAllFile函数部分和CopyFile函数部分

1. main函数部分：

主要步骤：

* 1. 判断main函数参数数量是否正确
  2. 判断原文件是否能正常打开
  3. 创建目标文件最外层目录并修改时间属性
  4. 调用myCopyFile函数对内部文件进行递归复制

主要涉及的函数和数据结构

1. struct stat

是用来描述一个linux系统文件系统中的文件属性的结构。可以通过stat和lstat函数来获取文件属性

结构：

struct stat {

        mode\_t   st\_mode;//文件对应的模式，文件，目录等

        ino\_t   st\_ino;//inode节点号

        dev\_t   st\_dev;//设备号码

        dev\_t   st\_rdev;//特殊设备号码

        nlink\_t  st\_nlink;//文件的连接数

        uid\_t   st\_uid;//文件所有者

        gid\_t   st\_gid;//文件所有者对应的组

        off\_t   st\_size;//普通文件，对应的文件字节数

        time\_t   st\_atime;//文件最后被访问的时间

        time\_t   st\_mtime;//文件内容最后被修改的时间

        time\_t   st\_ctime;//文件状态改变时间

        blksize\_t st\_blksize;//文件内容对应的块大小

        blkcnt\_t  st\_blocks;//伟建内容对应的块数量

      };

1. struct utmbuf结构体

用于存储文件时间相关属性值的结构

结构：

struct utimbuf

{

time\_t actime;

time\_t modtime;

};

1. DIR\*

目录指针

1. opendir系统函数

函数功能描述：打开指定的目录，并返回DIR\*形态的目录流，操作不成功返回NULL

函数原型：

DIR \* opendir(const char \* name);

1. stat系统函数

函数功能描述：获取指定的文件状态

函数原型：

int stat(

const char \* file\_name, //文件名

struct stat \*buf //用于存储文件状态的结构

);

1. mkdir系统函数

函数功能描述：以指定方式创建一个目录，若目录创建成功返回0，创建失败返回-1

函数原型：

 int mkdir(

const char \*pathname, //创建的目录名

mode\_t mode //指定模式

);

1. myCopyAllFile用户自定义函数

函数功能描述：递归地对文件进行复制，并修改目标文件属性使其与原文件属性一致

函数原型：

void myCopyAllFile(

char\* f1, //原文件文件名

char\* f2 //目标文件名

);

1. myCopyFile函数部分

主要步骤：

* + - * 1. 新建数据结构，存储文件目录的属性信息
        2. 调用makedir函数创建目标目录
        3. 调用readir读取原文件目录
        4. 根据读取的文件不同进行不同的操作，若读取的是符号链接文件，调用symlink函数创建目标文件内的符号链接文件，调用lstat函数获取原文件软连接的时间属性，存储到struct timeval结构体中，最后修改目标文件软连接的时间使之与原文件的时间一样
        5. 如果读取到的是目录，递归调用myCopyFile函数
        6. 如果读取到的是文件，调用CopyFile函数复制文件并修改文件属性
        7. 最后修改当前递归层数下目录的时间信息

主要涉及的结构体和函数

1. struct stat
2. struct utimbuf
3. struct dirent

用于存储目录信息的结构体

原型：

struct dirent

{

long d\_ino; /\* inode number 索引节点号 \*/

off\_t d\_off; /\* offset to this dirent 在目录文件中的偏移 \*/

unsigned short d\_reclen; /\* length of this d\_name 文件名长 \*/

unsigned char d\_type; /\* the type of d\_name 文件类型 \*/

char d\_name [NAME\_MAX+1]; /\* file name (null-terminated) 文件名，最长255字符 \*/

}

1. DIR\*
2. opendir系统函数
3. stat系统函数
4. lstat系统函数

函数功能描述：用于获取文件或者符号连接的信息

函数原型：

int lstat(const char \*path, struct stat \*buf);

1. mkdir系统函数
2. readdir系统函数

函数功能描述：返回参数dir 目录流的下个目录进入点

函数原型：

struct dirent \* readdir(DIR \* dir);

1. readlink系统函数

函数功能描述：

readlink()会将参数path 的符号连接内容存到参数buf 所指的内存空间, 返回的内容不是以NULL作字符串结尾, 但会将字符串的字符数返回. 若参数bufsiz 小于符号连接的内容长度, 过长的内容会被截断

函数原型：

int readlink(const char \* path, char \* buf, size\_t bufsiz);

1. symlink系统函数

函数功能描述：

以指定的名称创建一个新连接，创建成功返回0，失败返回-1

函数原型：

int symlink(

const char \* oldpath, //原文件名

const char \* newpath //创建的新连接的连接名

);

1. lutimes系统函数

函数功能描述：修改文件或者连接的时间属性

函数原型：

int lutimes(const char \*filename, const struct timeval tv[2])

1. utime系统函数

函数功能描述：

修改指定文件所属的inode的存取时间

函数原型：

int utime(const char \* filename, struct utimbuf \* buf);

1. CopyFile函数部分

主要步骤：

* + - * 1. 打开原文件，返回文件句柄
        2. 创建缓冲区并初始化
        3. 调用creat函数创建目标文件，并修改文件属性
        4. 调用read函数将原文件的内容写入到缓冲区中
        5. 调用write函数将缓冲区中的内容写入到新创建的目标文件中
        6. 修改目标文件的时间属性
        7. 调用close函数关闭文件句柄

主要涉及的函数以及结构体

1. struct stat结构体
2. struct utimbuf结构体
3. stat系统函数
4. open系统函数
5. creat系统函数

函数功能描述：以指定模式创建文件

函数原型：

int creat(const char \* pathname, mode\_tmode);

1. read系统函数

函数功能描述：从打开的设备或文件中读取数据，成功返回读取的字节数，出错返回-1并设置errno

函数原型：

ssize\_t read(int fd, void \*buf, size\_t count);

1. write系统函数

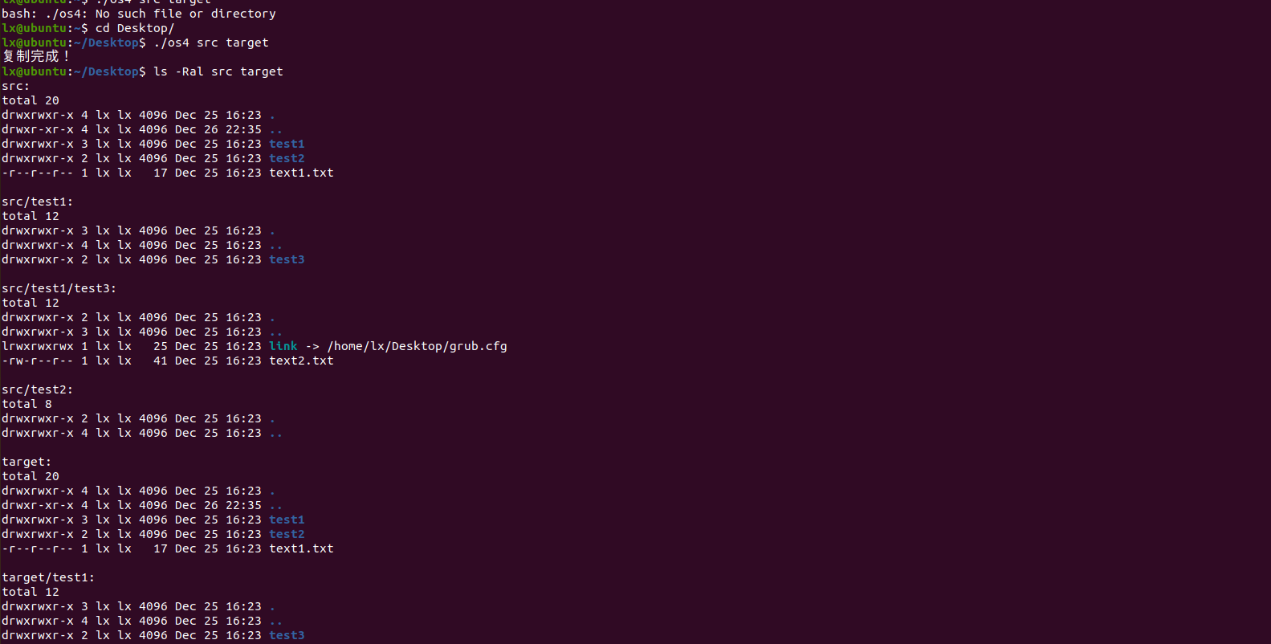
函数功能描述：将指定缓冲区中的数据写入到文件中，成功执行会返回写入的字节数，发生错误则返回-1

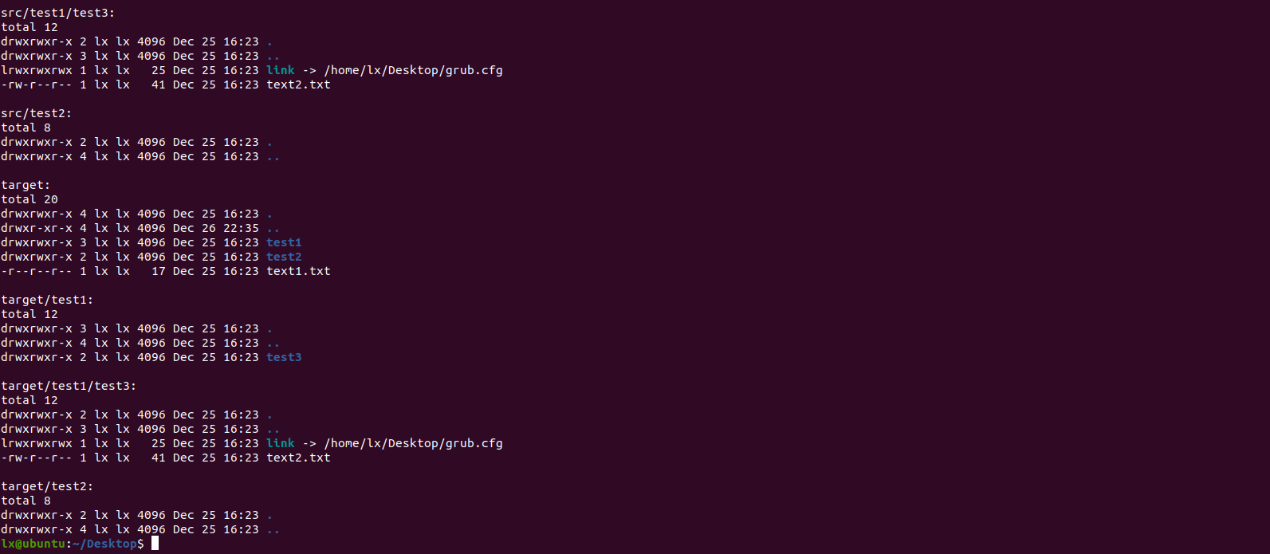
函数原型：

ssize\_t write (int fd, const void \* buf, size\_t count);

1. close系统函数

运行结果：





Linux部分实验成功！

1. **实验收获与体会**

在这次实验中，我对操作系统的文件复制有了更加深入和本质的认识，了解了不同系统下目录和文件存储机制的，同时对于文件和目录操作的API有了大致的了解，也对相关的数据结构操作更加熟悉。