1. **实验目的及内容**

完成一个目录复制命令mycp，包括目录下的文件和子目录。

需要实现Windows版本和Linux版本。

目录拷贝时需要支持多级目录（子目录）的拷贝，需要支持Linux里的soft link和windows中的快捷方式拷贝。

说明：

Linux: creat，read，write等系统调用

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

**二、实验环境**

操作系统：windows10 64位

编译器：Visual Studio 2012

Linux Ubuntu 14.04.5

编译器：gcc

**三、基本原理及方法概述**

**3.1 windows实现**

使用CreateDirectory()在指定路径下创建目录；

使用FindFirstFile()查找指定文件路径的文件；

使用CreateFile()打开或创建文件；

使用ReadFile()读文件；

使用WriteFile()写文件；

使用WIN32\_FIND\_DATA结构体保存文件属性信息；

**3.1.1用到的系统调用函数及结构体**

**(1)WIN32\_FIND\_DATA结构体**

数据结构WIN32\_FIND\_DATA的[成员变量](https://baike.baidu.com/item/%E6%88%90%E5%91%98%E5%8F%98%E9%87%8F" \t "https://baike.baidu.com/item/WIN32_FIND_DATA/_blank)里包含了所有的[文件属性](https://baike.baidu.com/item/%E6%96%87%E4%BB%B6%E5%B1%9E%E6%80%A7" \t "https://baike.baidu.com/item/WIN32_FIND_DATA/_blank)，可以通过这个结构作为获取和更改文件属性的手段。

typedef struct \_WIN32\_FIND\_DATA {

DWORD dwFileAttributes; //文件属性

FILETIME ftCreationTime; // 文件创建时间

FILETIME ftLastAccessTime; // 文件最后一次访问时间

FILETIME ftLastWriteTime; // 文件最后一次修改时间

DWORD nFileSizeHigh; // 文件长度高32位

DWORD nFileSizeLow; // 文件长度低32位

DWORD dwReserved0; // 系统保留

DWORD dwReserved1; // 系统保留

TCHAR cFileName[ MAX\_PATH ]; // 长文件名

TCHAR cAlternateFileName[ 14 ]; // 8.3格式文件名

} WIN32\_FIND\_DATA, \*PWIN32\_FIND\_DATA;

**(2) FindFirstFile()函数**

查找指定文件路径的文件

FindFirstFile

HANDLE FindFirstFile(

LPCTSTRlpFileName, //filename

LPWIN32\_FIND\_DATAlpFindFileData //databuffer

);

调用成功返回一个句柄；

调用失败 返回为INVALID\_HANDLE\_VALUE

**(3) FindNextFile()函数**

根据指定的一个文件名查找下一个文件

BOOL FindNextFile(

HANDLE hFindFile, //调用FindFirstFile返回的句柄

LPWIN32\_FIND\_DATA lpFindFileData //结构地址

);

**(4) CreateDirectory()函数**

创建目录

BOOL CreateDirectory(

LPCTSTR lpPathName, //新创建目录的路径名

LPSECURITY\_ATTRIBUTES lpSecurityAttributes //安全属性

);

**(5) CreateFile()函数**

可打开或创建以下对象，并返回可访问的句柄

HANDLE WINAPI CreateFile(

\_In\_ LPCTSTR lpFileName,

\_In\_ DWORD dwDesiredAccess, //访问方式

\_In\_ DWORD dwShareMode, //共享方式

\_In\_opt\_ LPSECURITY\_ATTRIBUTES lpSecurityAttributes,//安全属性

\_In\_ DWORD dwCreationDisposition,//OPEN\_EXISTING 文件必须已经存在

\_In\_ DWORD dwFlagsAndAttributes, //指示系统为文件的打开或创建执行一个备份或恢复操作. 系统保证调用进程忽略文件的安全选项

\_In\_opt\_ HANDLE hTemplateFile

);

**(6) ReadFile()函数**

从文件中读取数据

BOOL ReadFile(

HANDLE hFile, //文件的句柄

LPVOID lpBuffer, //保存读入数据的一个缓冲区

DWORD nNumberOfBytesToRead, //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

);

**(7) WriteFile()函数**

向文件中写数据

BOOL WriteFile(

HANDLE hFile, //文件的句柄

LPVOID lpBuffer, //保存写入数据的一个缓冲区

DWORD nNumberOfBytesToRead, //写入的字节数

LPDWORD lpNumberOfBytesRead, //指向实际写入字节数的指针

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

);

**3.1.2程序的主要函数说明**

1. **void copydir(char \*path1,char \*path2)**

参数path1为源目录，path2为目标目录

此函数主要功能为获取源目录中的所有文件的句柄，判断其类型。若为目录则创建新的目录后递归调用自身，若为文件，则在目标目录中创建新的文件并将原文件的内容拷贝。

1. **void copyfile(char \*path1,char \*path2)**

参数path1为源目录，path2为目标目录

此函数的主要功能为将一个文件的内容复制到另一个文件。用CreateFile函数打开源文件和创建新的文件。分别用ReadFile和WriteFile函数来读写文件。在读写前通过FindFirstFile获取文件的属性信息，并根据属性信息中文件的大小信息求出文件的大小，一次完成整个文件的读写。

**3.2 linux实现**

**3.2.1用到的系统调用函数及结构体**

1. **stat结构体**

描述linux系统文件系统中的文件属性的结构

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. **stst() 与lstat函数**

获取一个文件的属性

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

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

两个函数的第一个参数都是文件的路径，第二个参数是struct stat的指针。返回值为0，表示成功执行。

**(3)opendir()函数**

打开一个目录，在失败的时候返回一个空的指针。

DIR\* opendir (const char \* path ); （获取path子目录下的所有文件和目录的列表，如果path是个文件则返回值为NULL）

**(4)readdir()函数**

读取opendir 返回值的那个列表

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

循环读取dir\_handle,目录和文件都读。

1. **mkdir()函数**

创建一个子目录

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

mkdir()函数以mode方式创建一个以参数pathname命名的目录，mode定义新创建目录的权限。

1. **creat()函数**

创建新文件

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

pathname是要创建文件的路径名，mode用来规定文件的属性

1. **open()函数**

打开文件

int open(const char \*pathname,int oflg[,mode\_t mode]);

Pathname是待打开的文件路径名，参数oflg是文件的打开方法。

1. **read()和write()函数**

读和写文件

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

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

fd为文件描述符，buf为读出或写入文件数据的字节数组，第三个参数是要传递的字节个数。

1. **reaklink()函数**

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

readlink()会将参数path的[符号链接](http://baike.baidu.com/view/1955541.htm" \t "https://blog.csdn.net/jk198310/article/details/_blank)内容存储到参数buf所指的内存空间。

**(10)symlink()函数**

创建一个符号链接

int symlink(const char \*actualpath,const char\*sympath) ;  
返回：若成功则为0若出错则为-1。  
该函数创建了一个指向actualpath的新目录项sympath在创建此符号连接时，并不要求actualpath已经存在，并且，actualpath和sympath并不需要位于同一文件系统中。

**3.2.2程序用到的主要函数说明**

1. **void copydir(char \*path1,char \*path2)**

参数path1为源目录，path2为目标目录

此函数的主要功能是根据readdir()获取的文件及目录判断出他们的类型，若文件属性为目录则创建新的目录并递归调用自身，若为文件则调用copyfile函数将源文件复制到目标目录下。

**(2)void copyfile(char \*path1,char \*path2)**

参数path1为源目录，path2为目标目录

此函数的主要功能为利用creat()函数在指定位置创建文件后利用open()函数打开源文件并用read()函数从源文件中读取指定长度的字节之后利用write()函数写入新创建的文件，循环读出并写入直至文件读完为止。

**四、实验结果截图及分析**

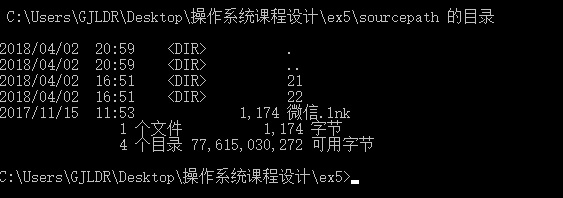
**4.1 windows**

运行cmd后输入windows-mycp.exe sourcepath despath命令进行复制结果如下图所示。

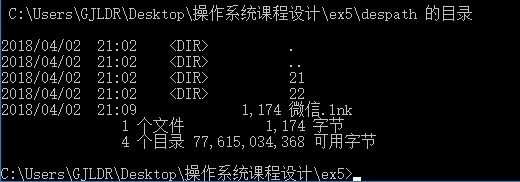
****

用dir命令对比复制前后文件夹的情况

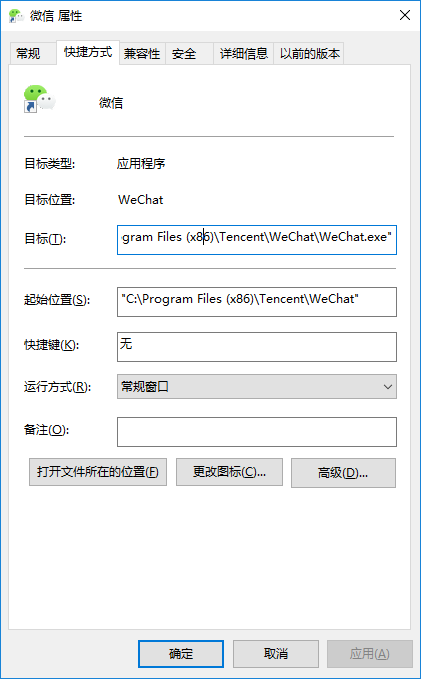
源文件夹Sourcepath

****

目标文件夹despath

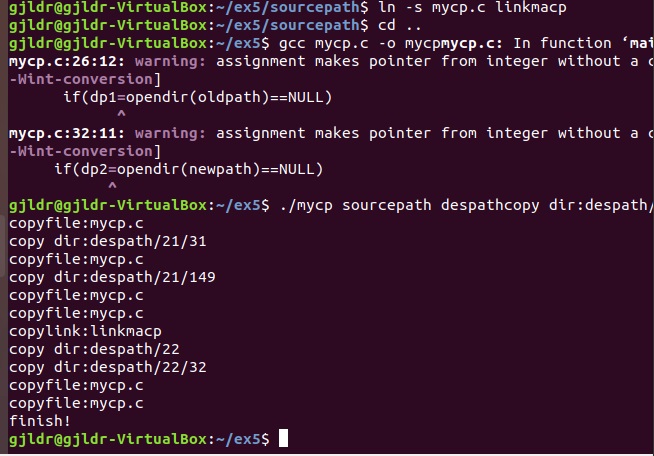


对比复制前后文件的目录情况完全相同，且目录中的微信.1nk快捷方式复制后可正常运行。

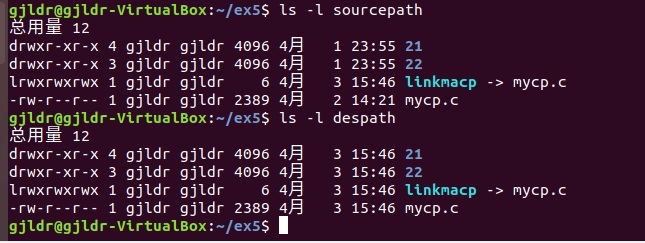


**4.2 linux**

输入命令gcc mycp.c -o mycp编译源代码后输入./mycp sourcepath despath命令运行，结果如下图所示。



用ls -l pathname 命令对比复制前后的文件列表。



对比复制前后文件列表相同，且软链接文件可正常打开。

