
操作系统课程设计实验报告

实验名称： _____ 文件复制 _____

姓名/学号： _____ 张惟振/1120170117 _____

一、 实验目的

了解在 Windows 中，如何通过文件系统提供的各种 API，对文件进行一系列操作，并深入理解 Windows 文件系统的功能和作用。

熟悉 Linux 文件系统提供的有关文件操作的系统调用函数。通过使用文件系统的系统调用命令操作文件，以实现对文件系统功能的理解和掌握。

二、 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，运行结果如下：

```
beta@bugs.com [~/]# ls -la sem
total 56
drwxr-xr-x  3 beta beta 4096 Dec 19 02:53 ./
drwxr-xr-x  8 beta beta 4096 Nov 27 08:49 ../
-rw-r--r--  1 beta beta  128 Nov 27 09:31 Makefile
-rwxr-xr-x  1 beta beta 5705 Nov 27 08:50 consumer*
-rw-r--r--  1 beta beta  349 Nov 27 09:30 consumer.c
drwxr-xr-x  2 beta beta 4096 Dec 19 02:53 subdir/

beta@bugs.com [~/]# mycp sem target
beta@bugs.com [~/]# ls -la target
total 56
drwxr-xr-x  3 beta beta 4096 Dec 19 02:53 ./
drwxr-xr-x  8 beta beta 4096 Nov 27 08:49 ../
-rw-r--r--  1 beta beta  128 Nov 27 09:31 Makefile
-rwxr-xr-x  1 beta beta 5705 Nov 27 08:50 consumer*
-rw-r--r--  1 beta beta  349 Nov 27 09:30 consumer.c
drwxr-xr-x  2 beta beta 4096 Dec 19 02:53 subdir/
```

说明：

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, 不仅权限一致, 而且时间属性也一致。

三、 实验环境

1、软件环境

Windows10 操作系统、Ubuntu 18.04.3 LTS

2、硬件环境

Intel® Core™ i5-7200U CPU @ 2.50GHz×4

四、 程序设计 with 实现

1、实验思路: 遍历所要复制的目录下的所有文件, 分别编写目录复制、文件复制的函数, Linux 中额外编写一个软链接文件复制的函数, 程序首先调用目录复制函数, 在对文件进行的遍历过程中如遇到子目录则递归调用目录复制函数, Linux 中如遇到软链接文件则调用软链接文件复制函数, 遇到其他文件则调用文件复制函数, 直至所有文件遍历完成。在文件和目录复制完成后对其时间属性进行修改。

2、Windows 系统实现中的 API

(1) FindFirstFile() 打开一个目录, 在失败时返回一个 INVALID_HANDLE_VALUE 值, 用于检测输入路径是否存在

实验使用如下:

```
if (FindFirstFile(argv[1], &lpfindfiledata) == INVALID_HANDLE_VALUE) //查找文件
{
    printf("findfirstfile error!\n");
    exit(0);
}
```

(2) CreateDirectory () 创建一个目录

实验使用如下:

```
CreateDirectory(argv[2], NULL); //创建目录文件
```

(3) FindNextFile () 遍历目录或文件时, 判断当前目录下是否有下一个目录或文件

实验使用如下:

```
for (; FindNextFile(hfindfile, &lpfindfiledata) != 0;) //遍历所有文件
{
```

(4) GetFileTime () 获取文件的时间信息

实验使用如下：

```
GetFileTime(hsourcefile, &creationtime, &accesstime, &writetime);
```

(5) SetFileTime () 设置文件的创建、访问及上次修改时间

实验使用如下：

```
SetFileTime(hobjectfile, &creationtime, &accesstime, &writetime);
```

(6) CreateFile () 创建或打开一个文件

实验使用如下：

```
HANDLE hsource = CreateFile(sourcefile, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);  
// 文件名, 访问模式, 共享模式, 安全属性, 如何创建, 属性, 句柄  
HANDLE hobject = CreateFile(objectfile, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
```

(7) ReadFile () 从打开的文件中读取数据

实验使用如下：

```
ReadFile(hsource, buffer, size, &temp, NULL);
```

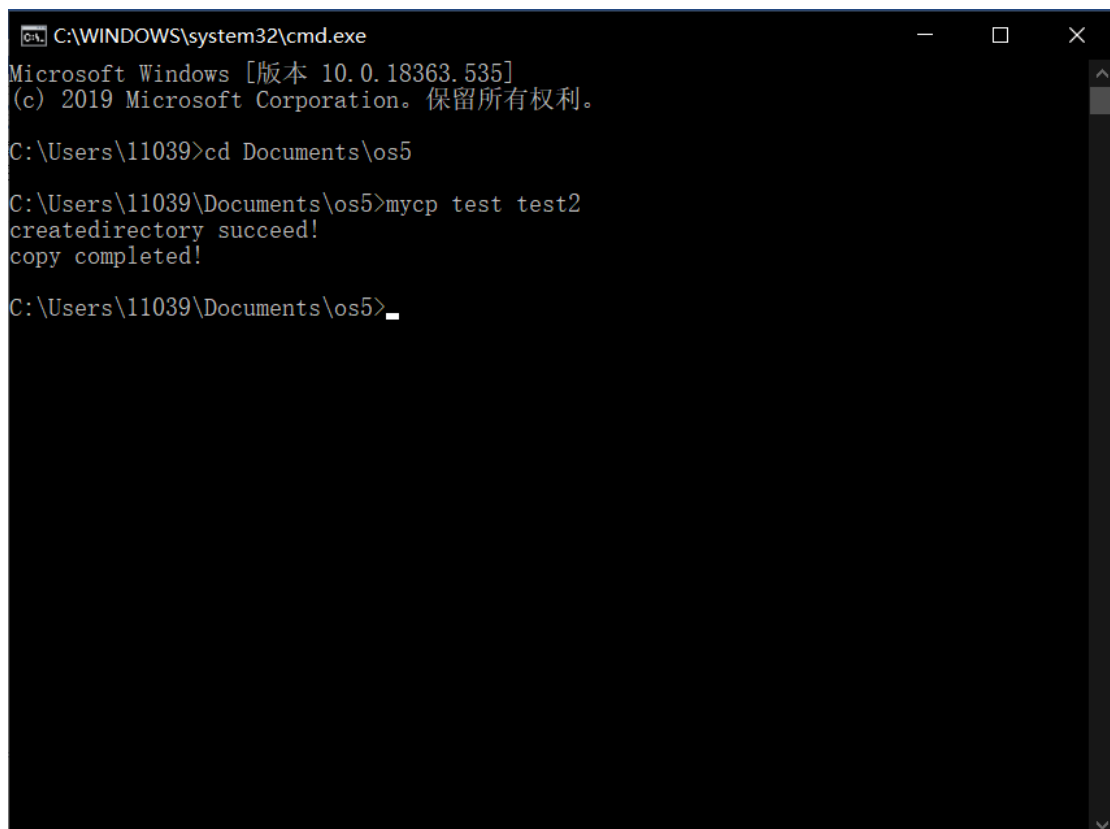
(8) WriteFile () 将数据写入文件中

实验使用如下：

```
WriteFile(hobject, buffer, size, &temp, NULL);
```

3、Windows 下实验结果

复制文件



```
C:\WINDOWS\system32\cmd.exe  
Microsoft Windows [版本 10.0.18363.535]  
(c) 2019 Microsoft Corporation。保留所有权利。  
  
C:\Users\11039>cd Documents\os5  
  
C:\Users\11039\Documents\os5>mycp test test2  
createdirectory succeed!  
copy completed!  
  
C:\Users\11039\Documents\os5>_
```

检验复制结果

```
C:\WINDOWS\system32\cmd.exe

C:\Users\11039\Documents\os5>dir test
驱动器 C 中的卷是 Windows
卷的序列号是 022F-0929

C:\Users\11039\Documents\os5\test 的目录
2019.12.18 15:09 <DIR> .
2019.12.18 15:09 <DIR> ..
2019.12.18 15:09          37 md. txt
2019.12.15 17:30      12,939 nm. docx
2019.12.16 15:12 <DIR> sub
                2 个文件          12,976 字节
                3 个目录 44,460,388,352 可用字节

C:\Users\11039\Documents\os5>dir test2
驱动器 C 中的卷是 Windows
卷的序列号是 022F-0929

C:\Users\11039\Documents\os5\test2 的目录
2019.12.18 15:09 <DIR> .
2019.12.18 15:09 <DIR> ..
2019.12.18 15:09          37 md. txt
2019.12.15 17:30      12,939 nm. docx
2019.12.16 15:12 <DIR> sub
                2 个文件          12,976 字节
                3 个目录 44,460,388,352 可用字节

C:\Users\11039\Documents\os5>
```

子目录下的实验结果

```
C:\WINDOWS\system32\cmd.exe

                3 个目录 44,460,388,352 可用字节

C:\Users\11039\Documents\os5>dir test\sub
驱动器 C 中的卷是 Windows
卷的序列号是 022F-0929

C:\Users\11039\Documents\os5\test\sub 的目录
2019.12.16 15:12 <DIR> .
2019.12.16 15:12 <DIR> ..
2019.12.18 15:09      271 nnn. txt
                1 个文件          271 字节
                2 个目录 44,459,339,776 可用字节

C:\Users\11039\Documents\os5>dir test2\sub
驱动器 C 中的卷是 Windows
卷的序列号是 022F-0929

C:\Users\11039\Documents\os5\test2\sub 的目录
2019.12.16 15:12 <DIR> .
2019.12.16 15:12 <DIR> ..
2019.12.18 15:09      271 nnn. txt
                1 个文件          271 字节
                2 个目录 44,459,339,776 可用字节

C:\Users\11039\Documents\os5>
```

4、Linux 系统实现中的 API

(1) opendir () 打开一个目录，在失败的时候返回一个空的指针，用于检测输入路径是否存在

实验使用如下：

```
if ((dir = opendir(argv[1])) == NULL) //查找目录
{
    printf("opendir error!\n");
    exit(0);
}
```

(2) mkdir () 创建一个目录

实验使用如下：

```
mkdir(argv[2], statbuf.st_mode);
```

(3) stat () 获取文件相关信息

实验使用如下：

```
stat(argv[1], &statbuf);
```

(4) lstat () 获取文件相关信息，但当文件为软链接时，会获取链接文件本身的信息

实验使用如下：

```
struct stat statbuf;
lstat(sourcefile, &statbuf);
```

(5) readdir () 读取 opendir 打开的目录的内容

实验使用如下：

```
for (; (entry = readdir(dir)) != NULL;) //遍历所有文件
{
```

(6) utime () 修改文件的时间属性

实验使用如下：

```
struct utimbuf timebuf;
timebuf.actime = statbuf.st_atime;
timebuf.modtime = statbuf.st_mtime;
utime(object, &timebuf);
```

(7) lutimes () 修改软链接文件的时间属性

实验使用如下：

```
struct timeval timebuf[2]; //utimes需要使用timeval数组修改时间
timebuf[0].tv_sec = statbuf.st_atime;
timebuf[0].tv_usec = 0;
timebuf[1].tv_sec = statbuf.st_mtime;
timebuf[1].tv_usec = 0;
lutimes(objectfile, timebuf);
```

(8) open () 打开一个文件

实验使用如下：

```
int source = open(sourcefile, 0); //打开文件
```

(9) creat () 创建一个文件，若该文件已存在，则会将其长度截断为 0

实验使用如下：

```
int object = creat(objectfile, statbuf.st_mode); //创建一个文件，若该文件已存在，则会将其长度截为零
```

(10) read () 从打开的文件中读取数据

实验使用如下：

```
for (; (temp = read(source, buffer, 2000)) > 0;)
```

(11) write () 向打开的文件中写入数据

实验使用如下：

```
if (write(object, buffer, temp) != temp) //判断是否写入完全
{
    printf("write error!\n");
    exit(0);
}
```

(12) readlink () 读取软链接文件本身的内容存储到指定位置

实验使用如下：

```
char buffer[2000];
readlink(sourcefile, buffer, 2000); //找到软链接的目标
symlink(buffer, objectfile); //创建新的软链接
```

(13) symlink () 创建软链接文件

实验使用如下：

```
char buffer[2000];
readlink(sourcefile, buffer, 2000); //找到软链接的目标
symlink(buffer, objectfile); //创建新的软链接
```

5、Linux 下实验结果

复制文件

```
zhangwei@zhangwei: ~/文档/实验五
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
zhangwei@zhangwei:~$ cd 文档/实验五
zhangwei@zhangwei:~/文档/实验五$ l
mycp* mycp.c test/ test2/
zhangwei@zhangwei:~/文档/实验五$ ls
mycp mycp.c test test2
zhangwei@zhangwei:~/文档/实验五$ rm -rf test2
zhangwei@zhangwei:~/文档/实验五$ ls
mycp mycp.c test
zhangwei@zhangwei:~/文档/实验五$ ./mycp test test2
mkdir succeed!
copy completed!
zhangwei@zhangwei:~/文档/实验五$
```

检验复制结果

```
zhangwei@zhangwei: ~/文档/实验五
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
zhangwei@zhangwei:~/文档/实验五$ ls
mycp mycp.c test
zhangwei@zhangwei:~/文档/实验五$ ./mycp test test2
mkdir succeed!
copy completed!
zhangwei@zhangwei:~/文档/实验五$ ls -al test
总用量 20
drwxr-xr-x 3 zhangwei zhangwei 4096 12月 18 16:00 .
drwxr-xr-x 6 zhangwei zhangwei 4096 12月 18 19:52 ..
-rw-rw-rw- 1 zhangwei zhangwei 184 12月 16 15:29 a.txt
-rw-rw-rw- 1 zhangwei zhangwei 182 12月 16 15:27 b.docx
drwxr-xr-x 2 zhangwei zhangwei 4096 12月 16 15:29 sub
lrwxrwxrwx 1 zhangwei zhangwei 40 12月 18 16:00 sublink -> /home/zhangwei/文档/实验五/test/sub
zhangwei@zhangwei:~/文档/实验五$ ls -al test2
总用量 20
drwxr-xr-x 3 zhangwei zhangwei 4096 12月 18 16:00 .
drwxr-xr-x 6 zhangwei zhangwei 4096 12月 18 19:52 ..
-rw-r--r-- 1 zhangwei zhangwei 184 12月 16 15:29 a.txt
-rw-r--r-- 1 zhangwei zhangwei 182 12月 16 15:27 b.docx
drwxr-xr-x 2 zhangwei zhangwei 4096 12月 16 15:29 sub
lrwxrwxrwx 1 zhangwei zhangwei 40 12月 18 16:00 sublink -> /home/zhangwei/文档/实验五/test/sub
zhangwei@zhangwei:~/文档/实验五$
```

子目录下的实验结果


```
zhangwei@zhangwei: ~/文档/实验五
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
-rw-rw-rw- 1 zhangwei zhangwei 182 12月 16 15:27 b.docx
drwxr-xr-x 2 zhangwei zhangwei 4096 12月 16 15:29 sub
lrwxrwxrwx 1 zhangwei zhangwei 40 12月 18 16:00 sublink -> /home/zhangwei/文档
/实验五/test/sub
zhangwei@zhangwei:~/文档/实验五$ ls -al test2
总用量 20
drwxr-xr-x 3 zhangwei zhangwei 4096 12月 18 16:00 .
drwxr-xr-x 6 zhangwei zhangwei 4096 12月 18 19:52 ..
-rw-r--r-- 1 zhangwei zhangwei 184 12月 16 15:29 a.txt
-rw-r--r-- 1 zhangwei zhangwei 182 12月 16 15:27 b.docx
drwxr-xr-x 2 zhangwei zhangwei 4096 12月 16 15:29 sub
lrwxrwxrwx 1 zhangwei zhangwei 40 12月 18 16:00 sublink -> /home/zhangwei/文档
/实验五/test/sub
zhangwei@zhangwei:~/文档/实验五$ ls -al test/sub
总用量 8
drwxr-xr-x 2 zhangwei zhangwei 4096 12月 16 15:29 .
drwxr-xr-x 3 zhangwei zhangwei 4096 12月 18 16:00 ..
lrwxrwxrwx 1 zhangwei zhangwei 8 12月 16 15:29 softlink -> ../a.txt
zhangwei@zhangwei:~/文档/实验五$ ls -al test2/sub
总用量 8
drwxr-xr-x 2 zhangwei zhangwei 4096 12月 16 15:29 .
drwxr-xr-x 3 zhangwei zhangwei 4096 12月 18 16:00 ..
lrwxrwxrwx 1 zhangwei zhangwei 8 12月 16 15:29 softlink -> ../a.txt
zhangwei@zhangwei:~/文档/实验五$
```

五、 实验收获与体会

通过本次实验，自己又学到了许多以前没有用过的系统 API，对系统的文件系统有了一定的了解，同时也接触了许多系统预定义的结构体，如 Windows 中的 `struct _WIN32_FIND_DATA`，`FILETIME` 结构体以及 Linux 中的 `struct stat`，`dirent`，`utimebuf`，`timeval` 结构体，这让我们获得系统或者文件的相关信息变的十分便利。借此，我成功完成了文件的遍历和复制操作。

实验中有需要注意的一点是对目录进行复制时要先对其进行复制操作，然后再对复制后的目录文件进行修改以使其时间信息与源文件一致，否则在该目录下进行复制粘贴操作时系统会自动更新其时间。

另外，Linux 实验中 `utime` 系统调用也令我倍感棘手。其实大部分的修改文件时间操作都可以使用 `utime` 这一系统调用来实现，然而对于符号链接文件来说，则需要使用 `lutimes` 才能完成。之前用 `utime` 来做，符号连接文件的文件时间老是为系统时间。后来在 Linux 官方论坛上找到了 `lutimes` 这一 API 才把问题解决。

还有，在 Linux 下通过 `readlink` 读取原有的软链接文件的内容，然后通过 `symlink` 创建新的软链接文件的时候，若是原软链接文件存储的绝对路径还好说，若是存储的相对路径则在复制到非平级目录下之后新的软链接就会失效，后来经过老师和同学的提醒之后对代码进行了修改，在复制软链接文件时先读取原软链

接文件的绝对路径，然后将其写入新创建的软链接文件中即可避免此问题。

在 **Windows** 下实现和在 **Linux** 下实现的大致原理是一样的，但是也要注意其中不一样的地方，把握住这些不一样的地方，对这些 **API** 和结构体的理解就会更加的深刻了。