

操作系统课程设计实验报告 实验四

实验名称： 复制文件

- 班级： 07112005
- 姓名： 董若扬
- 学号： 1120202944

一、实验目的

- 熟悉Linux文件系统提供的有关文件操作的系统调用。文件系统是使用计算机信息系统的重要接口，通过使用文件系统的系统调用命令操作文件，以达到对文件系统实现功能的理解和掌握。
- 了解Windows的文件系统时如何管理保存在磁盘、光盘等存储介质上的信息。并通过文件系统提供的各种API，对文件进行操作，深入理解Windows文件系统的功能和作用，理解文件系统的系统调用的功能。

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：

```
./gitclones/osexp/mycp ~/test/ ~/sametest

ls -Rla test/ sametest/
sametest/:
总用量 16
drwxr-xr-x  3 unv unv 4096 12月 16 09:01 ./
drwx----- 24 unv unv 4096 12月 18 16:02 ../
-rw-r--r--  1 unv unv   20 12月 16 09:00 example
drwxr-xr-x  2 unv unv 4096 12月 16 09:02 exp_folder/
lrwxrwxrwx  1 unv unv   35 12月 16 09:01 proxychains.conf -> /home/unv/dotfiles/proxychains.conf

sametest/exp_folder:
总用量 16
drwxr-xr-x  2 unv unv 4096 12月 16 09:02 ./
drwxr-xr-x  3 unv unv 4096 12月 16 09:01 ../
-rw-r--r--  1 unv unv   35 12月 16 09:02 exp2
-rw-r--r--  1 unv unv    9 12月 16 09:02 exp3

test/:
总用量 16
drwxr-xr-x  3 unv unv 4096 12月 16 09:01 ./
drwx----- 24 unv unv 4096 12月 18 16:02 ../
-rw-r--r--  1 unv unv   20 12月 16 09:00 example
drwxr-xr-x  2 unv unv 4096 12月 16 09:02 exp_folder/
lrwxrwxrwx  1 unv unv   35 12月 16 09:01 proxychains.conf -> /home/unv/dotfiles/proxychains.conf

test/exp_folder:
总用量 16
drwxr-xr-x  2 unv unv 4096 12月 16 09:02 ./
drwxr-xr-x  3 unv unv 4096 12月 16 09:01 ../
-rw-r--r--  1 unv unv   35 12月 16 09:02 exp2
-rw-r--r--  1 unv unv    9 12月 16 09:02 exp3
```

说明：

- Linux: create, 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。

三、实验环境

Windows: 版本 Windows 11 家庭中文版; 版本 22H2; 安装日期 2022/5/14; 操作系统版本 22623.746; 体验 Windows Feature Experience Pack 1000.22636.1000.0.

Linux : Ubuntu 18.04.5 LTS

四、程序设计与实现

设计思路

对于给定的文件，首先检查它是否是目录文件。如果是目录文件，首先复制目录文件的权限和其他信息，然后调用编写的 `MyCp()` 函数递归复制文件夹的内容。`MyCp()` 函数的基本思想是调用 `CopyFile()` 函数，将文件内容、权限、时间和其他信息复制到指定的文件路径（如果所查询的文件是普通文件）；如果当前查询的文件是软链接文件，则调用 `CopyLinkFile()` 函数将软连接文件复制到指定的文件路径；如果查询的文件是目录文件，则递归调用 `MyCp()` 函数来处理目录文件，直到该文件不再是目录文件。对于给定的文件，如果它是非目录文件，程序将直接调用 `MyCp()` 函数将文件复制到指定的路径。由于时间信息是在创建目录文件或其他文件时设置的，因此后续访问将导致修改文件的时间信息。因此，每个目录文件的时间属性只能在复制其所有子文件后进行修改。最后，关闭打开的文件或打开的文件句柄。

Windows实现

1.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]` : 文件名
- TCHAR `cAlternateFileName[14]` : 格式文件名

```
1 typedef struct _WIN32_FIND_DATA {
2     DWORD dwFileAttributes;
3     FILETIME ftCreationTime;
4     FILETIME ftLastAccessTime;
5     FILETIME ftLastWriteTime;
6     DWORD nFileSizeHigh;
7     DWORD nFileSizeLow;
8     DWORD dwReserved0;
9     DWORD dwReserved1;
10    TCHAR cFileName[ MAX_PATH ];
11    TCHAR cAlternateFileName[ 14 ];
12 } WIN32_FIND_DATA, *PWIN32_FIND_DATA;
```

2.FindFirstFile()

查找指定路径的文件，成功则返回文件或目录句柄。

- LPCTSTR `lpFileName`：目录名
- LPWIN32_FIND_DATA `lpFindFileData`：文件信息结构体

```
1 HANDLE FindFirstFile(
2     LPCTSTR lpFileName,
3     LPWIN32_FIND_DATA lpFindFileData
4 );
```

3.FindNextFile()

查找 `FindFirstFile` 函数搜索后的下一个文件，成功则返回非0，否则返回0。

- HANDLE `hFindFile`：调用 `FindFirstFile` 返回的句柄
- LPWIN32_FIND_DATA `lpFindFileData`：文件信息结构体

```
1 FindNextFile(
2     HANDLE hFindFile,
3     LPWIN32_FIND_DATA lpFindFileData
4 );
```

4.CreateDirectory()

创建文件夹()目录文件

- LPCTSTR `lpPathName`：文件路径
- LPSECURITY_ATTRIBUTES `lpSecurityAttributes`：常为NULL

```
1 BOOL CreateDirectory(
2     LPCTSTR lpPathName,
3     LPSECURITY_ATTRIBUTES lpSecurityAttributes
4 );
```

5.CreateFile()

打开或创建文件，返回该文件的句柄

- 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`：安全属性
- DWORD `dwCreationDisposition`：创建方式
- DWORD `dwFlagsAndAttributes`：文件属性
- HANDLE `hTemplateFile`：用于复制文件句柄

```

1  HANDLE CreateFile(
2      LPCTSTR lpFileName,
3      DWORD dwDesiredAccess,
4      DWORD dwShareMode,
5      LPSECURITY_ATTRIBUTES lpSecurityAttributes,
6      DWORD dwCreationDisposition,
7      DWORD dwFlagsAndAttributes,
8      HANDLE hTemplateFile
9  );

```

6.SetFileTime()

设置文件的时间属性，创建时间、最后一次修改和访问时间

- Long `hFile`：文件句柄
- FILETIME `lpCreationTime`：文件创建时间
- FILETIME `lpLastAccessTime`：文件最后一次访问时间
- FILETIME `lpLastWriteTime`：文件最后一次修改时间

```

1  Long SetFileTime(
2      Long hFile,
3      FILETIME lpCreationTime,
4      FILETIME lpLastAccessTime,
5      FILETIME lpLastWriteTime
6  );

```

7.SetFileAttributes()

设置文件属性

- LPCTSTR `lpFileName`：文件路径
- DWORD `dwFileAttributes`：文件属性

```

1  BOOL SetFileAttributes(
2      LPCTSTR lpFileName,
3      DWORD dwFileAttributes
4  );

```

8.ReadFile()

从指定路径的文件中读取指定字节长度的数据

- 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`：实际读取字节数的指针
- LPOVERLAPPED `lpOverlapped`：一般置为NULL

```

1  BOOL ReadFile(
2      HANDLE hFile,
3      LPVOID lpBuffer,
4      DWORD nNumberOfBytesToRead,
5      LPDWORD lpNumberOfBytesRead,
6      LPOVERLAPPED lpOverlapped
7  );

```

9. WriteFile()

从文件指针指向的位置开始将数据写入到一个文件中

- HANDLE `hFile` : 文件句柄
- LPCVOID `lpBuffer` : 数据缓存区指针
- DWORD `nNumberOfBytesToWrite` : 写入字节数
- LPDWORD `lpNumberOfBytesWritten` : 指向实际写入字节数
- LPOVERLAPPED `lpOverlapped` : 结构体指针, 一般为NULL

```

1  BOOL WriteFile(
2      HANDLE hFile,
3      LPCVOID lpBuffer,
4      DWORD nNumberOfBytesToWrite,
5      LPDWORD lpNumberOfBytesWritten,
6      LPOVERLAPPED lpOverlapped
7  );

```

Linux实现

1. stat

- dev_t `st_dev` : 文件所在设备的ID
- ino_t `st_ino` : 节点号
- mode_t `st_mode` : 保护模式
- nlink_t `st_nlink` : 链向此文件的连接数(硬连接)
- uid_t `st_uid` : user id
- gid_t `st_gid` : group id
- dev_t `st_rdev` : 设备号, 针对设备文件
- off_t `st_size` : 文件大小, 字节为单位
- blksize_t `st_blksize` : 系统块的大小
- blkcnt_t `st_blocks` : 文件所占块数
- time_t `st_atime` : 最近存取时间
- time_t `st_mtime` : 最近修改时间
- time_t `st_ctime` : 最后修改时间

```

1  struct stat {
2      dev_t    st_dev;
3      ino_t    st_ino;
4      mode_t   st_mode;
5      nlink_t  st_nlink;
6      uid_t    st_uid;
7      gid_t    st_gid;
8      dev_t    st_rdev;

```

```

9     off_t  st_size;
10    blksize_t  st_blksize;
11    blkcnt_t  st_blocks;
12    time_t  st_atime;
13    time_t  st_mtime;
14    time_t  st_ctime;
15    };

```

2.utimbuf

- time_t `atime`: 访问时间
- time_t `modtime`: 更改时间

```

1 struct utimbuf{
2     time_t  atime;
3     time_t  modtime;
4 };

```

3.dirent

- long `d_ino`: 索引节点号
- off_t `d_off`: 在目录文件中的偏移
- unsigned short `d_reclen`: 文件名长
- unsigned char `d_type`: 文件类型
- char `d_name [NAME_MAX+1]`: 文件名, 最长255字符

```

1 struct dirent{
2     long d_ino;
3     off_t d_off;
4     unsigned short d_reclen;
5     unsigned char d_type;
6     char d_name [NAME_MAX+1];
7 };

```

其他函数

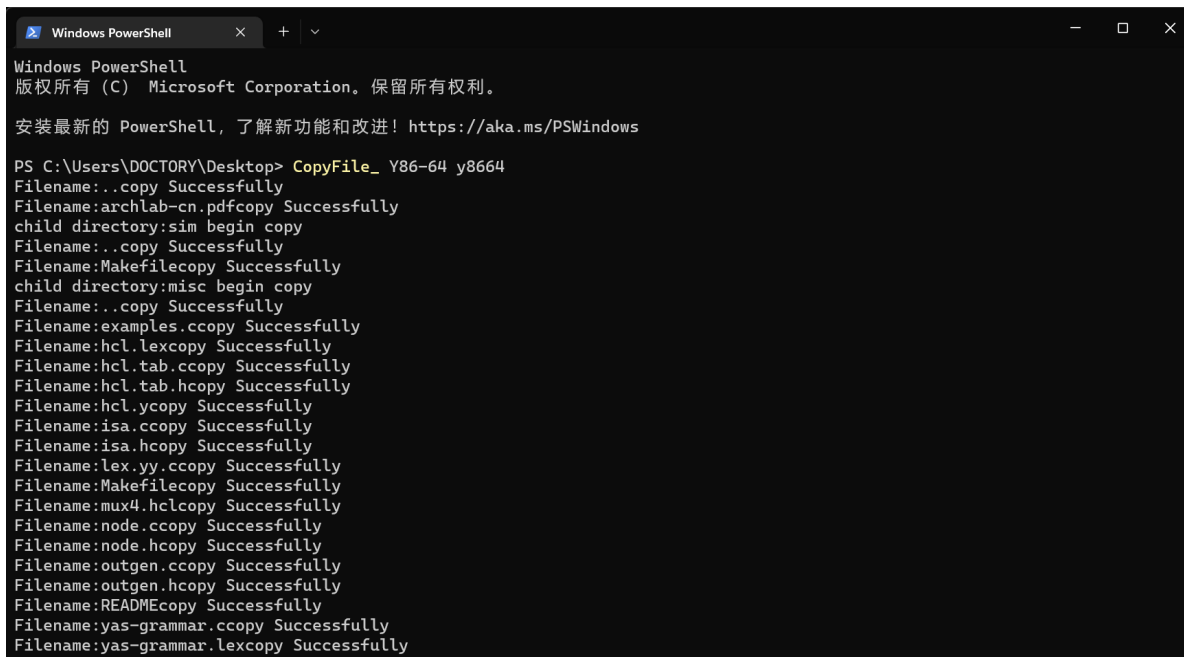
- DIR * opendir(const char * name);
 - 功能: 打开指定的目录, 并返回DIR*型的目录流。
- int mkdir (const char *filename, mode_t mode)
 - 功能: 创建名为filename的目录, 创建成功返回0, 失败返回-1。*
- int stat(const char *path, struct stat *buf)
 - 参数: 文件路径名, stat类型结构体。
 - 功能: 获取path路径下的文件属性信息, 成功返回0, 失败返回-1。
- int utime(const char *filename, const struct utimbuf *times)
 - 参数: 文件路径名, utimbuf类型结构体。
 - 功能: 修改文件的时间属性, 包括最后一词访问和修改时间, 成功则返回0, 失败则返回-1。
- int closedir(DIR* dir)
 - 功能: 关闭目录文件指针。

- `struct dirent* readdir(DIR* dir_handle);`
 - 功能：读取下一个目录，返回DIR* 类型的指针。
- `S_ISDIR()`
 - 功能：判断一个文件是否为目录文件。
- `S_ISLNK()`
 - 功能：判断一个文件是否为符号链接文件。
- `int lstat(const char *path, struct stat *buf);`
 - 功能：与stat()相似，只是path下的文件是符号链接文件时，lstat()函数获取该符号链接文件的信息。
- `int readlink(const char *path, char *buf, size_t bufsize)`
 - 功能：读取符号链接文件本身的信息，组合了打开、读和关闭的所有操作。成功则返回读到缓冲区buf的字节数，否则返回-1。
- `int symlink(const char *oldpath, const char *sympath)`
 - 参数：已存在的文件路径名，将要创建的符号链接文路径名。
 - 功能：创建一个符号链接文件。成功时返回0，否则返回-1。
- `int chmod(const char filename, int pmode)`
 - 参数：文件路径名，文件读取许可权限。
 - 功能：改变文件的读写许可权限，成功改变则返回0，否则返回-1。
- `lutimes(const char *filename, struct timeval *tv)`
 - 参数：符号链接文件路径名，timeval结构体
 - 功能：改变符号链接文件的时间属性，成功返回0，否则返回-1。

实验结果

Windows

开始复制



```

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

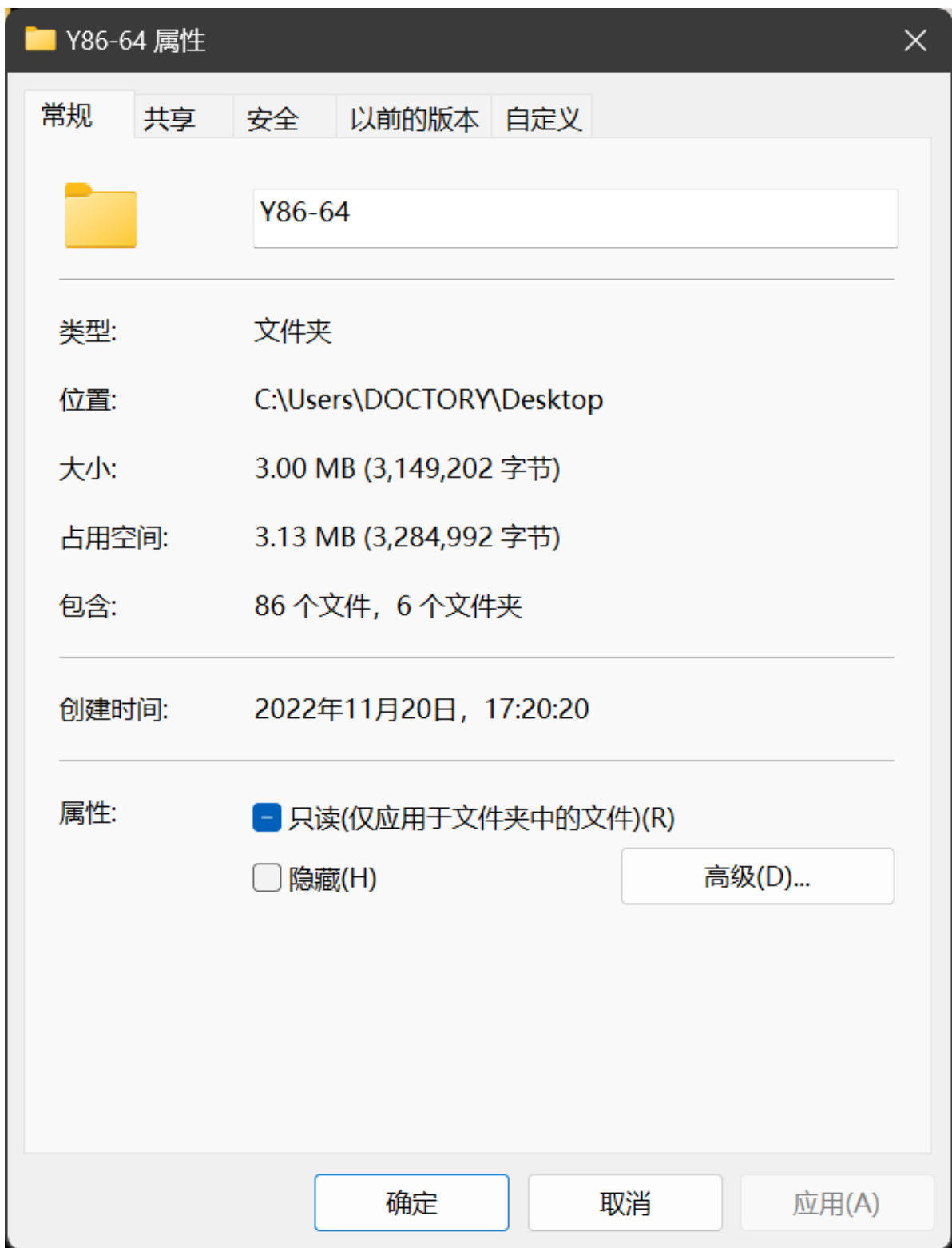
安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS C:\Users\DOCTORY\Desktop> CopyFile_ Y86-64 y8664
Filename:..copy Successfully
Filename:archlab-cn.pdfcopy Successfully
child directory:sim begin copy
Filename:..copy Successfully
Filename:Makefilecopy Successfully
child directory:misc begin copy
Filename:..copy Successfully
Filename:examples.ccopy Successfully
Filename:hcl.lexcopy Successfully
Filename:hcl.tab.ccopy Successfully
Filename:hcl.tab.hcopy Successfully
Filename:hcl.ycopy Successfully
Filename:isa.ccopy Successfully
Filename:isa.hcopy Successfully
Filename:lex.yy.ccopy Successfully
Filename:Makefilecopy Successfully
Filename:mux4.hclcopy Successfully
Filename:node.ccopy Successfully
Filename:node.hcopy Successfully
Filename:outgen.ccopy Successfully
Filename:outgen.hcopy Successfully
Filename:READMEcopy Successfully
Filename:yas-grammar.ccopy Successfully
Filename:yas-grammar.lexcopy Successfully
  
```

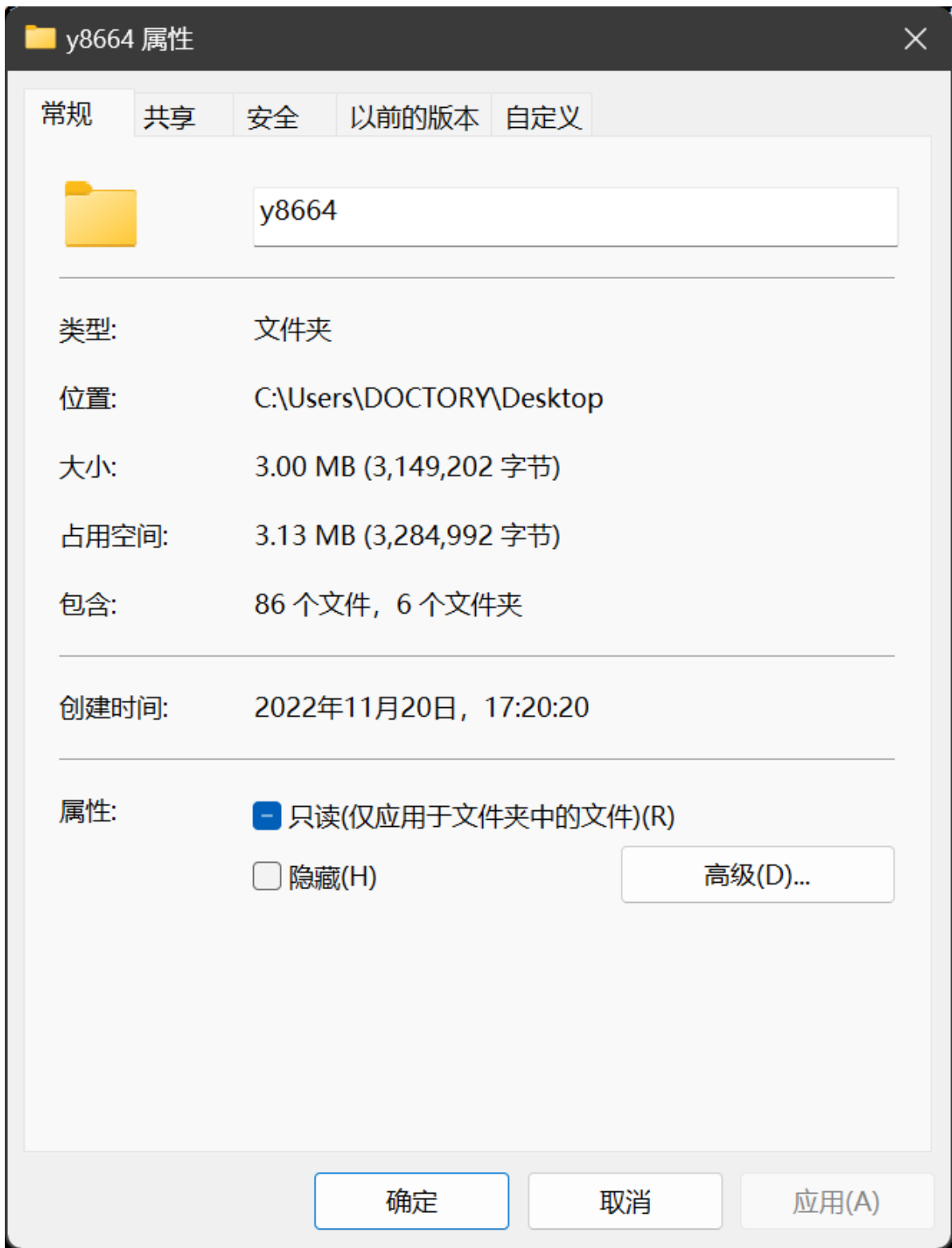
复制完成

```
Windows PowerShell
Filename:..copy Successfully
Filename:abs-asum-cmov.yscopy Successfully
Filename:abs-asum-jmp.yscopy Successfully
Filename:asum.yscopy Successfully
Filename:asumi.yscopy Successfully
Filename:asumr.yscopy Successfully
Filename:cjr.yscopy Successfully
Filename:j-cc.yscopy Successfully
Filename:Makefilecopy Successfully
Filename:poptest.yscopy Successfully
Filename:prog1.yscopy Successfully
Filename:prog10.yscopy Successfully
Filename:prog2.yscopy Successfully
Filename:prog3.yscopy Successfully
Filename:prog4.yscopy Successfully
Filename:prog5.yscopy Successfully
Filename:prog6.yscopy Successfully
Filename:prog7.yscopy Successfully
Filename:prog8.yscopy Successfully
Filename:prog9.yscopy Successfully
Filename:pushquestion.yscopy Successfully
Filename:pushtest.yscopy Successfully
Filename:READMEcopy Successfully
Filename:ret-hazard.yscopy Successfully
child directory:y86-code copy Successfully
child directory:sim copy Successfully
Filename:sim.tarcopy Successfully
Filename:simguide-cn.pdfcopy Successfully
Copy Successfully!
PS C:\Users\DOCTORY\Desktop>
```

原文件



新文件



Linux

开始复制

```

dry@ubuntu:~$ ./mycp sim sim1
Directory: sim/y86-code, 正在拷贝...
FileName: sim/y86-code/prog5.yo, 正在拷贝...
FileName: sim/y86-code/prog5.yo, 拷贝完成...
FileName: sim/y86-code/prog7.yo, 正在拷贝...
FileName: sim/y86-code/prog7.yo, 拷贝完成...
FileName: sim/y86-code/prog9.yo, 正在拷贝...
FileName: sim/y86-code/prog9.yo, 拷贝完成...
FileName: sim/y86-code/cjr.yo, 正在拷贝...
FileName: sim/y86-code/cjr.yo, 拷贝完成...
FileName: sim/y86-code/ret-hazard.yo, 正在拷贝...
FileName: sim/y86-code/ret-hazard.yo, 拷贝完成...
FileName: sim/y86-code/asumt.yo, 正在拷贝...
FileName: sim/y86-code/asumt.yo, 拷贝完成...
FileName: sim/y86-code/prog10.yo, 正在拷贝...
FileName: sim/y86-code/prog10.yo, 拷贝完成...
FileName: sim/y86-code/cjr.yo, 正在拷贝...
FileName: sim/y86-code/cjr.yo, 拷贝完成...
FileName: sim/y86-code/abs-asum-cmov.yo, 正在拷贝...
FileName: sim/y86-code/abs-asum-cmov.yo, 拷贝完成...
FileName: sim/y86-code/asumt.yo, 正在拷贝...
FileName: sim/y86-code/asumt.yo, 拷贝完成...
FileName: sim/y86-code/prog7.yo, 正在拷贝...
FileName: sim/y86-code/prog7.yo, 拷贝完成...
FileName: sim/y86-code/asumr.yo, 正在拷贝...
FileName: sim/y86-code/asumr.yo, 拷贝完成...
FileName: sim/y86-code/abs-asum-cmov.yo, 正在拷贝...
FileName: sim/y86-code/abs-asum-cmov.yo, 拷贝完成...
FileName: sim/y86-code/prog8.yo, 正在拷贝...
FileName: sim/y86-code/prog8.yo, 拷贝完成...
FileName: sim/y86-code/prog3.yo, 正在拷贝...
FileName: sim/y86-code/prog3.yo, 拷贝完成...
FileName: sim/y86-code/abs-asum-jmp.yo, 正在拷贝...
FileName: sim/y86-code/abs-asum-jmp.yo, 拷贝完成...
FileName: sim/y86-code/prog4.yo, 正在拷贝...
FileName: sim/y86-code/prog4.yo, 拷贝完成...
FileName: sim/y86-code/ret-hazard.yo, 正在拷贝...
FileName: sim/y86-code/ret-hazard.yo, 拷贝完成...
FileName: sim/y86-code/prog1.yo, 正在拷贝...
FileName: sim/y86-code/prog1.yo, 拷贝完成...
FileName: sim/y86-code/prog1.yo, 正在拷贝...
FileName: sim/y86-code/prog1.yo, 拷贝完成...
FileName: sim/y86-code/prog2.yo, 正在拷贝...
FileName: sim/y86-code/prog2.yo, 拷贝完成...
FileName: sim/y86-code/prog9.yo, 正在拷贝...
FileName: sim/y86-code/prog9.yo, 拷贝完成...
FileName: sim/y86-code/prog3.yo, 正在拷贝...
FileName: sim/y86-code/prog3.yo, 拷贝完成...
FileName: sim/y86-code/progtest.yo, 正在拷贝...

```

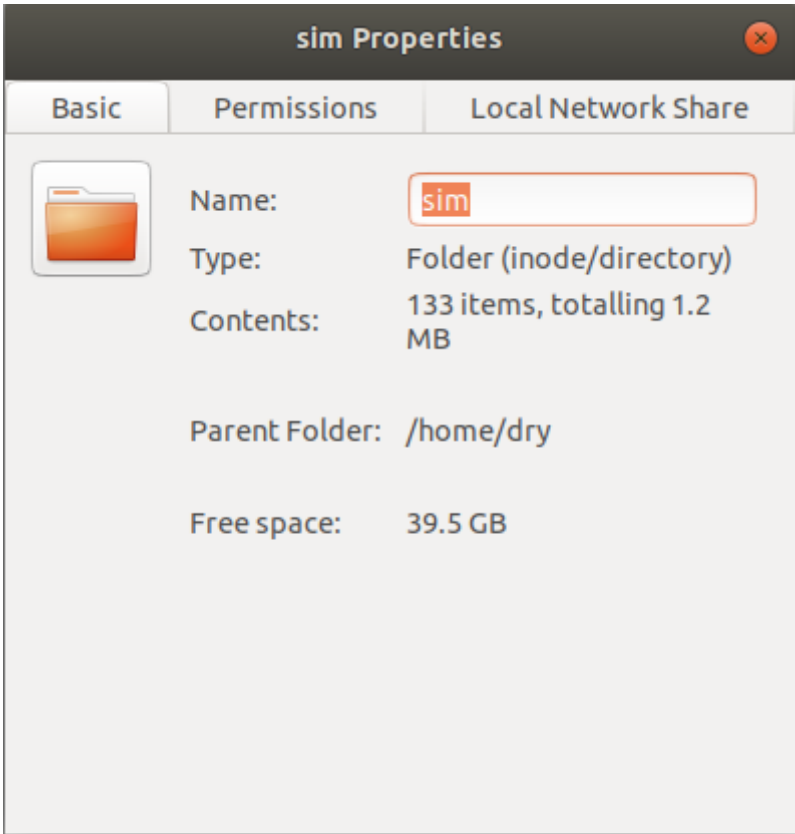
完成复制

```

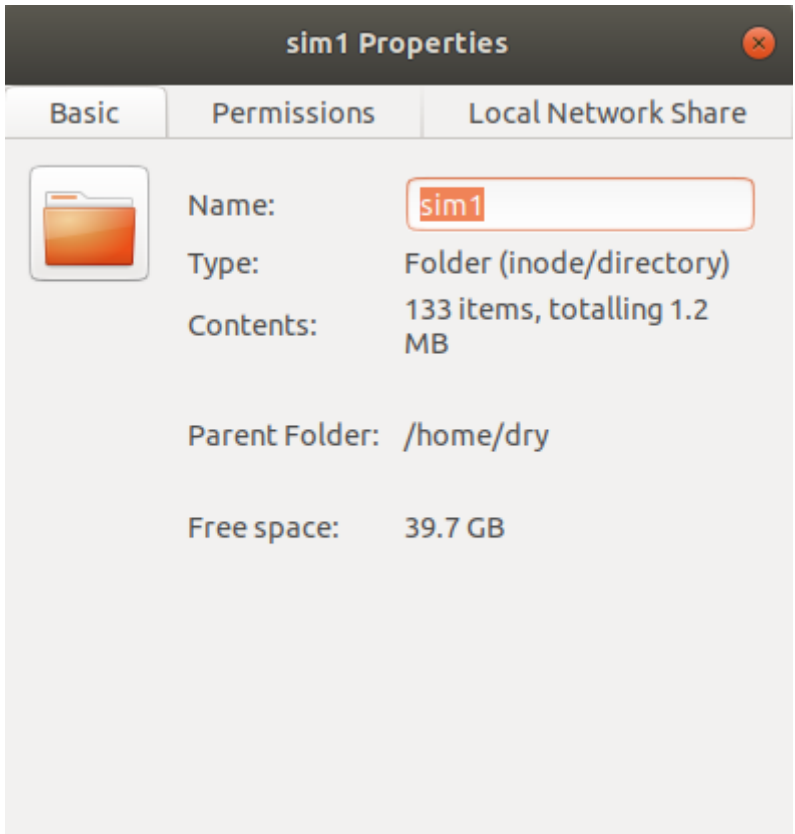
dry@ubuntu:~$ ./mycp misc misc1
Directory: misc, 正在拷贝...
FileName: misc/isa.o, 正在拷贝...
FileName: misc/isa.o, 拷贝完成...
FileName: misc/node.c, 正在拷贝...
FileName: misc/node.c, 拷贝完成...
FileName: misc/hcl2c, 正在拷贝...
FileName: misc/hcl2c, 拷贝完成...
FileName: misc/yis.c, 正在拷贝...
FileName: misc/yis.c, 拷贝完成...
FileName: misc/hcl.lex, 正在拷贝...
FileName: misc/hcl.lex, 拷贝完成...
FileName: misc/lex.yy.c, 正在拷贝...
FileName: misc/lex.yy.c, 拷贝完成...
FileName: misc/yas-grammar.lex, 正在拷贝...
FileName: misc/yas-grammar.lex, 拷贝完成...
FileName: misc/yas-grammar.o, 正在拷贝...
FileName: misc/yas-grammar.o, 拷贝完成...
FileName: misc/mux4.hcl, 正在拷贝...
FileName: misc/mux4.hcl, 拷贝完成...
FileName: misc/outgen.h, 正在拷贝...
FileName: misc/outgen.h, 拷贝完成...
FileName: misc/hcl.tab.h, 正在拷贝...
FileName: misc/hcl.tab.h, 拷贝完成...
FileName: misc/isa.c, 正在拷贝...
FileName: misc/isa.c, 拷贝完成...
FileName: misc/node.h, 正在拷贝...
FileName: misc/node.h, 拷贝完成...
FileName: misc/hcl.tab.c, 正在拷贝...
FileName: misc/hcl.tab.c, 拷贝完成...
FileName: misc/README, 正在拷贝...
FileName: misc/README, 拷贝完成...
FileName: misc/rsun.yo, 正在拷贝...
FileName: misc/rsun.yo, 拷贝完成...
FileName: misc/yis.o, 正在拷贝...
FileName: misc/yis.o, 拷贝完成...
FileName: misc/sum.yo, 正在拷贝...
FileName: misc/sum.yo, 拷贝完成...
FileName: misc/Makefile, 正在拷贝...
FileName: misc/Makefile, 拷贝完成...
FileName: misc/yis, 正在拷贝...
FileName: misc/yis, 拷贝完成...
FileName: misc/yas.h, 正在拷贝...
FileName: misc/yas.h, 拷贝完成...
FileName: misc/yas.c, 正在拷贝...
FileName: misc/yas.c, 拷贝完成...
FileName: misc/examples.c, 正在拷贝...
FileName: misc/examples.c, 拷贝完成...
Directory: misc, 拷贝完成
copy finished!
dry@ubuntu:~$

```

原文件



新文件



五、实验收获与体会

使用 `readlink()` 函数时，不需要打开或关闭文件，因为 `readlink()` 系统调用将完成打开、读取和关闭文件等一系列操作。`readlink()` 系统调用仅在读取链接文件本身的信息时使用。

在Windows上实现时，在许多情况下，很容易忘记修改新文件的权限,应调用 `SetFileAttributes()` 来完成文件保存权限的设置。

通过实验，我掌握了在Linux和Windows中使用系统API调用来操作文件和目录，并了解了其原理。了解并掌握文件系统的实现功能。在实践操作中加强了对课程知识的理解。