操作系统课程设计实验报告

实验名称	复制文件		
学号	1120141831	姓名	朴泉宇

一、实验目的

在 Windows 以及 Linux 平台下实现复制目录（包含子目录及目录下的文件）的功能。

二、实验内容

1. 在 Windows、Linux 下编写代码
2. 编译运行，查看复制结果

三、实验环境及配置方法

1. 利用 Visual Studio 编写 Windows 下的代码，并编译运行
2. 利用 Notepad++ 编写 Linux 下的代码，并编译运行

四、实验方法和实验步骤（程序设计与实现）

1. 编写 Windows 下的代码

(1) 思路

分别获得源目录（argv[1]）的句柄，目标目录（argv[2]）的句柄，若 argv[2]指向的目录不存在，则创建此目录。当然，对于子目录也会进行创建。

之后以文件夹为递归单位进行递归复制。

(2) 函数解释

1) main 函数主控

```
1.      WIN32_FIND_DATA FindSourceFileData;
2.      WIN32_FIND_DATA FindTargetFileData;
3.
4.      if (FindFirstFile(argv[1], &FindSourceFileData) == INVALID_HANDLE_VALUE)
5.      {
6.          printf("Source Folder Address Error!\n");
7.
8.          return 1;
9.      }
```

```
    else
    {
        if (FindFirstFile(argv[2], &FindTargetFileData) == INVALID_HANDLE_VALUE)
        {
            BOOL mycreateFlag = myCreateDirectory(argv[1], argv[2]);
            if (mycreateFlag == FALSE)
            {
                printf("myCreateDirectory Error!\n");

                return 1;
            }
        }

        BOOL mycopyFlag = myCopy(argv[1], argv[2]);
        if (mycopyFlag == FALSE)
        {
            printf("myCopy Error!\n");

            return 1;
        }
    }
```

2) BOOL myCreateDirectory(TCHAR *source, TCHAR *target);模块，用于复制目录：

```
/*Create the first directory*/
BOOL myCreateDirectory(TCHAR *source, TCHAR *target)
{
    FILETIME CreationTime;
    FILETIME LastAccessTime;
    FILETIME LastWriteTime;

    BOOL createFlag = CreateDirectory(target, NULL);

    BOOL getTimeFlag = myGetFileTime(source, &CreationTime, &LastAccessTime, &LastWriteTime);
    BOOL setTimeFlag = mySetFileTime(target, &CreationTime, &LastAccessTime, &LastWriteTime);

    return (createFlag&getTimeFlag&setTimeFlag);
}
```

3) `BOOL myGetFileTime(TCHAR *Directory, FILETIME *CreationTime, FILETIME *LastAccessTime, FILETIME *LastWriteTime);`

和

`BOOL mySetFileTime(TCHAR *Directory, FILETIME *CreationTime, FILETIME *LastAccessTime, FILETIME *LastWriteTime);`

模块，用于同步目录的创建时间等时间信息：

```
1.  /*Get the file(folder) time information*/
2.  BOOL myGetFileTime(TCHAR *Directory, FILETIME *CreationTime, FILETIME *LastAccessTime, FILETIME *LastWriteTime)
3.  {
4.      HANDLE hDirectory = CreateFile(Directory, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, NULL);  //For directory, must set FILE_FLAG_BACKUP_SEMANTICS
5.
6.      BOOL Flag = GetFileTime(hDirectory, CreationTime, LastAccessTime, LastWriteTime);
7.
8.      CloseHandle(hDirectory);
9.
10.     return Flag;
11. }
12.
13. /*Set the file(folder) time information*/
14. BOOL mySetFileTime(TCHAR *Directory, FILETIME *CreationTime, FILETIME *LastAccessTime, FILETIME *LastWriteTime)
15. {
16.     HANDLE hDirectory = CreateFile(Directory, GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, NULL); //For directory, must set FILE_FLAG_BACKUP_SEMANTICS
17.
18.     BOOL Flag = SetFileTime(hDirectory, CreationTime, LastAccessTime, LastWriteTime);
19.
20.     CloseHandle(hDirectory);
21.
22.     return Flag;
23. }
```

4) `BOOL myCopy(TCHAR *source, TCHAR *target);`模块，是复制的总控函数，进行了复制目录字符串的更改、判断、递归等操作：

```
1.  /*Copy files(folders), recursion*/
2.  BOOL myCopy(TCHAR *source, TCHAR *target)
3.  {
4.      TCHAR fileSource[MAX_PATH];
5.      TCHAR fileTarget[MAX_PATH];
6.
7.      _tcscpy(fileSource, source);
8.      _tcscat(fileSource, TEXT("\\*.*"));
9.      _tcscpy(fileTarget, target);
10.     _tcscat(fileTarget, TEXT("\\"));
11.
12.     WIN32_FIND_DATA FindFileData;
13.     HANDLE hFile = FindFirstFile(fileSource, &FindFileData);
14.     if (hFile == INVALID_HANDLE_VALUE)
15.     {
16.         return FALSE;
17.     }
18.     else
19.     {
20.         while (FindNextFile(hFile, &FindFileData))
21.         {
22.             if (FindFileData.dwFileAttributes == FILE_ATTRIBUTE_DIRECTORY)
23.             {
24.                 /*If the file is child directory*/
25.                 /*This is the Folder Copy segment*/
26.
27.                 if ((_tcscmp(FindFileData.cFileName, TEXT(".")) != 0) && (_tcscmp(
    FindFileData.cFileName, TEXT("..")) != 0))    //If the folder name valid
28.                 {
29.                     memset(fileSource, 0, sizeof(fileSource));
30.                     _tcscpy(fileSource, source);
31.                     _tcscat(fileSource, TEXT("\\"));
32.
33.                     /*Add the folder name to the directory*/
34.                     _tcscat(fileSource, FindFileData.cFileName);
35.                     _tcscat(fileTarget, FindFileData.cFileName);
36.
37.                     /*Create the folder*/
38.                     myCreateDirectory(fileSource, fileTarget);
39.
```

```
40.                     /*Recursion*/
41.                     BOOL myCFlag = myCopy(fileSource, fileTarget);
42.                     if (myCFlag == FALSE)
43.                     {
44.                         printf("myCopy Error!\n");
45.
46.                         return FALSE;
47.                     }
48.
49.                     /*Return to parent directory*/
50.                     memset(fileSource, 0, sizeof(fileSource));
51.                     _tcscpy(fileSource, source);
52.                     _tcscat(fileSource, TEXT("\\"));
53.                     memset(fileTarget, 0, sizeof(fileTarget));
54.                     _tcscpy(fileTarget, target);
55.                     _tcscat(fileTarget, TEXT("\\"));
56.                 }
57.             }
58.             else
59.             {
60.                 /*If the file is the FILE*/
61.                 /*This is the file copy segment*/
62.
63.                 memset(fileSource, 0, sizeof(fileSource));
64.                 _tcscpy(fileSource, source);
65.                 _tcscat(fileSource, TEXT("\\"));
66.
67.                 /*Add the file name to the directory*/
68.                 _tcscat(fileSource, FindFileData.cFileName);
69.                 _tcscat(fileTarget, FindFileData.cFileName);
70.
71.                 BOOL myCFFlag = myCopyFiles(fileSource, fileTarget, FindFileData.n
    FileSizeLow); //Only used for 32bit OS.
72.                 if (myCFFlag == FALSE)
73.                 {
74.                     printf("myCopyFiles Error!\n");
75.
76.                     return FALSE;
77.                 }
78.
79.                 /*Return to directory*/
80.                 memset(fileSource, 0, sizeof(fileSource));
81.                 _tcscpy(fileSource, source);
82.                 _tcscat(fileSource, TEXT("\\"));
```

```
                memset(fileTarget, 0, sizeof(fileTarget));
                _tcscpy(fileTarget, target);
                _tcscat(fileTarget, TEXT("\\"));
            }
        }
    }

    return TRUE;
}
```

5) `BOOL myCopyFiles(TCHAR *source, TCHAR *target, DWORD size);` 子模块，用于实际地复制文件：

```
/*Just copy the file, only used for 32bit OS.*/
BOOL myCopyFiles(TCHAR *source, TCHAR *target, DWORD size)
{
    HANDLE hSourceFile = CreateFile(source, GENERIC_READ, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    HANDLE hTargetFile = CreateFile(target, GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

    DWORD fileSize = size;
    char *Buffer = new char[fileSize + 1];
    DWORD readSize;

    BOOL RFFlag = ReadFile(hSourceFile, Buffer, fileSize, &readSize, NULL);
    BOOL WFFlag = WriteFile(hTargetFile, Buffer, fileSize, &readSize, NULL);

    delete[]Buffer;

    CloseHandle(hTargetFile);
    CloseHandle(hSourceFile);

    return (RFFlag&WFFlag);
}
```

2. 编写 Linux 下的代码

(1) 思路

直接使用 ftw.h（file tree walk）下的 nftw 函数进行目录树的便利，只需要编写在对源目录树进行遍历时，遇到新目录、文件、符号链接文件的相应复制方案即可。

(2) 函数解释

1) main 函数主控

```
1.      SourceDir = argv[1];
2.      TargetDir = argv[2];
3.
4.      nftw(SourceDir.c_str(), myCopyFiles, 512, FTW_PHYS);
```

2) `int myCopyFiles(const char* nowSourceDir, const struct stat * pStat, int typeFlag, struct FTW *pftw);`模块，定义对源目录树进行遍历时，遇到新目录、文件、符号链接文件的相应复制方案：

```
1.  int myCopyFiles(const char* nowSourceDir, const struct stat * pStat, int typeFlag,
     struct FTW *pftw)
2.  {
3.      string nowTargetDir = TargetDir + &nowSourceDir[SourceDir.length()];    //Get
    full target directory
4.
5.      switch(typeFlag)
6.      {
7.          case FTW_D:
8.          {
9.              /*If is a directory*/
10.             mkdir(nowTargetDir.c_str(), pStat->st_mode);
11.             break;
12.         }
13.         case FTW_F:
14.         {
15.             /*If is a normal file*/
16.             ifstream Source(nowSourceDir, ios::in);
17.             ofstream Target(nowTargetDir, ios::out);
18.             Target << Source.rdbuf();
19.             break;
20.         }
```

```
21.         case FTW_SL:
22.         {
23.             /*If is a symbolic link*/
24.             char linkPath[512];
25.             char currentWorkDir[512];
26.
27.             readlink(nowSourceDir, linkPath, sizeof(linkPath)); //Read the link, g
    et the LINK TO path
28.             getcwd(currentWorkDir, sizeof(currentWorkDir)); //Get current work dir
    ectory
29.             chdir(TargetDir.c_str());   //Change work directory to Target root
30.             symlink(linkPath, &nowSourceDir[SourceDir.length() + 1]);   //Create l
    ink file. '+ 1' is because in Terminal, now work directory is changed, needn't '/'
     symbol
31.             chdir(currentWorkDir);  //Return to original work directory
32.             break;
33.         }
34.     }
35.
36.     chmod(nowTargetDir.c_str(), pStat->st_mode);
37.
38.     return 0;
39. }
```

其中，在进行符号链接文件的复制的时候，我们需要改变工作路径：

通过 readlink();函数来读取符号链接文件所链接的源文件路径；

通过 getcwd();来获取当前的工作路径，以恢复现场；

通过 chdir();来改变当前工作路径，到目标目录下；

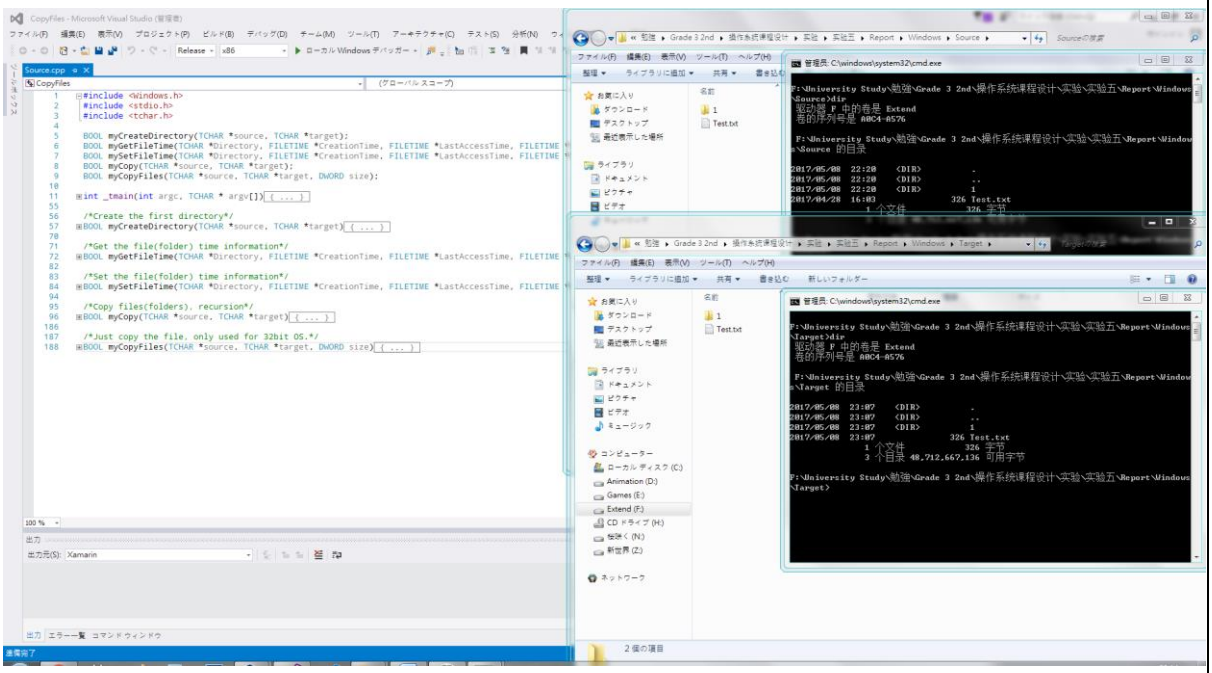
通过 symlink();来创建链接文件

通过 chdir();恢复工作路径现场。

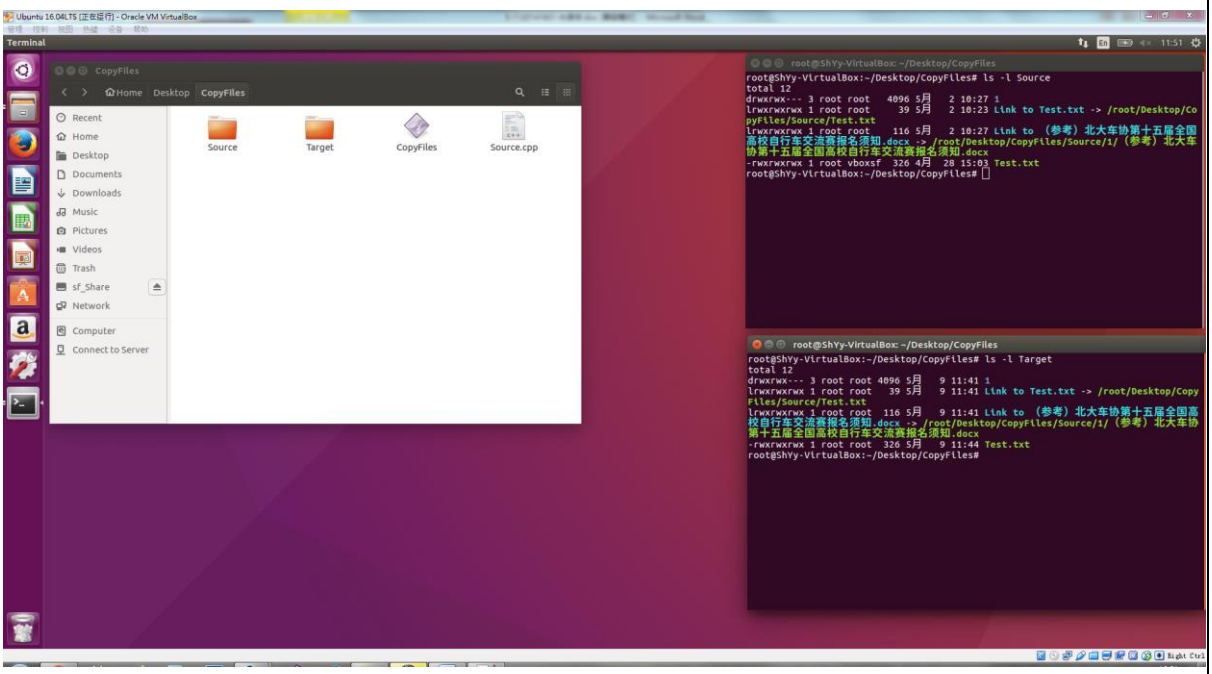
最后的 chmod();函数用于同步源与目标的权限。

五、 实验结果和分析

1. Windows 下的实验结果



2. Linux 下的实验结果



六、 讨论、心得

这次实验，继续学习了 Windows 的相关 API，也学习了 Linux 下的实用的库。Linux 的 ftw.h 库中的 ftw();函数和 nftw();函数很好地解决了遍历目录的需求，而无需像 Windows 一样进行目录字符串的复杂处理。

在 Windows 下进行复制的时候，对返回的 WIN32_FIND_DATA 结构中，DWORD dwFileAttributes 参数的值上会出现一些问题：在 MSDN 文档中给出的参数列表中找不到此参数值（17、8208 等），在网上也没有更多解释资料，在此统一按照“此文件为目录（文件夹）”来处理。

在 Linux 下进行复制的时候，对于 Linux 特殊的符号链接文件需要特别的处理；以及在复制之后，需要同步目标文件或目录的权限。

这次实验，Windows 提醒我们需要考虑一些在文档中没有给出的信息的处理方式，以及 Linux 的库函数的方便让我感受到了开源、共同完善项目的魅力。