操作系统课程设计实验报告

实验名称	内存监视		
学号	1120191600	姓名	张驰

一、实验目的

设计并实现 Windows 和 Linux 下目录复制的命令。学习使用 Linux 和 Windows 下目录读写和文件读写的函数，对 Windows 和 Linux 的文件系统有进一步的理解。

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 -l 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 -l 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/
```

分别在 Windows 和 Linux 平台上做。要求复制后，不仅读写权限一直，时间属性也一致。

三、实验环境及配置方法

Windows10

VMware workstation pro

Ubuntu18

四、实验方法：

1.实验核心思路的描述：

在示例中可以看到，复制文件的命令，包含两个参数，第一个是复制源目录，第二

个是目标目录，复制文件需要将源目录的所有文件（包含目录文件和子文件），修改其中各个文件的属性与源目录中的文件一致。在复制目录文件时，需要复制目录文件下的子目录以及子目录中的文件，对子目录的复制操作和父目录的复制可以采用相同的复制方法。由于 Linux 和 Windows 的文件都是多目录型的树型结构，其中普通文件为文件树的子节点，目录文件为非子节点，所以可以采用 dfs 形式进行递归操作。dfs 递归方法代码结构如下所示：

```
void Filecopy(char* sourcefile , char* targetfile){
    打开源文件，读源文件
    创建目标文件，写目标文件
}

void Dircopy(char* sourcefile , char* targetfile){
    while(找到源目录下的文件){
        if(文件是目录文件){
            创建目录文件
            Dircopy();
        }
        else{
            Filecopy();
        }
    }
}
```

同时，要注意在复制文件的时候，要将复制文件的属性进行修改，修改成与源文件相同的属性。同时，Linux 系统下需要单独判断链接文件，做特殊处理。

四、 实验步骤：

1. Windows 系统下文件复制的实现步骤：

主函数文件中：首先判断参数个数，命令行语句应该是 mycp 源文件 目标文件,所以参数个数应该有 3 个，如果参数个数不等于 3，应该报错。WIN32_FIND_DATA 是文件内容的类型变量,定义变量 filecontent。调用 Windows 查找文件的接口函数 FindFirstFile（）,将源文件的内容放到 filecontent 中。之后查找 filecontent 的 dwFileAttributes，没有找到该目录的话，就先创建一个目录，如果找到了该目录，就调用 Copydir()函数，开始复制。

```
int main(int argc, char* argv[])
{
    if (argc != 3)
    {
```

```
        printf("Error input\n");
        return 0;
    }
    WIN32_FIND_DATA filecontent;
    if (FindFirstFile(argv[1], &filecontent) != INVALID_HANDLE_VALUE)
    {
        if (filecontent.dwFileAttributes == FILE_ATTRIBUTE_DIRECTORY)
        {
            if (FindFirstFile(argv[2], &filecontent) == INVALID_HANDLE_VALUE) {
                //没有找到该目录，就要先创建一个
                CreateDirectory(argv[2], NULL);
            }
            Copydir(argv[1], argv[2]);
        }
    }
    else
    {
        printf("No Source File name: %s", argv[1]);
    }
    return 0;
}
```

Copydir(char* sourcedir, char* targetdir)函数：

同样先调用 FindFirstFile()函数，查找指定目录的第一文件或目录并返回，之后根据查找的第一个文件，同时 FindNextFile()函数，将子文件的属性结构体赋值给 Filecontent，之后复制目标文件的名字，判断是否是目录文件。如果是目录文件，创建新目录，之后调用 Copydir()函数，如果是普通文件，直接调用 Copyfile()函数。再复制完成之后，要更改目标文件的属性，取出源文件的属性，赋值给目标文件。

```
void Copydir(char* sourcedir, char* targetdir)
{
    char fsource[buffer_size], ftarget[buffer_size];//源文件子路径，目标文件子路径
    strcpy(fsource, sourcedir);
    strcat(fsource, "\\*.*");//FindFirstTime 函数所要求的
    WIN32_FIND_DATA Filecontent;//包含所有文件的属性
    HANDLE hfile = FindFirstFile(fsource, &Filecontent);//查找指定目录的第一个文件或目录并返回它的句柄
    if (hfile != INVALID_HANDLE_VALUE)
    {
        while (FindNextFile(hfile, &Filecontent))
        {
            //Filecontent 为当前指向的子文件的属性结构体
            if (strcmp(Filecontent.cFileName, ".") == 0 ||
```

```
strcmp(Filecontent.cFileName, "..") == 0)continue;
            strcpy(fsource, sourcedir);
            strcat(fsource, "\\");
            strcat(fsource, Filecontent.cFileName);

            strcpy(ftarget, targetdir);
            strcat(ftarget, "\\");
            strcat(ftarget, Filecontent.cFileName);
            //如果是目录文件,需要创建新目录调用,递归调用CopyDir,普通文件直接CopyFile
即可

            if(Filecontent.dwFileAttributes == FILE_ATTRIBUTE_DIRECTORY)
            {
                CreateDirectory(ftarget, NULL);
                Copydir(fsource, ftarget);
            }
            else
            {
                Copyfile(fsource, ftarget);
            }
        }
    }
    //打开源文件和目标文件
    HANDLE hsource = CreateFile(sourcedir,
        GENERIC_READ | GENERIC_WRITE,
        FILE_SHARE_READ,
        NULL,
        OPEN_ALWAYS,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_BACKUP_SEMANTICS,
        NULL);
    HANDLE htarget = CreateFile(targetdir,
        GENERIC_READ | GENERIC_WRITE,
        FILE_SHARE_READ,
        NULL,
        OPEN_ALWAYS,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_BACKUP_SEMANTICS,
        NULL);

    //在将数据写文件之后，要改版目标文件的属性
    DWORD attr = GetFileAttributes(sourcedir);
    SetFileAttributes(targetdir, attr);
    FILETIME createtime, lastvisittime, writetime;
    GetFileTime(hsource, &createtime, &lastvisittime, &writetime);
    SetFileTime(htarget, &createtime, &lastvisittime, &writetime);
```

```
    CloseHandle(hsource);
    CloseHandle(htarget);
    CloseHandle(hfile);
    return;
}
```

Copyfile 函数：完成对普通文件的复制

打开源文件和目标文件，将每次读取的数据保存在 buffer 中，记录每次读取的长度，保存到 DWORD tp 中，读取过之后，使用 WriteFile()将数据写入目标文件中，写入后，更改目标文件的属性与源文件相同。

```
void Copyfile(char* sourcefile, char* targetfile)
{
    //打开源文件和目标文件
    HANDLE source = CreateFile(
        sourcefile,         //源文件
        GENERIC_READ,       //只读访问
        FILE_SHARE_READ,    //共享读
        NULL,               //默认安全
        OPEN_EXISTING,      //打开已存在的文件
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_BACKUP_SEMANTICS,
        NULL
    );
    HANDLE target = CreateFile(
        targetfile,          //目标文件路径
        GENERIC_READ | GENERIC_WRITE,  //读写访问
        FILE_SHARE_READ,    //共享读
        NULL,               //默认安全
        CREATE_ALWAYS,      //创建新文件
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_BACKUP_SEMANTICS,
        NULL);
    if (source == INVALID_HANDLE_VALUE)
    {
        源文件错误 关闭
    }

    char buffer[buffer_size];
    DWORD tp = 0;//记录每次读取的数据长度
    //ReadFile:(文件句柄，读入数据缓冲区，读入字节数，指向实际读取字节数指针)
    //当程序调用成功时,将实际读出文件的字节数保存到 &tp 指明的地址空间中。
    while (ReadFile(source, buffer, buffer_size, &tp, NULL))
    {
        //使用 WriteFile()将数据写入一个文件。
```

```
        WriteFile(target, buffer, tp, &tp, NULL);
        if (tp < buffer_size)break;
    }

    //在将数据写文件之后，要改版目标文件的属性
    更改目标文件的属性，与 Copydir()函数一致，不再赘述
    return;
}
```

2. Linux 系统下文件复制的实现步骤：

Linux 文件系统出了判断目录文件和普通文件歪，还需要特别判断链接文件做出处理。

主函数与 Windows 系统下的主函数系统差别不大，调用接口为 lstat()函数以及判断该路径是否有目录的 S_ISDIR()函数。如果找不到目标目录，就 mkdir()，之后调用 Copydir()函数。

```
int main(int argc, char* argv[])
{
    if (argc != 3)
    {
        printf("Error input. Please Check.\n");
        return 0;
    }

    struct stat statbuf;
    lstat(argv[1], &statbuf);//返回关于文件相关的信息

    if (S_ISDIR(statbuf.st_mode))//判断该路径是否是目录
    {
        if (opendir(argv[1]) == NULL)
        {
            printf("No Source File name: %s", argv[1]);
            return 0;
        }
        if (opendir(argv[2]) == NULL)
        {
            mkdir(argv[2], statbuf.st_mode);
        }
        Copydir(argv[1], argv[2]);
    }
    printf("Copy Finished.\n");
    return 0;
}
```

Copydir(char* sourcedir,char*targetdir)函数中，首先打开文件目录，调用readdir()接口，跳过当前目录和上级目录(.以及..)，之后保存源文件和目标文件的信息结构。如果是链接文件的话，就调用 Copylink()函数；如果是目录文件的话，就调用 Copydir()函数；如果是普通文件，就调用 Copyfile()函数。之后要通过 ChangeAttr()函数改变源文件和目标文件的属性。

```
void Copydir(char* sourcedir,char*targetdir)
{
    printf("in copydir\n");
    //打开文件或目录
    DIR* pd_source = opendir(sourcedir);
    struct dirent* entry_source = NULL;
    struct stat statbuf;

    while((entry_source = readdir(pd_source))) {
        printf("IN COPYDIR CIRCLE\n");
        //跳过当前目录和上级目录:
        if (strcmp(entry_source->d_name, ".") == 0 ||
            strcmp(entry_source->d_name, "..") == 0)
            continue;

        char temps[buffer_size], tempt[buffer_size];

        strcpy(temps, sourcedir);
        strcat(temps, "/");
        strcat(temps, entry_source->d_name);

        strcpy(tempt, targetdir);
        strcpy(tempt, "/");
        strcat(tempt, entry_source->d_name);
        //保存源文件的信息结构
        lstat(temps, &statbuf);
        //如果是 LINK
        if (S_ISLNK(statbuf.st_mode))
        {
            Copylink(temps, tempt);
        }
        //如果是 DIR
        else if (S_ISDIR(statbuf.st_mode))
        {
            mkdir(tempt, statbuf.st_mode);
            printf("IS DIR\n");
            Copydir(temps, tempt);
```

```
        }
        //如果是 REG
        else if (S_ISREG(statbuf.st_mode))
        {
            Copyfile(temps, tempt);
        }

    }
    ChangeAttr(sourcedir, targetdir);
    return;
}
```

Copylink(char* sourcelink, char* targetlink)函数，只需要读取软链接内容，复制到目标链接文件即可，之后修改目标链接文件的属性。

```
void Copylink(char* sourcelink, char* targetlink)
{
    printf("in copy link\n");
    //读取软连接内容并复制
    unsigned char path[buffer_size];
    readlink(sourcelink, path, buffer_size);
    //复制内容到目标连接文件
    symlink(path, targetlink);
    //修改属性
    ChangeAttr(sourcelink, targetlink);
}
```

Copyfile(char* sourcelink, char* targetlink)函数，与 Windows 中的文件函数差别不大。

```
void Copyfile(char* sourcefile, char* targetfile)
{
    printf("Copying file\n");
    //打开源文件
    int fd_sor = open(file_sor, O_RDONLY);
    //创建目标文件
    int fd_tar = creat(file_tar, O_WRONLY);
    //文件读写
    unsigned char buff[buffer_size];
    int len;
    while ((len = read(fd_sor, buff, buffer_size)) > 0)
    {
        printf("IN CIRCLE\n");
        write(fd_tar, buff, len);
    }
    //修改属性
```

```
    ChangeAttr(file_sor, file_tar);

    close(fd_sor);
    close(fd_tar);
}
```

ChangeAttr(char* sourcefile, char* targetfile),需要先保存源文件属性信息结构，修改文件所有者，之后修改链接文件的访问和修改时间，修改其中的文件访问和修改时间。调用 utime()和 lutimes()的 API 接口。

```
void ChangeAttr(char* sourcefile, char* targetfile)
{
    struct stat statbuf;
    lstat(sourcefile, &statbuf);
    //保存源文件属性信息结构
    chmod(targetfile, statbuf.st_mode);
    //修改文件所有者
    chown(targetfile, statbuf.st_uid, statbuf.st_gid);

    if (S_ISLNK(statbuf.st_mode))
    {
        //修改链接文件访问和修改时间
        struct timeval time_source[2];
        time_source[0].tv_sec = statbuf.st_mtime;
        time_source[1].tv_sec = statbuf.st_ctime;
        lutimes(targetfile, time_source);
    }
    else
    {
        //修改文件访问和修改时间
        struct utimbuf utime_source;
        utime_source.actime = statbuf.st_atime;
        utime_source.modtime = statbuf.st_mtime;
        utime(targetfile, &utime_source);
    }
    return;
}
```

五、实验结果和分析：

1. Windows 下的实验结果：

Windows 下 sourcefile 文件内容如下图所示：

```
D:\张驰\学习\OS课设\实验5\windows>dir/q sourcefile
 驱动器 D 中的卷是 D:
 卷的序列号是 1C4E-9BDF

 D:\张驰\学习\OS课设\实验5\windows\sourcefile 的目录

2021/12/10  11:36    <DIR>          LAPTOP-QFOIDU1C\LX     .
2021/12/10  11:36    <DIR>          LAPTOP-QFOIDU1C\LX     ..
2021/12/09  13:50             3,825 LAPTOP-QFOIDU1C\LX     1 - 副本 (2).txt
2021/12/09  11:20                 5 LAPTOP-QFOIDU1C\LX     1 - 副本 (3).txt
2021/12/09  11:20                 5 LAPTOP-QFOIDU1C\LX     1 - 副本 (4).txt
2021/12/09  11:20                 5 LAPTOP-QFOIDU1C\LX     1 - 副本.txt
2021/12/09  11:57                 8 LAPTOP-QFOIDU1C\LX     1.txt
2021/12/10  11:36    <DIR>          LAPTOP-QFOIDU1C\LX     2020.10月-12月
2021/12/09  13:50    <DIR>          LAPTOP-QFOIDU1C\LX     a1
2021/11/18  23:06            22,615 LAPTOP-QFOIDU1C\LX     张驰.docx
               6 个文件         26,463 字节
               4 个目录 123,295,879,168 可用字节
```

使用命令：mycp sourcefile targetfile，将 sourcefile 文件夹下的内容放到 targetfile 中。文件命令和 targetfile 文件的内容如图所示：

```
D:\张驰\学习\OS课设\实验5\windows>mycp sourcefile targetfile

D:\张驰\学习\OS课设\实验5\windows>dir/q targetfile
 驱动器 D 中的卷是 D:
 卷的序列号是 1C4E-9BDF

 D:\张驰\学习\OS课设\实验5\windows\targetfile 的目录

2021/12/10  11:36    <DIR>          LAPTOP-QFOIDU1C\LX     .
2021/12/10  11:36    <DIR>          LAPTOP-QFOIDU1C\LX     ..
2021/12/09  13:50             3,825 LAPTOP-QFOIDU1C\LX     1 - 副本 (2).txt
2021/12/09  11:20                 5 LAPTOP-QFOIDU1C\LX     1 - 副本 (3).txt
2021/12/09  11:20                 5 LAPTOP-QFOIDU1C\LX     1 - 副本 (4).txt
2021/12/09  11:20                 5 LAPTOP-QFOIDU1C\LX     1 - 副本.txt
2021/12/09  11:57                 8 LAPTOP-QFOIDU1C\LX     1.txt
2021/12/10  11:36    <DIR>          LAPTOP-QFOIDU1C\LX     2020.10月-12月
2021/12/09  13:50    <DIR>          LAPTOP-QFOIDU1C\LX     a1
2021/11/18  23:06            22,615 LAPTOP-QFOIDU1C\LX     张驰.docx
               6 个文件         26,463 字节
               4 个目录 123,279,601,664 可用字节
```

目标文件与源文件信息完全相同，复制命令完成。

2. Linux 系统下的实验结果：

源文件内容：

```
zc1120191600@ubuntu:~/code/EX5$ ls -l sourcefile
total 12
-rw-r--r-- 1 zc1120191600 zc1120191600  125 Dec  8 22:57 1.txt
drwxr-xr-x 2 zc1120191600 zc1120191600 4096 Dec  8 22:58 a.a
-rw-r--r-- 1 zc1120191600 zc1120191600   80 Dec  8 22:57 text.txt
```

调用 mycp 命令，目标文件内容如下：

```
zc1120191600@ubuntu:~/code/EX5$ ./mycp sourcefile targetfile
in copydir
IN COPYDIR CIRCLE
IN COPYDIR CIRCLE
IN COPYDIR CIRCLE
Copying file
IN CIRCLE
IN COPYDIR CIRCLE
in copydir
IN COPYDIR CIRCLE
IN COPYDIR CIRCLE
IN COPYDIR CIRCLE
Copying file
IN CIRCLE
IN COPYDIR CIRCLE
Copying file
IN CIRCLE
IN COPYDIR CIRCLE
Copying file
IN CIRCLE
Copy finished
zc1120191600@ubuntu:~/code/EX5$ ls -l targetfile
total 12
-rw-r--r-- 1 zc1120191600 zc1120191600  125 Dec  8 22:57 1.txt
drwxr-xr-x 2 zc1120191600 zc1120191600 4096 Dec  8 22:58 a.a
-rw-r--r-- 1 zc1120191600 zc1120191600   80 Dec  8 22:57 text.txt
```

可以看到，目标目录下的个文件复制完成，大小、时间、权限等属性与源目录下的文件一致。

六、讨论、心得

通过本次文件复制的实验，我进一步了解了 windows 和 linux 中的文件复制、读写的接口函数 API。由于对文件读写的函数不熟悉，开始时采用一次性读取整个文件，全部写入文件中。但是这样做会在大文件的复制与读写上产生问题。经过仔细研究之后，我发现采用缓冲区读写的方式，可以完美解决大文件的读写与复制问题。同时，Linux 系统中包含软链接和硬链接两种方式，所以需要对 Linux 的软链接文件进行了一定的处理。

通过本次实验，我更进一步理解了 Windows 操作系统中的多级文件目录的模型，更深入的理解了 Linux 系统中的软链接和硬链接的两种文件类型，对两个系统的 API 有了进一步的了解，可以使用在之后的学习中。