实验四 文件复制

班级： 07812101 学号： 1820211062 姓名： 洪子翔

一、实验目的

独立设计并实现一个文件复制命令，熟悉Linux文件系统提供的有关文件操作的系统调用，加深对文件系统实现功能的理解。

1. 实验内容
2. 在Linux系统下实现目录复制命令，新实现的命令命名为"mycopy"。
3. 使用mycopy命令能够支持多级目录（子目录）的复制，支持Linux下的soft link复制。
4. 要求使用命令行参数接受参与文件复制的源文件和目标文件，mycopy命令的使用方法为： mycopy src dest，其中src为源文件，dest为目标文件。 4. 示例：若源目录src结构如图 (a)所示，则运行“mycopy Group dest”后，目录dest的结构应如图(b)所示。

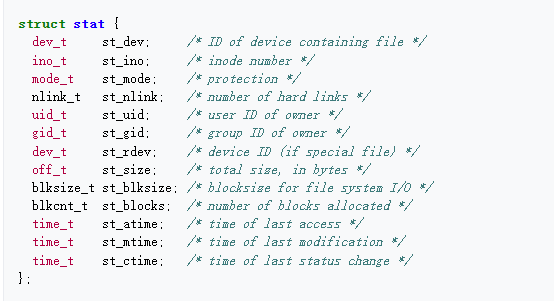
|  |  |
| --- | --- |
| Group | dest |
| |\_\_\_\_Paper  |\_\_\_\_1.pdf  |\_\_\_\_2.docx  |\_\_\_\_Slide  |\_\_\_\_x.pptx  |\_\_\_\_Project  |\_\_\_\_source code  |\_\_\_\_code.rar  |\_\_\_\_documents  |\_\_\_\_doc.zip  (a) | |\_\_\_\_Paper  |\_\_\_\_1.pdf  |\_\_\_\_2.docx  |\_\_\_\_Slide  |\_\_\_\_x.pptx  |\_\_\_\_Project  |\_\_\_\_source code  |\_\_\_\_code.rar  |\_\_\_\_documents  |\_\_\_\_doc.zip  (b) |

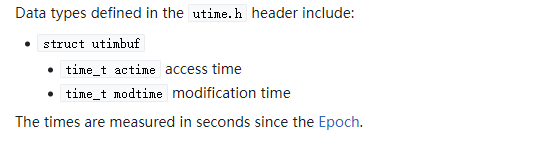
5. 在Linux系统下可使用mkdir、opendir、readdir、symlink、readlink等函数。

1. 程序设计与实现

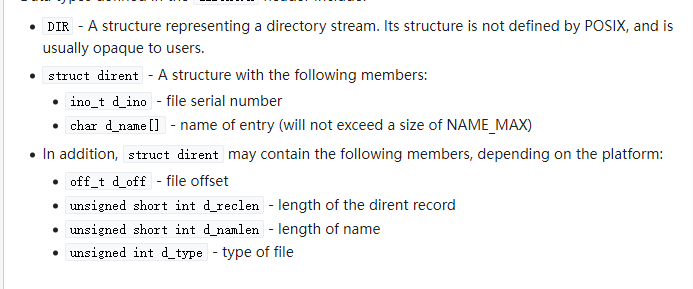
实验环境：Linux，Ubuntu18.04，Visual Studio Code,C语言

首先我们先来查看struct stat 里的结构

 Utime结构



Dirent结构体



首先是导入库

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/time.h>

#include <utime.h>

#include <dirent.h>

#define READSIZE 1024

#define LinkPathLen 1024

接着是main函数

int main(int argc, char \*argv[])

{

if (argc != 3)

{

printf("复制语句格式出现错误!\n");

printf("格式为 ./mycopy3 src dest\n");

exit(-1);

}

struct stat statbuf;

struct utimbuf uTime;

DIR \*dptr2 = NULL;

if (stat(argv[1], &statbuf) == 0)

{

if (S\_ISREG(statbuf.st\_mode))

{

// 如果源是文件而不是目录，则直接复制文件

printf("FileName: %s, 正在拷贝...\n", argv[1]);

CopyFile(argv[1], argv[2]);

printf("FileName: %s, 拷贝完成\n", argv[1]);

}

else if (S\_ISDIR(statbuf.st\_mode))

{

// 如果源是目录，则创建目录，然后递归复制目录内容

if ((dptr2 = opendir(argv[2])) == NULL)

{

uTime.actime = statbuf.st\_atime;

uTime.modtime = statbuf.st\_mtime;

if (mkdir(argv[2], statbuf.st\_mode) < 0)

{

printf("创建目录失败\n");

exit(-1);

}

MyCopy(argv[1], argv[2]);

utime(argv[2], &uTime);

printf("目录复制完成\n");

}

else

{

printf("目标目录已存在\n");

exit(-1);

}

}

else

{

printf("源既不是文件也不是目录\n");

exit(-1);

}

}

else

{

printf("获取文件信息失败\n");

exit(-1);

}

if (dptr2 != NULL)

{

closedir(dptr2);

}

return 0;

}

文件复制部分

CopyFile 函数用于复制单个文件。它打开源文件，创建目标文件，然后逐 块复制数据，最后设置目标文件的访问和修改时间。

void CopyFile(char \*src, char \*dest)

{

struct stat statbuf;

struct utimbuf uTime;

int destFd;

int srcFd;

int size = 0;

char buffer[READSIZE];

memset(buffer, 0, sizeof(buffer));

stat(src, &statbuf);

destFd = creat(dest, statbuf.st\_mode);

if ((srcFd = open(src, O\_RDONLY)) < 0)

{

printf("打开文件:%s 出现错误!\n", src);

exit(-1);

}

while ((size = read(srcFd, buffer, READSIZE)) > 0)

{

write(destFd, buffer, size);

}

uTime.actime = statbuf.st\_atime;

uTime.modtime = statbuf.st\_mtime;

utime(dest, &uTime);

close(srcFd);

close(destFd);

}

软连接文件复制

CopyLinkFile 函数用于复制软连接文件。它获取软连接文件的路径信息， 然后在目标目录下创建一个软连接，并设置权限和时间属性。

void CopyLinkFile(char \*LinkPath, char \*dest)

{

struct stat statbuf;

struct timeval tv[2];

char path\_buf[LinkPathLen];

memset(path\_buf, 0, sizeof(path\_buf));

lstat(LinkPath, &statbuf);

readlink(LinkPath, path\_buf, LinkPathLen);

if (symlink(path\_buf, dest) == -1)

{

printf("建立软连接失败!\n");

\_exit(-1);

}

printf("软连接文件复制成功\n");

chmod(dest, statbuf.st\_mode);

tv[0].tv\_sec = statbuf.st\_atime;

tv[0].tv\_usec = 0;

tv[1].tv\_sec = statbuf.st\_mtime;

tv[1].tv\_usec = 0;

lutimes(dest, tv);

}

目录递归复制

MyCopy 函数用于递归复制目录。它遍历源目录中的文件和子目录，根据文 件类型调用相应的复制函数，同时递归处理子目录。

void MyCopy(char \*src, char \*dest)

{

struct stat statbuf;

struct stat copybuf;

struct utimbuf uTime;

struct dirent \*entry = NULL;

DIR \*dptr1 = NULL;

char src\_path[128], dest\_path[128];

strcpy(src\_path, src);

strcpy(dest\_path, dest);

dptr1 = opendir(src\_path);

if (dptr1 != NULL)

{

while ((entry = readdir(dptr1)) != NULL)

{

if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0)

continue;

strcpy(dest\_path, dest);

strcpy(src\_path, src);

strcat(src\_path, "/");

strcat(dest\_path, "/");

strcat(src\_path, entry->d\_name);

strcat(dest\_path, entry->d\_name);

lstat(src\_path, &statbuf);

if (S\_ISDIR(statbuf.st\_mode))

{

printf("Directory: %s, 正在拷贝...\n", src\_path);

stat(src\_path, &copybuf);

mkdir(dest\_path, copybuf.st\_mode);

MyCopy(src\_path, dest\_path);

uTime.actime = copybuf.st\_atime;

uTime.modtime = copybuf.st\_mtime;

utime(dest\_path, &uTime);

printf("Directory: %s, 拷贝完成\n", src\_path);

}

else if (S\_ISLNK(statbuf.st\_mode))

{

printf("LinkFileName: %s, 正在拷贝...\n", src\_path);

CopyLinkFile(src\_path, dest\_path);

}

else

{

printf("FileName: %s, 正在拷贝...\n", src\_path);

CopyFile(src\_path, dest\_path);

printf("FileName: %s, 拷贝完成\n", src\_path);

}

}

closedir(dptr1);

}

else

{

// 如果源是文件而不是目录，则直接复制文件

printf("FileName: %s, 正在拷贝...\n", src);

CopyFile(src, dest);

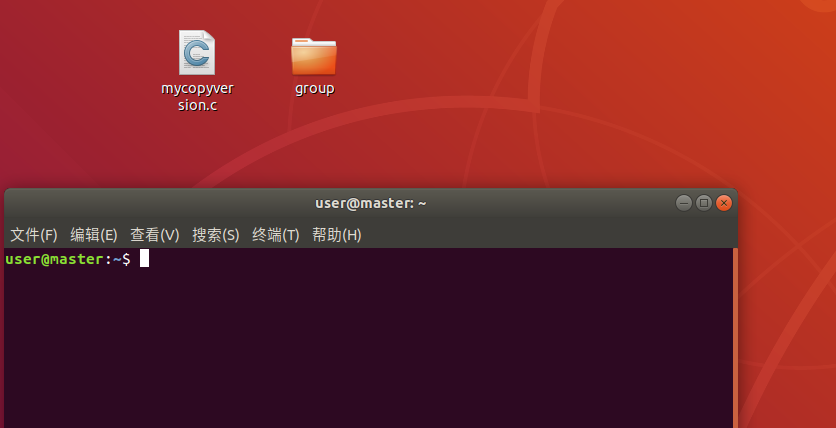
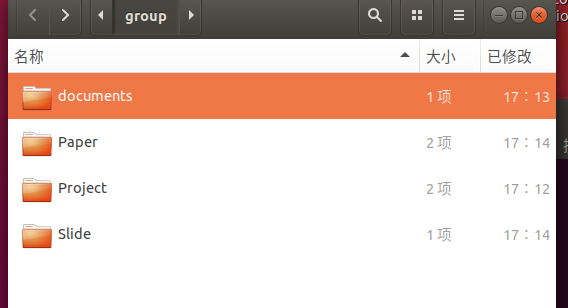
printf("FileName: %s, 拷贝完成\n", src);

}

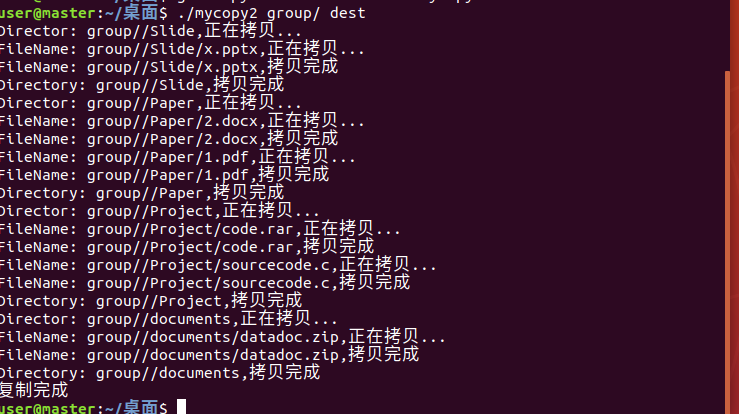
}

四、实验运行结果及分析

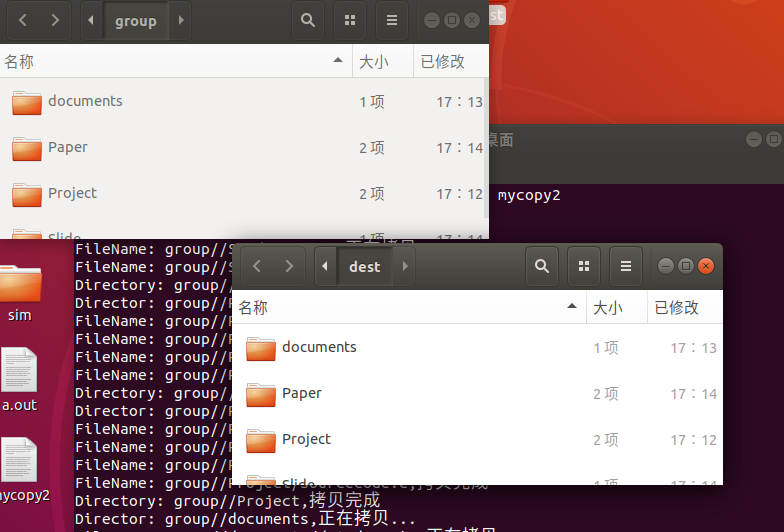
准备好程序以及将要复制的目录，这里的文件效仿要求。

目录中包含了各种文件如pdf，word，pptx，zip和rar以展示复制效果。

编译后进行拷贝



拷贝成功



1. 预先创建测试文件，其结构如下：

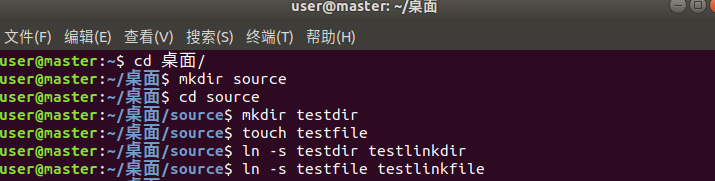
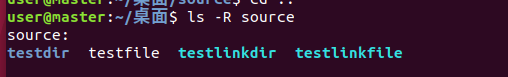
source

|\_testdir

|\_testfile

|\_testlinkdir

|\_testlinkfile



其中testlinkdir与testlinkfile为soft link，分别指向testdir与 testfile

1. 进入source所在目录，测试非空文件夹复制

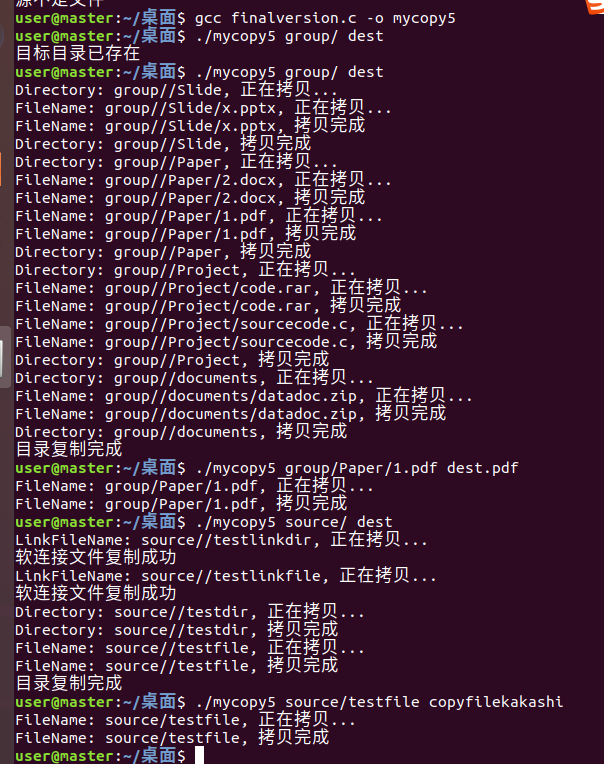
./mycopy2 source dest



1. 测试文件复制

mycopy source/testfile destfile

这里可以注意到我使用了修改后的代码，此前的代码只能复制目录

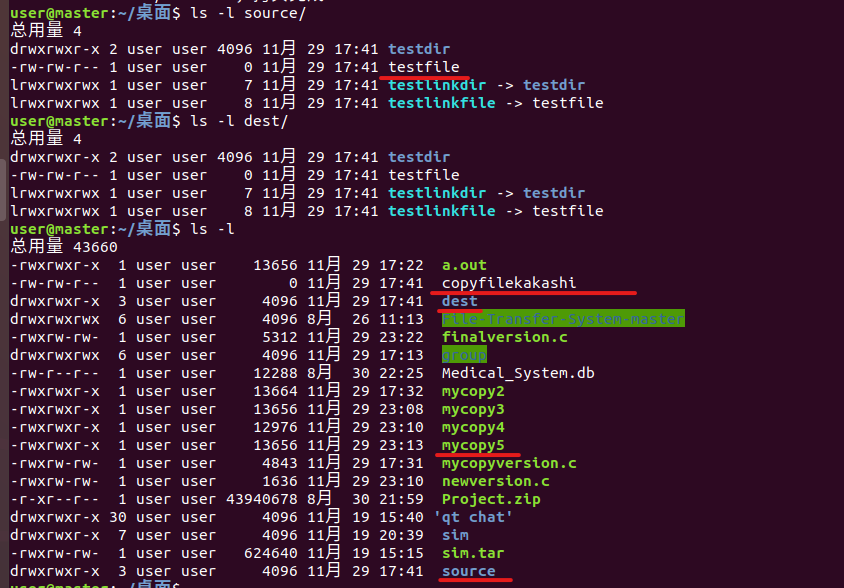


4. 展示复制结果

ls -l

ls -l source

ls -l dest



可以看到成功了！

五、实验收获与体会

在本次实验中，我无意中忽略了单独复制文件的功能。因此，在接近完成时，我进行了测试，发现我的主函数仅设置了复制源目录的逻辑。为了解决这个问题，我在这一阶段付出了更多的努力，重新设计并重写了主函数，最终成功实现了文件和目录的复制功能。这次实验要求在Linux上进行，相较于Windows，我发现在Linux平台上更加容易上手。总的来说，通过这次实验，我获得了丰富的经验和知识。

附录：程序清单及说明

Finalversion.c