实验四 文件复制

班级： 07112201 学号： 1120220715 姓名： 刘秉致

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。

二、实验内容

1. 在Linux系统下实现目录复制命令，新实现的命令命名为"mycopy"。

2. 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。

3. 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。

三、实验步骤

**1.实验环境**

本次实验将在WSL2 + Ubuntu环境下使用C++代码编写。

**2.思路分析**

考虑到自定义命令需要实现对普通文件、软连接和目录的复制，因此考虑对不同的文件类型分开进行操作，采用模块化的设计思路。

**1） 文件复制**

对文件的复制考虑采用C库函数中的文件操作函数。首先根据源文件和目的文件的路径信息分别创建两个打开文件描述符。接下来将源文件中的内容按块写入目的文件中。最终，关闭两个打开文件

bool copy\_file(const char\* src,const char\* dest){

    FILE\* in = fopen(src,"rb");

    if(in == NULL){

        perror("fail to open file.");

        return false;

    }

    FILE\* out = fopen(dest,"wb");

    if(out == NULL){

        perror("fail to open file.");

        fclose(in);

        return false;

    }

    char buf[4096]={0};

    size\_t size;

    while((size = fread(buf,1,sizeof(buf),in)) > 0){

        fwrite(buf,1,size,out);

    }

    fclose(in);

    fclose(out);

    return true;

}

**2）软连接复制**

对软连接的复制需要先访问源软连接，获得源软连接所指向的文件路径，并在新的目标路径下建立新的软连接。

bool copy\_link(const char\* src,const char\* dest){

    char file\_path[4096]={0};

    ssize\_t size=readlink(src,file\_path,sizeof(file\_path)-1);//an empty char for '\0'

    if(size==-1){

        perror("fail to read link.");

        return false;

    }

    file\_path[size]  ='\0';//add '\0' to the end of the string

    if(symlink(file\_path,dest)==-1){

        perror("fail to create link.");

        return false;

    }

    return true;

}

**3）目录复制**

对目录的复制需要进行递归操作：首先打开源目录，获得源目录结构体；接下来尝试创建目的目录，如果目的目录已经存在则这一步可以忽略；对源目录内每一条目录项进行遍历，分别进行复制，必要时递归的进行复制；最后关闭当前目录结构体

bool copy\_dir(const char\* src,const char\* dest){

    DIR\* dir=opendir(src);

    if(dir==NULL){

        perror("fail to open directory.");

        return false;

    }

    if(mkdir(dest,0755)==-1 && errno!=EEXIST){

        perror("fail to create directory.");

        return false;

    }

    //Copy directory contents recursively

    struct dirent\* entry;

    while((entry = readdir(dir))!=NULL){

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

            continue;

        }

        char src\_path[4096]={0};

        char dest\_path[4096]={0};

        sprintf(src\_path,"%s/%s",src,entry->d\_name);

        sprintf(dest\_path,"%s/%s",dest,entry->d\_name);

        struct stat st;

        if(lstat(src\_path,&st)==-1){

            perror("fail to get file status.");

            return false;

        }

        if(S\_ISREG(st.st\_mode)){

            if(copy\_file(src\_path,dest\_path)==false){

                perror("fail to copy file.");

                return false;

            }

        }

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

            if(copy\_link(src\_path,dest\_path)==false){

                perror("fail to copy link.");

                return false;

            }

        }

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

            if(copy\_dir(src\_path,dest\_path)==false){

                perror("fail to copy directory.");

                return false;

            }

        }

        else{

            perror("unknow file type.");

            return false;

        }

    }

    closedir(dir);

    return true;

}

**4）主程序**

主要对源文件进行分析，判断文件属于以上哪种，调用相关函数进行复制。

int main(int argc,char\*\* argv){

    if(argc!=3){

        fprintf(stderr,"Usage: %s <src> <dest>\n",argv[0]);

        return -1;

    }

    const char\* src=argv[1];

    const char\* dest=argv[2];

    struct stat st;

    if(lstat(src,&st)==-1){

        perror("fail to get file status.");

        return -1;

    }

    if(S\_ISREG(st.st\_mode)){

        if(copy\_file(src,dest)==false){

            perror("fail to copy file.");

            return -1;

        }

    }

    else if(S\_ISLNK(st.st\_mode)){

        if(copy\_link(src,dest)==false){

            perror("fail to copy link.");

            return -1;

        }

    }

    else if(S\_ISDIR(st.st\_mode)){

        if(copy\_dir(src,dest)==false){

            perror("fail to copy directory.");

            return -1;

        }

    }

    else{

        perror("unknow file type.");

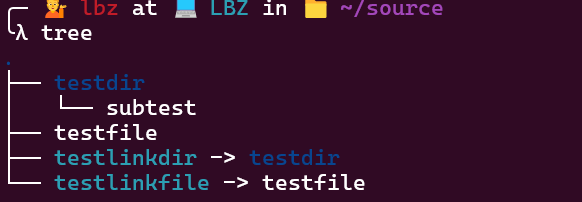
        return -1;

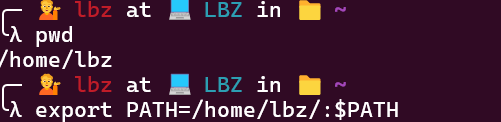
    }

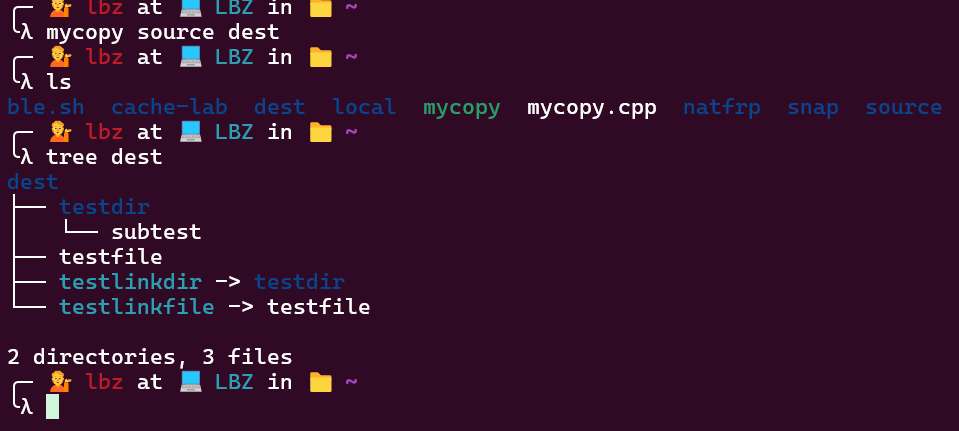
    return 0;

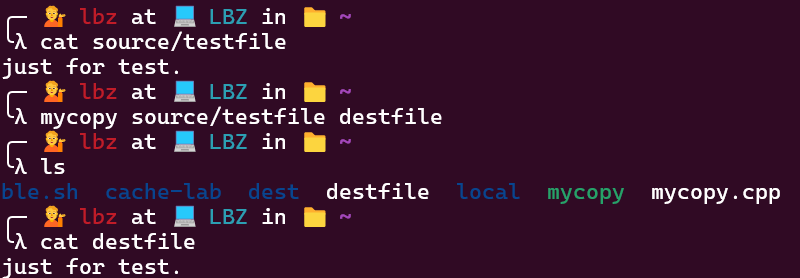
}

四、实验结果及分析

1.预先在用户目录下创建测试文件，结构如下：

2. 将mycopy可执行文件所在路径写入PATH变量

3. 进入source所在目录，测试非空文件夹复制，发现成功复制

4.测试文件复制，发现复制成功

至此，命令功能全部正常实现

五、实验收获与体会

本次的实验，首先教会了我如何实现在Linux系统下实现一个自定义命令；其次很好的帮助了我了解了Linux下目录与目录项的组织方式和软连接的实现原理，并且使我学习到了如何使用API来操作这些内容。

附录：程序清单及说明

1.mycopy.cpp 源代码