实验四 文件复制

班级： xxx 学号： xxx 姓名： xxx

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

三、程序设计与实现

1. 调用lstat方法来判断文件类型是目录、普通文件还是链接文件。

struct stat s;

if (lstat(sourcePath.c\_str(), &s) == -1)

{

    perror("stat error!");

    return false;

}

1. S\_ISDIR(s.st\_mode)：如果是目录则返回true
2. S\_ISREG(s.st\_mode)：如果是普通文件则返回true
3. S\_ISLNK(s.st\_mode)：如果是软链接则返回true
4. 对于普通文件，通过调用open、read和write函数来实现文件复制。
5. 对于源文件，通过只读方式打开文件；对于目标文件，通过文件存在可写、文件不存在时创建方式打开文件。

int sourceFileHandle, destFileHandle;

sourceFileHandle = open(sourcePath.c\_str(), O\_RDONLY);

destFileHandle = open(destPath.c\_str(), O\_WRONLY | O\_CREAT, s.st\_mode);

if (sourceFileHandle == -1 || destFileHandle == -1)

{

    perror("open error!");

    return false;

}

1. 创建缓存区，对源文件循环调用read方法将内容读到缓存区，并调用write方法将缓存区内容写入到目标文件中。

// 缓存区

char buf[1024] = {0};

int len;

while ((len = read(sourceFileHandle, buf, 1024)) != 0)

{

    write(destFileHandle, buf, len);

}

1. 关闭打开的文件资源

close(sourceFileHandle);

close(destFileHandle);

1. 对于链接文件，通过调用readlink将源文件中路径内容读到缓存区，然后通过调用symlink创建软链接并指向缓存区中的路径。

char buf[1024] = {0};

int len = readlink(sourcePath.c\_str(), buf, sizeof(buf));

if (len == -1)

{

    perror("readlink error");

    return false;

}

if(symlink(buf, destPath.c\_str()) == -1)

{

    perror("symlink error!");

    return false;

}

1. 对于目录，首先要读取源目录下的目录项，然后对每个目录项判断其是普通文件、链接文件还是目录，即要进行递归处理。
2. 首先调用opendir方法打开源目录，然后创建目标目录。

DIR \*dirp = opendir(sourcePath.c\_str());

if (dirp == nullptr)

{

    perror("opendir error!");

    return false;

}

mkdir(destPath.c\_str(), s.st\_mode);

1. 然后调用readdir方法来读取源目录下的目录项，对每个目录项递归调用mycopy函数，注意要排除当前目录（.）和上级目录（..），否则会造成错误。

struct dirent \*direntp;

while ((direntp = readdir(dirp)) != nullptr)

{

    string tempSourcePath = sourcePath;

    string tempDestPath = destPath;

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

        continue;

    bool res = mycopy(tempSourcePath.append("/").append(direntp->d\_name).c\_str(), tempDestPath.append("/").append(direntp->d\_name).c\_str());

    if (res == false)

        return false;

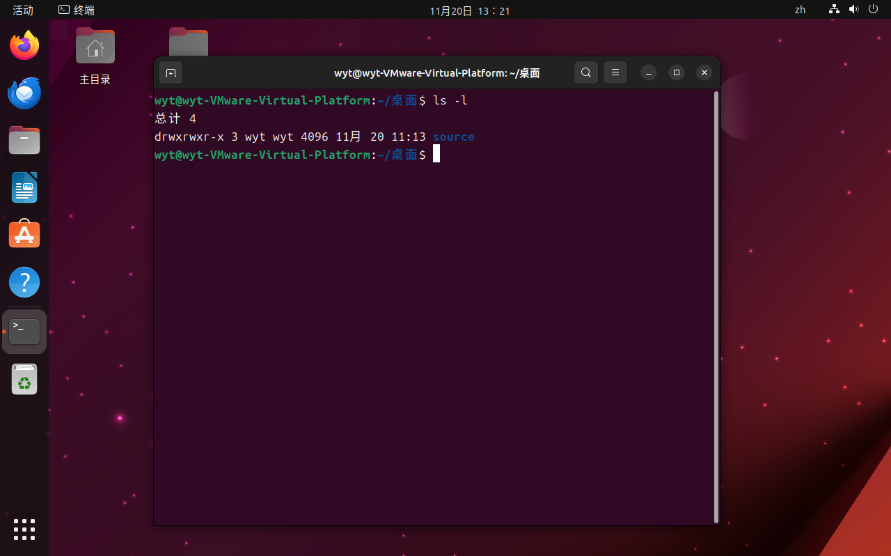
}

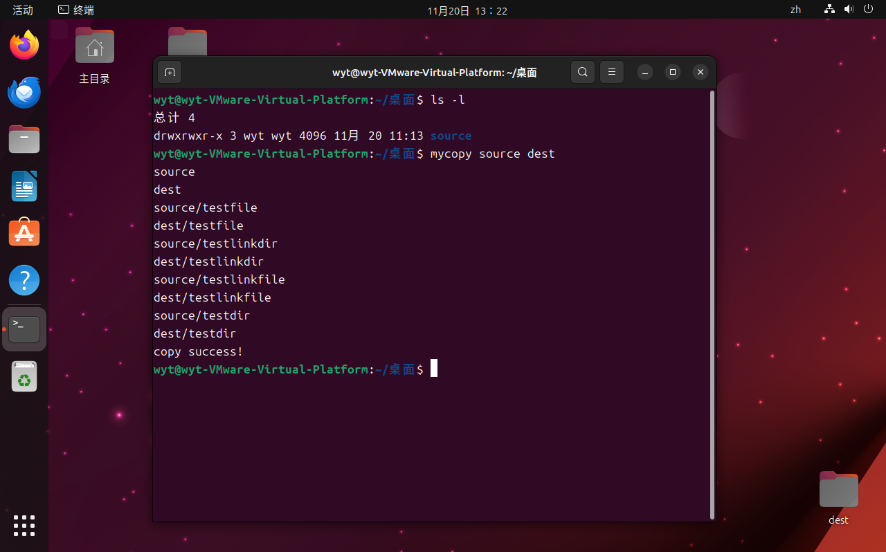
1. 最后调用closedir关闭打开的源目录。

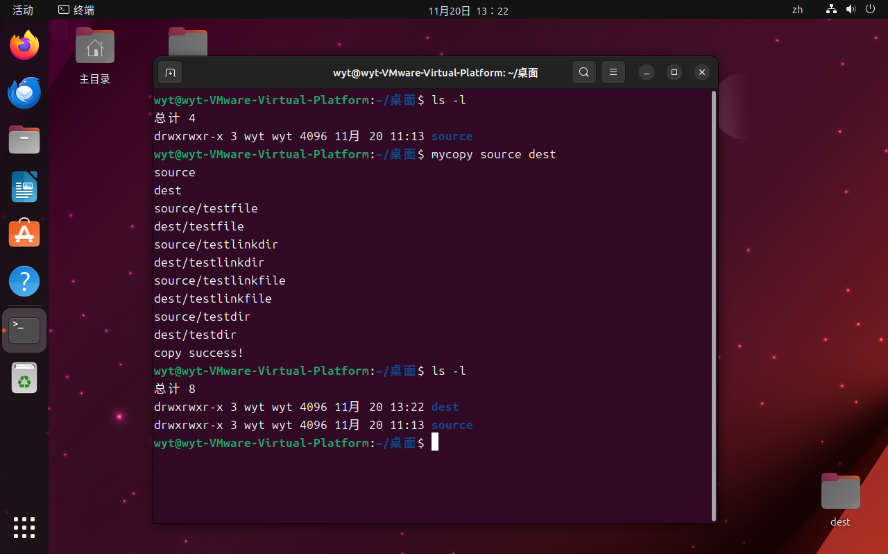
closedir(dirp);

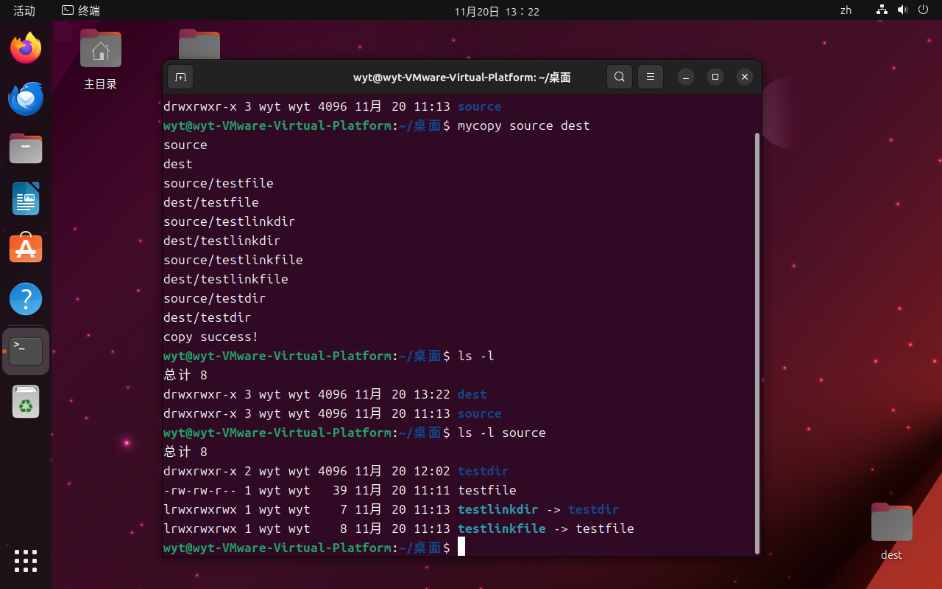
四、实验结果及分析

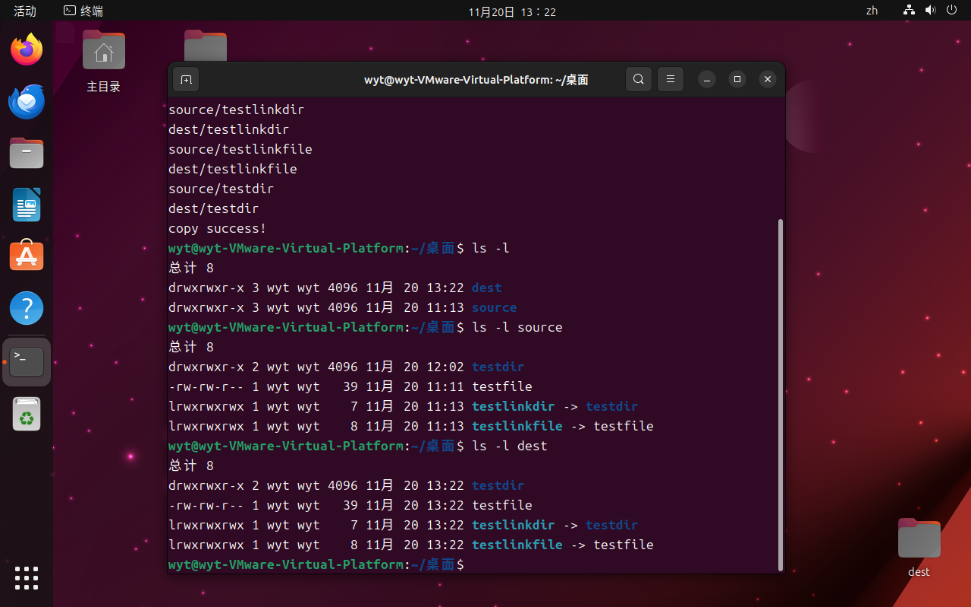
首先在终端中输入命令“g++ mycopy.cpp -o mycopy”来编译源文件，并将生成的mycopy加入到环境变量，从而可以在任意位置使用mycopy程序。

运行mycopy命令前的目录项如下：

运行mycopy命令过程如下：

运行mycopy命令后的目录项如下：

查看source文件夹下的目录项：

查看dest文件夹下的目录项：

从上面的结果可以看出，我成功实现了mycopy命令，它可以在Linux系统下复制多级目录（包括子目录）并支持软链接的复制。

同时我还进行了一系列功能测试，包括复制单个文件、复制目录、复制软链接等场景，mycopy命令成功地完成了所有测试，并得到了预期的结果；我也测试了大文件复制，由于使用了较大的缓存区，所以大文件的复制速度也很快。

五、实验收获与体会

在实验内容中，要求支持多级目录的复制和Linux下的软链接复制。通过实现这些功能，我更加深入地了解了目录和软链接的内部机制。在复制目录时，需要使用opendir()和readdir()函数遍历目录中的所有文件和子目录，并逐个复制。而在复制软链接时，需要使用readlink()函数获取软链接指向的文件路径，并复制该路径所指向的文件。这些操作让我更加深入地了解了Linux文件系统中的目录和软链接的实现方式。

与Linux系统自带的cp命令相比，mycopy命令在功能上有所简化，但它成功实现了多级目录的复制和软链接的复制。通过实现一个文件复制命令，我更加深入地了解了Linux文件系统的内部机制，包括文件的打开、读取、写入等操作，这些操作在文件系统中扮演着重要的角色，而通过编程语言和系统调用的结合，我们可以直接操作文件系统，实现文件的复制、移动、删除等操作。

本次实验让我的编程能力得到进一步提高。在编程过程中，我进行了错误判断，从而增强了程序健壮性。如在打开文件和目录时，我进行了错误处理以检查系统调用是否成功，并在出现错误时输出错误信息并退出程序。

附录：程序清单及说明

**mycopy.cpp**：包含实现目录复制功能的函数