实验4 文件复制

班级： 07112002 学号： 1120201316 姓名： 金雅各

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（黑体四号字）

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. 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。

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

三、实验步骤

1. 了解熟悉linux 文件相关的数据结构与字段

① struct stat :描述一个系统文件系统中的文件属性的结构

本实验中需要使用到的字段有:

st\_mode (文件对应的模式，文件，目录等)

st\_atime (文件最后被访问的时间)

st\_mtime (文件内容最后被修改的时间)

② struct utimbuf :记录文件的访问时间与修改时间

本实验中需要使用到的字段有:

actime (文件最后被访问的时间)

modtime(文件内容最后被修改的时间)

③ DIR : 指向目录文件的指针

④ struct dirent :获取某文件夹目录下的内容

需要使用到的字段:

d\_name (指向的文件/文件夹的名称)

⑤ struct timeval :记录文件的访问时间与修改时间 （毫秒级别）

tv[0].tv\_sec 记录actime

tv[1].tv\_sec 记录modtime

2. 熟悉与文件相关的函数

1. DIR\* opendir (const char \* path );

该函数传入目录的路径，返回指向该目录的指针，失败时返回NULL

1. struct dirent\* readdir(DIR\* dir\_handle);

传入目录的指针，返回指向目录下的子文件/子文件夹的指针，每次 读取一个文件/文件夹，结束时返回NULL

1. int closedir(DIR \*dir);

传入目录的指针，关闭打开的目录，关闭成功返回0，失败返回-1，错误原因存于errno 中。

1. char \*strerror(int errno);

将错误代码转换为字符串错误信息.

1. int mkdir(const char \*pathname, mode\_t mode);

mkdir()函数以mode方式创建一个以参数pathname命名的目录，mode定义新创建目录的权限。

若目录创建成功，则返回0；否则返回-1，并将错误记录到全局变量errno中。

1. 宏: S\_ISLNK(st\_mode); 是否是一个软链接
2. 宏: S\_ISDIR(st\_mode); 是否是一个目录
3. 宏:S\_ISREG(st\_mode);是否是一个常规文件
4. int creat(const char \* pathname, mode\_tmode);

根据路径与模式创建一个普通文

返回新的文件描述词, 若有错误发生则会返回-1, 并把错误代码设给errno

1. int open(const char \*pathname, int flags);

以flags的模式打开文件pathname, flags可以为： O\_RDONLY(只读打开) O\_WRONLY(只写打开) O\_RDWR(读，写打开)

成功返回文件描述符，失败返回-1

1. ssize\_t read(int fd, void \* buf, size\_t nbytes);

通过文件描述符fd, 从打开的文件中读数据, 到缓冲区buf，缓冲区大小为 nbytes

返回值：读到的字节数，若已到文件尾，返回0；出错，返回-1。

1. ssize\_t write (int fd, const void \* buf, size\_t count);

write()会把参数buf所指的内存写入count个字节到参数fd所指的文件内。

如果顺利write()会返回实际写入的字节数（len）。当有错误发生时则返回-1，错误代码存入errno中。

1. int close(int fd);

关闭文件fd

成功返回0，出错返回-1并设置errno

1. ssize\_t readlink(const char \*path, char \*buf, size\_t bufsiz);

readlink()会将参数path的符号链接内容存储到参数buf所指的内存空间,若参数bufsiz小于符号连接的内容长度，过长的内容会被截断

执行成功则返回字符串的字符数，失败返回-1， 错误代码存于errno

1. int symlink(const char \*target, const char \*linkpath);

symlink() 对于已有的 target 建立一个名为 link 的符号连接。

如果成功， symlink返回0.。否则返回-1并设置errno。

1. int chmod(const char \*pathname, mode\_t mode);

用来更改文件pathname权限为mode

成功返回0，失败返回-1

1. int stat(const char \*pathname, struct stat \*statbuf);

获取文件pathname信息, 并将信息存储到stat 结构体statbuf中

成功返回0，失败返回-1；

1. int lstat(const char \*pathname, struct stat \*statbuf);

当一个文件是符号链接时，lstat 函数返回的是该符号链接本身的信息.当是普通文件或目录是，返回的是文件或目录的信息.

若成功，返回0；若失败，返回 -1；

1. int utime(const char \*filename, const struct utimbuf \*times);

改变文件或目录（filename）的上次访问与修改时间为times中的字段

返回值：若成功则返回0，若出错则返回-1

1. int lutimes(const char \*filename, const struct timeval tv[2]);

改变符号链接文件filename的时间属性为tv[2]

返回值：若成功则返回0，若出错则返回-1

1. int access(const char \*pathname, int mode);

可用于判断文件或者目录是否存在

pathname为文件或文件夹的路径，当前目录直接使用文件或文件夹名；mode表示检查模式，共4种模式：

R\_OK, 4 只判断是否有读权限，对应宏定义里面的00 只存在

W\_OK, 2 只判断是否有写权限，对应宏定义里面的02 写权限

X\_OK, 1 判断是否有执行权限，对应宏定义里面的04 读权限

F\_OK, 0 只判断是否存在，对应宏定义里面的05 读和写权限

如果指定的存取方式有效，则函数返回0，否则函数返回-1

1. 编写普通文件复制函数

void copyFile(const char \*src, const char \*dest);

首先使用stat()函数得到源文件的属性信息fileStat

之后将源文件文件的访问与修改时间赋值给 utb

通过creat()函数新建目标文件，得到文件描述符destFd

使用open()函数打开源文件，得到文件描述符srcFd

通过while循环与read()函数不断从srcFd中读出数据到缓冲区buffer中，同时使用write()函数将缓冲区buffer中的内容写入目标文件destFd

使用utime()函数更新目标文件的时间

最后使用close()函数关闭文件描述符

1. 编写链接文件复制函数

void copyLinkFile(const char \*src, const char \*dest);

使用lstat()函数得到软链接文件所连接的文件信息fileStat

将源文件的修改与访问时间赋值给数组tv[]

使用readlink()函数得到链接文件src的连接路径linkPath

使用symlink()函数为linkPath新建一个链接文件dest

使用chomd()函数修改新建文件的属性

使用lutimes()函数修改链接文件的时间属性

1. 编写目录复制函数

void copyDirectory(const char \*src, const char \*dest);

使用stat()函数得到源目录的属性directoryStat

将源目录的修改与访问时间赋值给utb

使用mkdir 创建新的目录 dest

调用MyCopy(src, dest); 函数递归复制源目录下的子文件与子文件夹

使用utime() 函数更新新建目录的时间属性

1. 编写递归调用函数

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

调用opendir()函数获得目录指针dir

使用while与readdir()获得目录下子文件/子文件夹的指针entry

将目录路径 src与文件名/文件夹名entry->d\_name拼接得到子文件/子文件夹路径child\_src\_path, child\_dest\_path

使用lstat()函数得到源文件/目录的属性

调用access(）函数判断目标文件/文件夹是否存在

通过宏S\_ISLNK、S\_ISDIR、S\_ISREG判断文件类型

若为链接文件，调用copyLinkFile()

若为目录文件，调用copyDirectory()

若为普通文件，调用copyFile()

若为其他文件，直接返回-1

最后使用closedir() 关闭文件目录

1. 编写主函数

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

首先判断传入的命令行参数是否为3个,(启动文件、源目录名，目标目录名)

调用access(）函数判断目标文件/文件夹是否存在

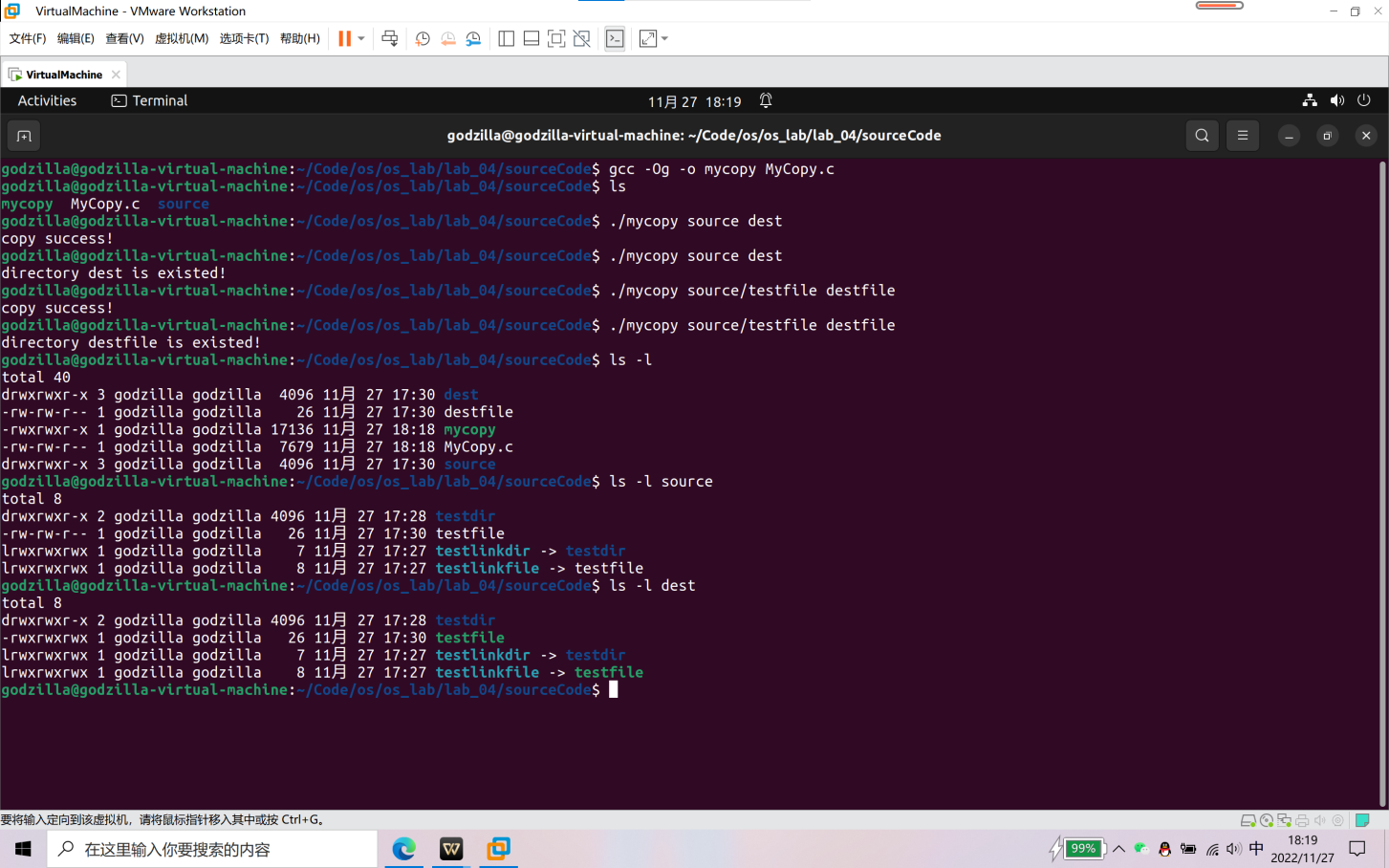
调用lstat()函数获得源文件的属性

若为目录，则调用copyDirectory()函数

若为链接文件，调用copyLinkFile()函数

若为普通文件,调用copyFile()函数

若为其他文件，直接返回-1

四、实验结果及分析

成功实现文件、目录与链接文件的复制，且会检查目标文件是否存在

五、实验收获与体会

通过本次实验，对于linun系统文件\目录属性有了一定的了解，同时对于链接文件的形成有了更深入的领会，其链接信息是相对路径指向的源文件。同时，我能够使用Linux系统进行文件复制的API,系统调用与数据结构来实现目录文件的递归复制。

附录：程序清单及说明

源代码路径sourceCode/MyCopy.c

录制视频 lab\_04.mp4

实验报告 实验报告.docx