shell程序设计

课程名称： Linux应用技术基础

实验类型：综合型

实验项目名称： shell程序设计

姓名： 吴梦堉

专业： 计算机科学与技术

学号：3180105091

电子邮箱： 995862798@qq.com

实验日期： 2020.6.14

一、实验环境

操作系统：Windows 10 家庭中文版 64位操作系统，基于x64的处理器

处理器： Intel(R) Core(TM)i7-8750H CPU 2.20GHz 2.21 GHz

内存(RAM)： 16.0GB

Linux版本： Ubuntu-1904

二、实验内容和结果及分析

1、编写一个shell脚本程序，它带一个命令行参数，这个参数是一个文件名。如果这个文件是一个普通文件，则打印文件所有者的名字和最后的修改日期。如果程序带有多个参数，则输出出错信息。

code

```
#!/bin/bash
if [ $2 ];then  #more than one parameter
    echo "Too many parmeters, please input exactly one parameter!"
elif [ $1 ];then
    if test -f "$1";then #get input
        ownername=$(ls -l $1 | awk '{print $3}') #get name
        month=$(ls -l $1 | awk '{print $6}') #get month
        day=$(ls -l $1 | awk '{print $7}') #get day
        time=$(ls -l $1 | awk '{print $8}') #get time
        #output
        echo "Filename:$1"
        echo "Owner's name:$ownername"
        echo "Modify time:$month $day $time"
    else    #not a file
        echo "$1 is not a ordinary file"
    fi
else
```

```
    echo "You should input one parameter!"
fi
```

test:

创建test文件和dir文件夹

```
ydream@ydream:~$ cd lab3
ydream@ydream:~/lab3$ touch test
ydream@ydream:~/lab3$ mkdir dir
```

以依次通过没有参数、一个文件参数、一个目录参数、两个参数来测试sh

```
ydream@ydream:~/lab3$ ./1.sh
You should input one parameter!
ydream@ydream:~/lab3$ ./1.sh test
Filename:test
Owner's name:ydream
Modify time:6月 17 20:51
ydream@ydream:~/lab3$ ./1.sh dir
dir is not a ordinary file
ydream@ydream:~/lab3$ ./1.sh test dir
Too many parmeters, please input exactly one parameter!
ydream@ydream:~/lab3$
```

如图 获得了预期的结果。

2、 编写shell程序，统计指定目录下的普通文件、子目录及可执行文件的数目，统计该目录下所有普通文件字节数总和，目录的路径名字由参数传入。

code

```
#!/bin/sh
echo "normal files:" `find $1 -type f | wc -l` #normal files
echo "subdirectory:"`find $1 -type d | wc -l` #subdirectory
echo "executable files:" `find $1 -type f -executable | wc -l` #executable
files
num=0
for file_name in `ls $1` #for each file
do
    file=$1"/"$file_name #get the path
    if [ -f $file ] #if it is an ordinary file
        then
            ch=$(cat $file | wc -c) #get the number of  chars
            num=$(($num+$ch))
    fi
done
echo "total char num: $num"
```

```
ydream@ydream:~/lab3$ ./2.sh ~/lab3
normal files: 3
subdirectory:2
executable files: 2
total char num: 1119
```

在lab3中实验，可以看到结果正确。

3、编写一个shell 脚本，输入一个字符串，忽略（删除）非字母后，检测该字符串是否为回文(palindrome)。对于一个字符串，如果从前向后读和从后向前读都是同一个字符串，则称之为回文串。例如，单词“mom”，“dad”和“noon”都是回文串。

```
#!/bin/sh
echo -n "Please input the string:"
read line     ##get input
str=`echo $line | tr -c -d [:alpha:] `    ##delete other chars
reverse=`echo $str | rev`
if [ $str = $reverse ];     ##check if int the same
then
    echo "$str is palindorme."
else
    echo "$str isn't palindorme"
fi
```

```
ydream@ydream:~/lab3$ ./3.sh
Please input the string:mom
mom is palindorme.
ydream@ydream:~/lab3$ ./3.sh
Please input the string:mo1m
mom is palindorme.
ydream@ydream:~/lab3$ ./3.sh
Please input the string:avd23?fd
avdfd isn't palindorme
ydream@ydream:~/lab3$
```

尝试了几组测试结果，成功忽略了非字母，并且判断回文结果正确。

4、编写一个shell脚本，把当前目录下文件大小大于100K的文件全部移动到~/tmp/ 目录下。

```
#!/bin/sh
find ./ -type f -size +100k -exec mv {} ~/tmp/ \;  ##find语句来实现
```

```
ydream@ydream:~/lab3$ cd ~
ydream@ydream:~$ mkdir tmp
ydream@ydream:~$ cd lab3
ydream@ydream:~/lab3$ ls -l
total 324
-rwxrw-r-- 1 ydream ydream    641 6月  17 21:11  1.sh
-rwxrw-r-- 1 ydream ydream    478 6月  17 21:30  2.sh
-rwxrw-r-- 1 ydream ydream    232 6月  17 21:46  3.sh
-rwxrw-r-- 1 ydream ydream     58 6月  17 21:52  4.sh
-rw-rw-r-- 1 ydream ydream 175104 6月  17 21:54 '实验2 shell命令.doc'
drwxrwxr-x 2 ydream ydream   4096 6月  17 20:51  dir
-rw-rw-r-- 1 ydream ydream 134341 6月  17 21:54 'lab1 研究报告.docx'
-rw-rw-r-- 1 ydream ydream      0 6月  17 20:51  test
ydream@ydream:~/lab3$ ./4.sh
ydream@ydream:~/lab3$ ls -l
total 20
-rwxrw-r-- 1 ydream ydream  641 6月  17 21:11 1.sh
-rwxrw-r-- 1 ydream ydream  478 6月  17 21:30 2.sh
-rwxrw-r-- 1 ydream ydream  232 6月  17 21:46 3.sh
-rwxrw-r-- 1 ydream ydream   59 6月  17 21:55 4.sh
drwxrwxr-x 2 ydream ydream 4096 6月  17 20:51 dir
-rw-rw-r-- 1 ydream ydream    0 6月  17 20:51 test
ydream@ydream:~/lab3$ cd ~/tmp
ydream@ydream:~/tmp$ ls
'实验2 shell命令.doc'  'lab1 研究报告.docx'
```

创建好tmp后，我们执行shell后发现两个大于100k的文件被移动到了/home/ydream/tmp下

5、编写一个实现文件备份和同步的shell脚本程序dirsync。程序的参数是两个需要备份同步的目录，如:

$dirsync\ /dir1\ /dir2\ \#\ /dir1$为源目录，$/dir2$为目标目录

dirsync程序实现两个目录内的所有文件和子目录（递归所有的子目录）内容保持一致。程序基本功能如下。

1) 备份功能：目标目录将使用来自源目录的最新文件，新文件和新子目录进行升级，源目录将保持不变。dirsync程序能够实现增量备份。

2) 同步功能：两个方向上的旧文件都将被最新文件替换，新文件都将被双向复制。源目录被删除的文件和子目录，目标目录也要对应删除。

代码1

代码1通过比较文件更改时间来进行，子目录使用递归进行更新备份

```
#!/bin/bash

# dirsync.sh
# Two modes:
# 1. back-up
# 2. synchronize

# Judge the input parameter
if [ $# -ne 2 ] #judge the number of parameter
then
        echo "program: $0 needs one prameter."
        exit 1
fi
if [[ ! -d $1 || ! -d $2 ]] #check if they are dictionary
then
    echo "parameters must be dirctory"
    exit 1
fi

# The definitions of functions used

# To update file.
# @parameters : source file, destination path and source path

function update_file {
    new_file=$2\/$1
    source_file=$3\/$1
    if [`! -f $new_file`] && [ `stat -c %Y $new_file` -gt `stat -c %Y
$source_file` ]
    then
        cp -fp $source_file $new_file
        echo "File "$source_file" has been copied"
        cnt_update=$cnt_update+1    # cnt the number
        return 1
    fi
    return 0
}
```

```
37
38  # To delete the file that doesn't exist in the given source file.
39  # @Parameters : source file, destination path and source path
40  # Use a dictionary /tmp to replace rm
41  function delete_file {
42      if test -f $2\/$1
43      then
44          :
45      else
46          mv $3\/$1 \/tmp
47          echo "File "$3\/$1" has been deleted"
48          cnt_delete=$cnt_delete+1 # cnt the number
49          return 1
50      fi
51      return 0
52  }
53
54  # To update a directory.
55  # @Parameters : source dir, destination path and source path
56
57  function update_dir {
58      cnt=0
59      if test -d $2\/$1
60      then
61          for item in $(ls $3\/$1)
62          do
63              if test -d $3\/$1\/$item
64              then
65                  update_dir $item $2\/$item $3\/$1 # recursively update
66              elif test -f $3\/$1\/$item
67              then
68                  update_file $item $2\/$1 $3\/$1 # check the file
69                  if (( $? == 1 ))
70                  then
71                      cnt=$cnt+1
72                  fi
73              fi
74          done
75      else
76          cp -fpr $3\/$1 $2\/$1
77          echo "Directory "$3\/$1" has been copied"
78      fi
79  }
80
81  # To delete the directory that doesn't exist in the given source file.
82  # @Parameters : source dir($1), destination path($2) and source path($3)
83  # Use a dictionary /tmp to replace rm
84  function delete_dir {
85      cnt=0
86      if test -d $2\/$1
87      then
88          for item in $(ls $3\/$1)
89          do
90              if test -d $3\/$1\/$item # check if it is a dir
91              then
92                  delete_dir $item $2\/$item $3\/$1 # recursively delete
93                  continue
94              fi
```

```
            if test -f $3\/$1\/$item # check if it is a file
            then
                delete_file $item $2\/$1 $3\/$1 # delete a file
                if (( $? == 1 ))
                then
                    cnt=$cnt+1
                fi
                continue
            fi
        done
    else
        mv $3\/$1 \/tmp # safely remove
        echo "Directory "$3\/$1" has been deleted"
    fi
}

# Main

source_dir=${1:-./}
destination_dir=${2:-./}
declare -i cnt_update=0 # counter of update
declare -i cnt_delete=0 # counter of deletion
echo "--------------------------------Welcome To Dirsync-----------------
----------------"
echo -e "Source directory       : \033[33m ${source_dir} \033[0m"
echo -e "Destination directory  : \033[33m ${destination_dir} \033[0m"
echo "mode1 Backup : ${destination_dir} will be updated with the file from
${source_dir}"
echo "mode2 Synchronize: all the old files will be updated."
read -p "Please input mode and press Enter (1/2) : " mode


if [ $mode = 1 ]
then
    # back-up
    echo -e "\033[31mbackup... \033[0m"
    # Note that the '/' is removed
    source_dir=${source_dir%/}
    destination_dir=${destination_dir%/}
    for item in $(ls ${source_dir}\/)
    do
        tmp_path=$source_dir\/$item
        if test -d $tmp_path
        then
            update_dir $item $destination_dir $source_dir
        elif test -f $tmp_path
        then
            update_file $item $destination_dir $source_dir
        fi
    done

    echo -e "\033[32mBackup done! \033[0m"
    echo "Totally updated ${cnt_update} files."
elif [ $mode = 2 ]
then
    # Synchronize

    echo -e "\033[31mSynchronize... \033[0m"
```

```
    source_dir=${source_dir%/}
    destination_dir=${destination_dir%/}

    tmp=$source_dir
    source_dir=$destination_dir
    destination_dir=$tmp

    for item in $(ls ${source_dir}\/)
    do
        tmp_path=$source_dir\/$item
        if test -d $tmp_path
        then
            delete_dir $item $destination_dir $source_dir
        elif test -f $tmp_path
        then
            delete_file $item $destination_dir $source_dir
        fi
    done

    for item in $(ls ${source_dir}\/)
    do
        tmp_path=$source_dir\/$item
        if test -d $tmp_path
        then
            update_dir $item $destination_dir $source_dir
        elif test -f $tmp_path
        then
            update_file $item $destination_dir $source_dir
        fi
    done

    tmp=$source_dir
    source_dir=$destination_dir
    destination_dir=$tmp

    for item in $(ls ${source_dir}\/)
    do
        tmp_path=$source_dir\/$item
        if test -d $tmp_path
        then
            update_dir $item $destination_dir $source_dir
        elif test -f $tmp_path
        then
            update_file $item $destination_dir $source_dir
        fi
    done
    # echo some relavant information
    echo -e "\033[32mSynchronize done!\033[0m"
    echo "Totally updated ${cnt_update} files."
    echo "Totally deleted ${cnt_delete} files."
else
    echo "Invalid input!"
fi

```

- 1）备份功能

```
ydream@ydream:~/lab3$ ls dir/ newDir/
dir/:
1  2  3

newDir/:
ydream@ydream:~/lab3$ ./5-1.sh dir newDir
--------------------------------Welcome To Dirsync--------------------------------
Source directory       :  dir
Destination directory  :  newDir
mode1 Backup : newDir will be updated with the file from dir
mode2 Synchronize: all the old files will be updated.
Please input mode and press Enter (1/2) : 1
backup...
File dir/1 has been copied
File dir/2 has been copied
File dir/3 has been copied
Backup done!
Totally updated 3 files.
ydream@ydream:~/lab3$ ls dir/ newDir/
dir/:
1  2  3

newDir/:
1  2  3
ydream@ydream:~/lab3$ touch dir/4
ydream@ydream:~/lab3$ vim dir/3
ydream@ydream:~/lab3$ ydream@ydream:~/lab3$ ./5-1.sh dir newDir
--------------------------------Welcome To Dirsync--------------------------------
Source directory       :  dir
Destination directory  :  newDir
mode1 Backup : newDir will be updated with the file from dir
mode2 Synchronize: all the old files will be updated.
Please input mode and press Enter (1/2) : 1
backup...
File dir/3 has been copied
File dir/4 has been copied
Backup done!
Totally updated 2 files.
ydream@ydream:~/lab3$ ls dir/ newDir/
dir/:
1  2  3  4

newDir/:
1  2  3  4
```

```
ydream@ydream:~/lab3$ ls dir/ dir/child/ newDir/
dir/:
1  2  3  child

dir/child/:
1

newDir/:
ydream@ydream:~/lab3$ ./5-1.sh dir newDir
--------------------------------Welcome To Dirsync--------------------------------
Source directory       :  dir
Destination directory  :  newDir
mode1 Backup : newDir will be updated with the file from dir
mode2 Synchronize: all the old files will be updated.
Please input mode and press Enter (1/2) : 1
backup...
File dir/1 has been copied
File dir/2 has been copied
File dir/3 has been copied
Directory dir/child has been copied
Backup done!
Totally updated 3 files.
ydream@ydream:~/lab3$ ls dir/ dir/child/ newDir/
dir/:
1  2  3  child

dir/child/:
1

newDir/:
1  2  3  child
```

如图1

dir为源目录，newDir为新目录

dir中包含1 2 3 三个文件，newDir中没有文件

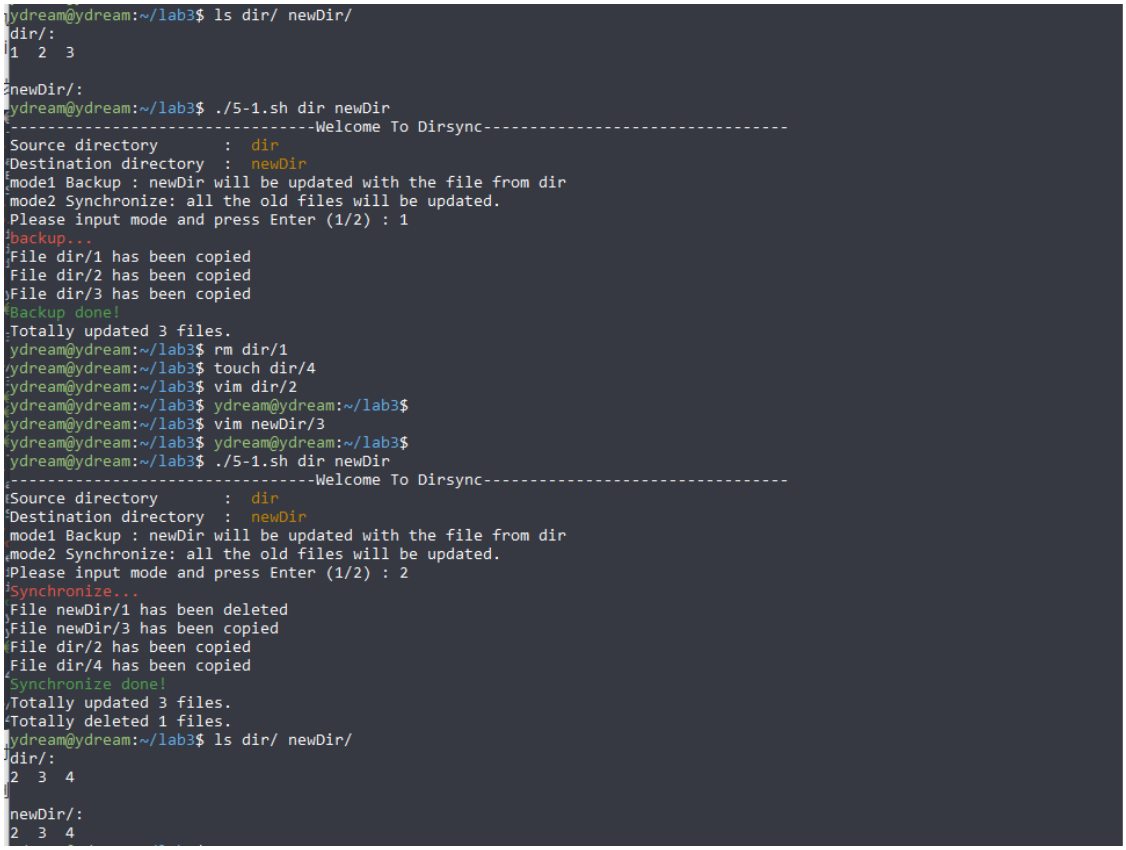
执行shell，选择模式1（备份模式），可以看到更新了1 2 3三个文件出现在了newDir中

然后我们在dir目录下更改文件3，并且新建文件4

再次执行shell，选择模式1（备份模式）， 根据log，我们发现只有3 4被更新了，查看发现1 2 3 4 均出现在newDir中

图2展示了子目录的情况。

- 2）同步功能



首先对新目录进行备份，使用模式1

接着删除dir/1 更改dir/2 更改newDir/3 新建dir/4

运行shell，使用模式2

根据log，可以看到，

newDir1 被删除，说明源目录中文件被删除，目标目录下文件也被删除

newDir3 被复制到dir3 中 dir2被复制到newDir2中，说明双向可以同步文件

dir中新建的文件4 被复制到newDir中，说明双向可以更新文件

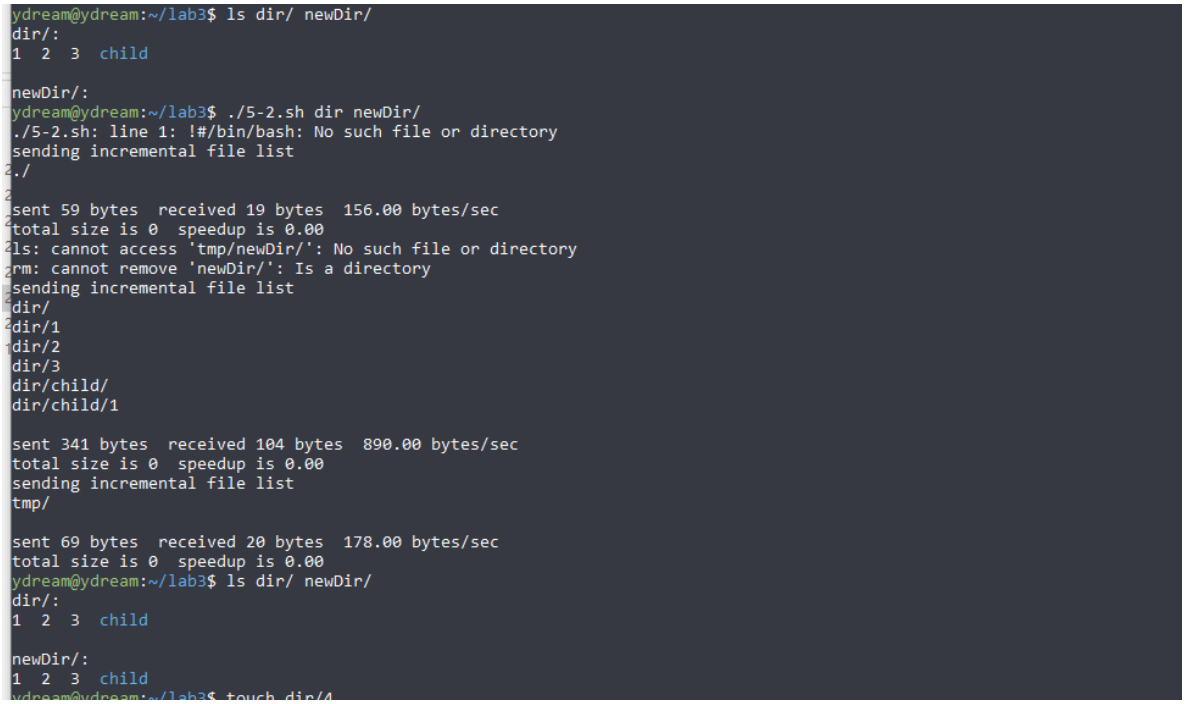
代码2

代码2使用了rsync命令

```
!#/bin/bash
mkdir tmp   ##使用临时文件夹
rsync -av $2 tmp   ##使用rsync来同步
for item in `ls tmp/$2`   ##遍历移动
do
    mv -f tmp/$2/$item tmp
done
rm -rf tmp/$2
rm -f $2    ##删除

rsync -av $1 $2   ##以下内容同上，通过一个临时文件夹来实现同步

for item in `ls $2/$1`
do
    mv -f $2/$1/$item $2
done
```

```
rm -rf $2/$1
rsync -av tmp $1

for item in `ls $1/tmp`
do
    mv -f $1/tmp/$item $1
done
rm -rf $1/tmp
rm -rf tmp
```

展示



```
ydream@ydream:~/lab3$ touch dir/4
ydream@ydream:~/lab3$ rm dir/1
ydream@ydream:~/lab3$ vim dir/2
ydream@ydream:~/lab3$ ydream@ydream:~/lab3$ vim newDir/3
ydream@ydream:~/lab3$ ./5-2.sh dir newDir/
./5-2.sh: line 1: !#/bin/bash: No such file or directory
sending incremental file list
./
1
2
3
child/
child/1

sent 342 bytes  received 103 bytes  890.00 bytes/sec
total size is 6  speedup is 0.01
ls: cannot access 'tmp/newDir/': No such file or directory
rm: cannot remove 'newDir/': Is a directory
sending incremental file list
dir/
dir/2
dir/3
dir/4
dir/child/
dir/child/1

sent 350 bytes  received 104 bytes  908.00 bytes/sec
total size is 4  speedup is 0.01
mv: cannot move 'newDir//dir/child' to 'newDir/child': Directory not empty
sending incremental file list
tmp/
tmp/1
tmp/2
tmp/3
tmp/child/
tmp/child/1

sent 350 bytes  received 104 bytes  908.00 bytes/sec
total size is 6  speedup is 0.01
mv: cannot move 'dir/tmp/child' to 'dir/child': Directory not empty
ydream@ydream:~/lab3$ ls dir/ newDir/
dir/:
1  2  3  4  child

newDir/:
1  2  3  4  child
```

过程同代码1，不赘述

三、感想和讨论

这个实验中，从简单到复杂，一共写了5个shell脚本，让我对于shell编程有了更深的理解，从简单的几个命令拼凑，到分支结构，到循环结构。在最后一个题目中更是使用了递归函数。在编程的过程中，我拿shell编程和c语言编程做类比，在逻辑上并没有遇到很大的困难，不过shell的语法十分严格，并且没有较好的编辑器来及时检查错误，或者代码补全，以致于经常写完以后没有办法运行。对于我的细心有很大的磨砺。

遇到的问题：

1）在windows环境下编写shell以后通过ftp保存到linux，运行会报错`/bin/bash^M: bad interpreter: No such file or directory`

查询后发现，原因在于windows下shell是dos格式，linux是unix格式，换行符存在差异。解决方案是用vim打开shell，输入`set ff=unix`命令就可以解决

2）在linux直接运行shell会报错`Permission denied`这是因为sh没有权限访问`/bin/bash`

只需要通过chmod为shell增加权限即可

3）在实验五中，理解题意花了挺长的时间。由于之前接触过rsync用来同步windows和远程linux服务器的文件，因此首先想到了直接使用rsync命令来完成实验。但是又觉得老师出题的目的应该是想要我们自己去写一个shell来完成这些功能。想了很久，并且与同学讨论之后，想到了使用时间来判断先后，通过直接复制文件来进行同步。当然，最后实现的功能是远远不及直接使用rsync的。并且我写的shell，只能粗暴的根据时间来进行复制，如果文件都有更新，则没办法正确的同步。最好还是使用git等源代码管理工具或者rsync等来进行多端同步。