浙江大学实验报告

课程名称：Linux 应用技术基础 实验类型：综合型

实验项目名称：实验三 程序设计

学生姓名：_____ 专业：信息安全 学号：_____

电子邮件地址：_____

实验日期：2020 年 6 月 16 日

一、 实验环境

计算机配置：

处理器: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (8 CPUs), ~1.8GHz

内存: 8192MB RAM

显卡名称: Intel(R) UHD Graphics 620

芯片类型: Intel(R) UHD Graphics Family

操作系统：

Windows 10 家庭中文版 64-bit (10.0, Build 18363)

Linux 版本：

Ubuntu 18.04.4 LTS

二、 实验内容和结果及分析

1. （15 分）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。

1.sh

```
#! /bin/bash
if test $# -ne 1 #not a parameter
  then
    echo "there must be 1 argument" #print the error info
    exit 1
fi
if test -f "$1" #ordinary file
  then
    filename=$1 #file type is ordinary file
    set -- $(ls -l $filename) #find the info
    echo "file owner: $3" #print the file owner
    echo "last edit time: $6 $7 $8" #print the last edit time
    exit 0
fi
echo "$1: argument must be an ordinary file!" #error if not ordinary file
exit 0
```

```
xx@xx-virtual-machine:~/temp$ ./1.sh 1.sh
file owner: xx
last edit time: 6月 20 22:08
xx@xx-virtual-machine:~/temp$ ./1.sh
there must be 1 argument
xx@xx-virtual-machine:~/temp$ ./1.sh 1
1: argument must be an ordinary file!
```

2. （15 分）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。

2.sh

```
#! /bin/bash
if (test $# -ne 1) || !(test -e $1) || !(test -d $1) #check whether there is a parameter or existing directory
then
  echo "there must be 1 argument"
  exit 1
fi
com=$(find $1 -type f|wc -l) #find the number of common files
echo "ordinary files are: $com"
dir=$(find $1 -type d|wc -l) #find the number of directory files
echo "directory files are: $dir"
exe=$(find $1 -executable -type f|wc -l) #find the number of executable files
echo "executable files are: $exe"
bit=$(find $1 -type f|wc -c) #find the number of bits of common files
echo "bits of common files are: $bit"
exit 0
```

```
xx@xx-virtual-machine:~/temp$ ./2.sh d1
ordinary files are: 1
directory files are: 1
executable files are: 0
bits of common files are: 8
xx@xx-virtual-machine:~/temp$ ./2.sh ../temp
ordinary files are: 6
directory files are: 7
executable files are: 3
bits of common files are: 116
xx@xx-virtual-machine:~/temp$ ./2.sh
there must be 1 argument
```

3. （15 分）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。

3.sh

```
#! /bin/bash
echo -n "input string: "
read line #read in string
str=$(echo $line | tr -c -d [:alpha:]) #-c to select the string other than alphabet and -d to delete
rev=$(echo $str | rev) #reverse the string
if [[ $str == $rev ]] #check if the two is the same
then
  echo "$str is palindorme" #equal and print palindorme
else
  echo "$str is not palindorme" #unequal and print not palindorme
fi
exit 0
```

```
xx@xx-virtual-machine:~/temp$ ./3.sh
input string: 123abccba
abccba is palindorme
xx@xx-virtual-machine:~/temp$ ./3.sh
input string: abcbd
abcbd is not palindorme
```

4. （15 分）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。

4.sh

```
#! /bin/bash
find . -size +100k -exec mv {} /tmp/ \;
echo "done."
ls -al /tmp
```

```
xx@xx-virtual-machine:~/temp$ ./4.sh
done.
总用量 100
drwxrwxrwt 25 root root 4096 6月  21 00:09  .
drwxr-xr-x 25 root root 4096 6月  17 02:40  ..
-rw-------  1 xx   xx      0 6月  20 19:41  config-err-Ad6ZeL
srwxr-xr-x  1 xx   xx      0 6月  20 20:17  ELiveClient_LiveCenter1000
drwxrwxrwt  2 root root 4096 6月  20 19:41  .font-unix
drwxrwxrwt  2 root root 4096 6月  20 19:41  .ICE-unix
drwx------  2 xx   xx   4096 6月  20 21:28  mozilla_xx0
-rw-r--r--  1 xx   xx      0 6月  20 20:17  __#_PROME_INSTANCE_EXIST_KEY_#__.lock
srwxr-xr-x  1 xx   xx      0 6月  20 20:17  QingBanGong_LocalPush1000
srwxr-xr-x  1 xx   xx      0 6月  20 20:17  _#_QingBanGong_Server_#_1000
-rw-r--r--  1 xx   xx      0 6月  20 20:17  __#_QING_EXIST_MUTEX_KEY_#__1000.lock
-rw-r--r--  1 xx   xx      0 6月  20 20:17  __#_QING_MUTEX_KEY_#__1000.lock
-rw-r-----  1 xx   xx      0 6月  20 20:17  qipc_sharedmemory_UPDLogFileZMutxNamefa9ef7dcc10a811d72a9a600bc8bcdc0a89f1a5b
-rw-r-----  1 xx   xx      0 6月  20 20:17  qipc_systemsem_UPDLogFileZMutxNamefa9ef7dcc10a811d72a9a600bc8bcdc0a89f1a5b
drwx------  3 root root 4096 6月  20 19:54  snap.shfmt
drwx------  2 xx   xx   4096 6月  20 19:41  ssh-OzXKSJmaWTP6
drwx------  3 root root 4096 6月  20 19:41  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-bolt.service-87569K
drwx------  3 root root 4096 6月  20 19:41  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-colord.service-MR5TsX
drwx------  3 root root 4096 6月  20 19:42  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-fwupd.service-K5UrOr
drwx------  3 root root 4096 6月  20 19:41  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-ModemManager.service-qxLn9w
drwx------  3 root root 4096 6月  20 19:41  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-rtkit-daemon.service-DE49YK
drwx------  3 root root 4096 6月  20 19:41  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-systemd-resolved.service-phJhDA
drwx------  3 root root 4096 6月  20 19:41  systemd-private-5fde8e5afe9f4cc882f0a87566fc2d47-systemd-timesyncd.service-oCZ1g3
drwx------  2 xx   xx   4096 6月  20 20:07  Temp-53794a44-27da-4b2a-b40e-ec14f7b6090c
drwx------  2 xx   xx   4096 6月  20 20:07  Temp-dc507c66-933f-4c1a-871d-24f700f2d7a1
drwxrwxrwt  2 root root 4096 6月  20 19:41  .Test-unix
srwxr-xr-x  1 xx   xx      0 6月  20 20:17  _Thrift_Qing_IPC_1000
srwxr-xr-x  1 xx   xx      0 6月  20 20:17  _Thrift_Qing_IPC_2_1000
drwxrwxrwt  2 root root 4096 6月  20 19:41  VMwareDnD
drwx------  2 root root 4096 6月  20 19:41  vmware-root
drwx------  2 root root 4096 6月  20 19:41  vmware-root_637-3979708642
drwx------  2 xx   xx   4096 6月  20 19:41  vmware-xx
drwx------  2 xx   xx   4096 6月  20 19:48 'VSCode Crashes'
drwxr-xr-x  2 xx   xx   4096 6月  20 21:47  vscode-typescript1000
drwxrwxrwt  2 root root 4096 6月  20 19:41  .X11-unix
drwxrwxrwt  2 root root 4096 6月  20 19:41  .XIM-unix
```

5. （30 分）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。
3) 其它功能自行添加设计

5.sh

```
#! /bin/bash
#Description: dirsync ~|dir1 ~|dir2 # ~|dir1为源目录，~|dir2为目标目录
#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。
#Author: xx
#Version: 1.0
#CreateTime: 2020-06-21
echo "Enter 'dirsync /dir1 /dir2'  where /dir1 is source directory and /dir2 is target directory" #input form if dir2 exists
read -a dir -p "dirsync " #read in dir1 dir2

#增量备份：根据源目录更新
source_dir="${dir[0]}/*"
#-r：递归复制，用于复制目录；-u：若目标文件比源文件有差异，则使用该选项可以更新目标文件，此选项可用于对文件的升级和备用;-v：显示指令执行过程。
cp -r -u -v ${source_dir} ${dir[1]};

#同步：双向替换复制
target_dir="${dir[1]}/*"
cp -r -u -v ${target_dir} ${dir[0]};

#删除目标目录中原目录中不存在的子目录和文件
#比较备份文件和源文件
files=$(ls ${source_dir})
for file in $files
do
  filename=$(basename $file)
  file_desc=${dir[1]}/${filename}

  diff ${file} ${file_desc} 1>/dev/null 2>&1 && result=0 || result=1

  if [ "$result" == 1 ];then
    rm -rf $filename
  fi
done
echo "dirsync ${dir[0]} ${dir[1]} done." #print completion info
exit 0
```

```
xx@xx-virtual-machine:~/temp$ ./5.sh
Enter 'dirsync /dir1 /dir2'  where /dir1 is source directory and /dir2 is target directory
dirsync /home/xx/temp /home/xx/temp1
'/home/xx/temp/1.sh' -> '/home/xx/temp1/1.sh'
'/home/xx/temp/2.sh' -> '/home/xx/temp1/2.sh'
'/home/xx/temp/3.sh' -> '/home/xx/temp1/3.sh'
'/home/xx/temp/4.sh' -> '/home/xx/temp1/4.sh'
'/home/xx/temp/5.sh' -> '/home/xx/temp1/5.sh'
'/home/xx/temp/d1' -> '/home/xx/temp1/d1'
'/home/xx/temp/d1/1.sh' -> '/home/xx/temp1/d1/1.sh'
'/home/xx/temp/d2' -> '/home/xx/temp1/d2'
'/home/xx/temp/d2/smallFile.soft' -> '/home/xx/temp1/d2/smallFile.soft'
'/home/xx/temp/d2/smallFile.hard' -> '/home/xx/temp1/d2/smallFile.hard'
'/home/xx/temp/d2/newFile.hard' -> '/home/xx/temp1/d2/newFile.hard'
'/home/xx/temp/d3' -> '/home/xx/temp1/d3'
'/home/xx/temp1/1.sh' -> '/home/xx/temp/1.sh'
'/home/xx/temp1/2.sh' -> '/home/xx/temp/2.sh'
'/home/xx/temp1/3.sh' -> '/home/xx/temp/3.sh'
'/home/xx/temp1/4.sh' -> '/home/xx/temp/4.sh'
'/home/xx/temp1/5.sh' -> '/home/xx/temp/5.sh'
'/home/xx/temp1/d1/1.sh' -> '/home/xx/temp/d1/1.sh'
已删除'/home/xx/temp/d2/smallFile.soft'
'/home/xx/temp1/d2/smallFile.soft' -> '/home/xx/temp/d2/smallFile.soft'
'/home/xx/temp1/d2/smallFile.hard' -> '/home/xx/temp/d2/smallFile.hard'
'/home/xx/temp1/Temp-dc507c66-933f-4c1a-871d-24f700f2d7a1' -> '/home/xx/temp/Temp-dc507c66-933f-4c1a-871d-24f700f2d7a1'
dirsync /home/xx/temp /home/xx/temp1 done.
```

三、　讨论、心得（必填）（10 分）

通过本次实验写一些简短的脚本，加深了对 shell 脚本的基本概念的理解，

学会了编写 Bourne shell 脚本程序的方法。实验过程中遇到的主要问题有：

1. 常常忘记直接执行.sh 文件会权限不够，因为只有读写而没有执行权限，因此每次必须通过命令 chmod +x 1.sh 为文件加上执行权限。

2. 还有对 shell 脚本的对齐格式等不是很了解，在网上寻找自动对齐方法时发现 VS Code 可以安装 shell-format：https://blog.csdn.net/zz153417230/article/details/103176747。但是网上对这个插件的安装介绍不多，配置中出现了一些问题没有解决，后来直接通过它依赖的 shfmt，配置后执行 shfmt -l -w 1.sh 就可以实现.sh 文件的自动对齐。

3. 脚本的注释格式等没有找到非常规范的参考，在网上寻找资料的时候找到了一个比较有意思的为脚本自动添加脚本头的脚本：https://blog.csdn.net/a595364628/article/details/53019098。

4. 变量赋值时" "、' '、[ ]、( )的作用各不相同，比如以单引号' '包围变量的值时，单引号里面是什么就输出什么，即使内容中有变量和命令（命令需要反引起来）也会把它们原样输出。这种方式比较适合定义显示纯字符串的情况，即不希望解析变量、命令等的场景。以双引号" "包围变量的值时，输出时会先解析里面的变量和命令，而不是把双引号中的变量名和命令原样输出。这种方式比较适合字符串中附带有变量和命令并且想将其解析后再输出的变量定义。variable=`command`和 variable=$(command)是一样的，而且要注意赋值号=左右不能有空格。

5. 最后一个程序设计题因为不熟悉一些变量读入和转换之类的操作浪费了较多时间。在思考这道题的解决方式时看到了 rsync+inotify 实现实时文件自动同步的方案 https://www.linuxidc.com/Linux/2017-12/149355.htm 以及多机器实时同步文件神器 lsyncd 这些早就造好的轮子，相比 cp 命令各有优势，也值得借鉴和参考，同时对 cp 命令的备份作用也有了进一步的体会。