Shell Programming

Sunday, 4 April 2021 18:20

1.

```
#!/bin/bash

#Display a report showing the full name of all the users currently connected, and the number of processe
s belonging to each of them

cat who.fake | \
        awk '{print $1}' | \
        while read U; do
                n=`grep -E "^$U " ps.fake | wc -l`
                echo "$U $n"
        done
```

```
bradu 2
horea 2
rares 3
```

2.

```
#Find recursively in a directory all ".c" files having more than 500 lines. Stop after finding 2 such fi
les.
c=0
for F in `find $1 -type f -name "*.c"`; do
        if [ $c -eq 2 ]; then
                break
        fi
        n=`cat $F | wc -l`
        if [ $n -gt 500 ]; then
                c=`expr $c + 1`
                echo $F
        fi
        done
```

```
alinaelenabrinza@DESKTOP-S6C1A6M:~/shell-prog$ ./2.sh dir
dir/11.c
dir/d/a/13.c
```

3.

```
#!/bin/bash
#Find recursively in a directory, all the files with the extension ".log" and sort their lines (replace
the original file with the sorted content).

#cat sortex.log | sort > sortex1.log | \
 #      rm sortex.log | mv sortex1.log sortex.log
for F in `find $1 -type f -name "*.log"`; do
       cat $F | sort > temp.log | rm $F | mv temp.log $F
done
```

Should be sorted in the files.

4. Gresit

7. Find recursively in a given directory all the symbolic links, and report those that point to files/directories that no longer exist. Use option -L to test if a path is a symbolic link, and option -e to test if it exists (will return false if the target to which the link points does not exist)
Gasiti recursiv intr-un director dat toate legaturile simbolice si raportati care din ele sunt legate de fisiere/directoare care nu exista. Folositi optiunea -L de la test pentru a verifica daca un string este link symbolic si optiunea -e casa verificati daca este valid (test va returna false daca fisierul/directorul referit de legatura simbolica nu exista)

```
#!/bin/bash

for link in $(find "$1" -type l); do
    if [ ! -e "$link" ]; then
        echo "Link $link is not valid"
    fi
done
```

```bash
#!/bin/bash

#Find recursively in a given directory all the symbolic links, and report those that point to files/dire
ctories that no longer exist. Use operator -L to test if a path is a symbolic link, and operator -e to t
est if it exists (will return false if the target to which the link points does not exist)

for F in `find $1`;  do
        if [ -L $F ]; then
                echo "File $F is a symbolic link"
                if [ -e $F ]; then
                        echo "File $F does not exist"
                fi
        fi
done
#can combine the test for symbolic link and existence in just one line
```

NU 6 OK

5.

```bash
#!/bin/bash

#       Write a script that receives dangerous program names as command line arguments. The script will
monitor all the processes in the system, and whenever a program known to be dangerous is run, the script
 will kill it and display a message.

if [ $# -eq 0 ]; then
        echo "Provide at least one name"
        exit 1
fi

while true; do
        for process in $@; do
                PIDs=""
                PIDs=$(ps -ef | awk '{print $8" "$2}' | grep -E "\<$process " | grep -E "danger" | awk '
{print $2}')
                if [ -n "$PIDs" ]; then
                        kill -9 $PIDs
                fi
        done
        sleep 3
done
```

6.

```bash
#!/bin/bash

#Find recursively in a directory, all the files that have write permissions for everyone. Display the
ir names, and the permissions before and after removing the write permission for everybody. You will
need to use chmod's symbolic permissions mode, instead of the octal mode we have used in class. The t
he chmod manual for details.
for F in `find $1 -type f`; do
        if [ -w $F ]; then
                echo `ls -l $F | awk '{print $1" "$9}'`
                chmod a-w $F
                echo `ls -l $F | awk '{print $1" "$9}'`
        fi
done
```

7.

```
#Consider a file containing a username on each line. Generate a comma-separated string with email
addresses of the users that exist. The email address will be obtained by appending "@scs.ubbcluj.r
o" at the end of each username. Make sure the generated string does NOT end in a comma.

if [ -z "$1" ]; then
        echo "Please provide one argument"
        exit 1
fi

if [ ! -f "$1" ]; then
        echo "Argument must be a file"
        exit 1
fi

result=""
for username in $(cat $1); do
        email=$(echo "$username" | sed -E "s/(.+)/\1@scs.ubbcluj.ro/")
        result="$email,$result"
done
result=$(echo $result | sed -E "s/(,$)//")
echo "$result"
```

```
alinaelenabrinza@DESKTOP-S6C1A6M:~/shell-prog$ ./7.sh usernames.txt
c@scs.ubbcluj.ro,b@scs.ubbcluj.ro,a@scs.ubbcluj.ro
```

8.

```
#       Display all the mounted file systems who are either smaller than than 1GB or have less tha
n 20% free space
cat df.fake | tail -n +2 | \
        while read f; do
                size=`echo $f | awk '{print $2}' | sed -E "s/(M$)//"`
                use=`echo $f | awk '{print $5}' | sed -E "s/(%$)//"`
                if [ $size -lt 1024 ] || [ $use -gt 80 ]; then
                        echo $f | awk '{print $6}'
                fi
        done
```

9. Fdupes -f <dir>

Horea

1. Write a bash script that calculates the sum of the sizes (in bytes) of all regular files in a folder given as a parameter.(use test to check if the folder exists and if a given file is a regular file)
Scrieti un script bash care calculeaza suma in octeti a tuturor fisierelor regulare dintr-un director dat ca argument.(folositi test ca sa verificati daca directorul dat exista si daca un fisier este fisier regular)

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "No parameters given"
    exit 1
fi

if [ ! -d "$1" ]; then
    echo "Parameter is not a folder"
    exit 1
fi

total=0
for item in $(ls "$1"); do
    f="$1/$item"
    if [ -f "$f" ]; then
        size=$(du -bs "$f" | awk '{print $1}')
        total=$((total+size))
    fi
done

echo "Total size of regular files from folder $1 is $total"
```

2. Write a script that reads filenames until the word "stop" is entered. For each filename, check if it is a text file and if it is, print the number of words on the first line.(Hint: test command to check if regular file; file command to check if text file)

```bash
#!/bin/bash

while true; do
        read -p "filename: " n
        if [ "$n" = "stop" ]; then
                break
        fi
        if [ -f $n ] && file $n | grep -q "text"; then
                p=`cat $n | awk -F'[: \t]+' '{if(NR==1)print NF}'`
                echo nr.words: $p
        else
                echo NOT TEXT FILE
        fi
done
```

ASCII

3. Write a script that receives as command line arguments pairs consisting of a filename and a word. For each pair, check if the given word appears at least 3 times in the file and print a corresponding message.
Scrieti un script bash care primeste ca argumente la linia de comanda perechi de nume de fisier si cuvant. Pentru fiecare pereche, afisati un mesaj daca in fisier cuvantul dat apare de cel putin 3 ori.

```bash
#!/bin/bash

if [ $# -lt 2 ]; then
    echo "Please provide at least 2 arguments"
    exit 1
fi

if [ $(($# % 2)) -eq 1 ]; then
    echo "You must provide an even number of arguments"
    exit 1
fi

while [ $# -gt 1 ]; do
    file=$1
    word=$2

    if [ ! -f "$file" ]; then
        echo "Name $file is not a file"
    else
        count=$(grep -E -o "\<$word\>" "$file" | wc -l)
        if [ $count -ge 3 ]; then
            echo "Word $word appears $count times in file $file"
        fi
    fi
    shift 2
done

if [ $# -eq 1 ]; then
    echo "Warning: final pair is incomplete"
fi
```

4. Write a bash script that sorts all files given as command line arguments descending by size.(first check if an argument is a file)
Scrieti un script bash care sorteaza descrescator dupa dimensiune toate fisierele date ca argumente la linia de comanda.(intai verificati daca un argument e fisier)

```bash
for f in $@; do
    if test -e $f; then
        v+=("$f")
    fi
done
du -b `echo ${v[@]}` |sort -n
```

5. Write a script that extracts from all the C source files given as command line arguments the included libraries and saves them in a file.(use the file command to check if a file is a C source file)
Scrieti un script care extrage bibliotecile incluse din toate fisierele sursa C date ca argumente la linia de comanda si le salveaza intr-un alt fisier. (verificati cu comanda file daca un fisier este sursa C)

```
for f in $@; do
        if file $f| grep -q -E "C source"; then
        cat $f|while read l;do
                echo $l|grep "^#include" >> save.txt
        done
        fi
done
```

6. Write a script that monitors the state of a given folder and prints a message when something changed.
Scrieti un script care monitorizeaza starea unui director dat ca argument si afiseaza un mesaj daca apar modificari in directorul dat.

```
#!/bin/bash

D=$1
if [ -z "$D" ]; then
  echo "ERROR: No directory provided for monitoring" >&2
  exit 1
fi

if [ ! -d "$D" ]; then
  echo "ERROR: Directory $D does not exist" >&2
  exit 1
fi

STATE=""
while true; do
    S=""
    for P in `find $D`; do
        if [ -f $P ]; then
          LS=`ls -l $P | shalsum`
          CONTENT=`shalsum $P`
        elif [ -d $P ]; then
          LS=`ls -l -d $P | shalsum`
          CONTENT=`ls -l $P | shalsum`
        fi
        S="$S\n$LS $CONTENT"
    done
    if [ -n "$STATE" ] && [ "$S" != "$STATE" ]; then
      echo "Directory state changed"
    fi
    STATE=$S
    sleep 1
done
```

7. Find recursively in a given directory all the symbolic links, and report those that point to files/directories that no longer exist. Use option -L to test if a path is a symbolic link, and option -e to test if it exists (will return false if the target to which the link points does not exist)
Gasiti recursiv intr-un director dat toate legaturile simbolice si raportati care din ele sunt legate de fisiere/directoare care nu exista. Folositi optiunea -L de la test pentru a verifica daca un string este link symbolic si optiunea -e casa verificati daca este valid (test va returna false daca fisierul/directorul referit de legatura simbolica nu exista)

```
#!/bin/bash

for link in $(find "$1" -type l); do
    if [ ! -e "$link" ]; then
        echo "Link $link is not valid"
    fi
done
```

8. Write a bash script that receives a folder name as argument. Find recursively in the folder the number of times each file name is repeated.
Scrieti un script bash care primeste un nume de director ca parametru. Cautati recursiv in director si numarati aparitiile fiecarui nume de fisier.

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "Please provide one argument"
    exit 1
fi

if [ ! -d "$1" ]; then
    echo "Argument must be a directory"
    exit 1
fi

find "$1" -type f | awk -F/ '{print $NF}' | sort | uniq -c
```

9. Calculate the average of all process ids in the system per user.
Calculati media id-urilor proceselor din sistem pentru fiecare utilizator.

```
#!/bin/bash
# Solution w/o arrays

prev_user=""
count=0
sum=0
for user_pid in $(ps -ef | awk 'NR > 1{print $1","$2}' | sort); do
    curr_user=$(echo "$user_pid" | cut -d, -f1)
    pid=$(echo "$user_pid" | cut -d, -f2)
    if [ "$curr_user" != "$prev_user" ]; then
        if [ $count -gt 0 ]; then
            echo "Avg for $prev_user is "$((sum/count))
        fi
        prev_user=$curr_user
        sum=0
        count=0
    fi
    sum=$((sum+pid))
    count=$((count+1))
done
```

10. Write a script that receives program/process names as command line arguments. The script will monitor all the processes in the system, and whenever a program with one of those names is run, the script will kill it and display a message. (see commands ps, kill, killall). Alternativ, comenzile pkill/pgrep pot fi folosite.
Scrieti un script care primeste ca argumente nume de procese. Scriptul va monitoriza toate procesele din sistem si, cand apare un proces cu unul din numele specificate, scriptul il va intrerupe si va afisa un mesaj. (Folositi comenzile ps, kill, killall). Alternatively, use pgrep/pkill

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Provide at least one name"
    exit 1
fi

while true; do
    for process in $@; do
        PIDs=""
        PIDs=$(ps -ef | awk '{print $8" "$2}' | grep -E "\<$process " | awk '{print $2}')
        if [ -n "$PIDs" ]; then
            kill -9 $PIDs
        fi
    done
    sleep 3
done
```

14. Write a shell script that recieves any number of words as command line arguments, and continuously reads from the keyboard one file name at a time. The program ends when all words received as parameters have been found at least once across the given files.

```
Let say that
file1.txt contains word1 and word2
file2.txt does not contain any of the 3 words
file3.txt contains word2 and word 3

./script.sh word1 word2 word3
We input the following:
file1.txt
file2.txt
file3.txt

The program stops after reading file3.txt because
word1 has been found in file1.txt
word2 has been found in file1.txt and file3.txt
word3 has been found in file3.txt

#!/bin/bash

declare -A words

for i in $@; do
    words[$i]=0
done

found_all=false

while ! $found_all; do
    found_all=true
    read -p "Input a file name: " file
    if [ -z "$file" ]; then
        echo "Empty input"
    elif [ ! -f "$file" ]; then
        echo "Not a file"
    else
        for word in ${!words[@]}; do
            if grep -q -E "\<$word\>" "$file"; then
                echo "Found $word in $file"
                words[$word]=1
            fi
            if [ 0 -eq ${words[$word]} ]; then
                found_all=false
            fi
        done
    fi

done

echo "All done"
```

11. Write a script that receives a directory as a command line argument. The script will delete all the C source files from the directory and will display all other text files sorted alphabetically.
Scrieti un script care primeste un director ca parametru la linia de comanda. Scriptul va sterge toate fisierele sursa C din director si va afisa celelalte fisiere text sortate alfabetic.

```
if test -d $1; then
      cd $1       ls -1|while read l; do
                     if file $l|grep -q "C source"; then
                                rm $l
                     fi
      done
      ls -1 |sort
fi
```