UNIX/Linux Operating System

Bash script exercises

Stefano Quer and Pietro Laface
Dipartimento di Automatica e Informatica
Politecnico di Torino

Exercise

- Write a bash script that computes the values of a function f(x) for all the triples of integer values stored in a file
  - $f(x) = 3 \cdot x^2 + 4 \cdot y + 5 \cdot z$
  - Example

1	1	2	17
2	1	3	31
1	3	4	35

File content

Output values

- Write two versions of the script using **`while`** and **`for`** statements, respectively

Solution 1

Using a for loop

Reads from file **one value at a time** because the output of the command goes in a list of strings

```
#!/bin/bash

iter=1
for val in `cat $1`
do
   if [ $iter -eq 1 ]
   then
     let f=3*val*val
   elif [ $iter -eq 2 ]
   then
     let f=f+4*val
   elif [ $iter -eq 3 ]
   then
     let f=f+5*val
     iter=0
     echo -n "$f "
   fi
   let iter=iter+1
done

exit 0
```

Solution 2

Using the while loop (reads a line at a time)

Reads a line and the string is assigned to variable **line**

```
#!/bin/bash

while read line
do
   iter=1
   for val in $line
   do
      if [ $iter -eq 1 ]
      then
        let f=3*val*val
      elif [ $iter -eq 2 ]
      then
        let f=f+4*val
      elif[ $iter -eq 3 ]
      then
        let f=f+5*val
      fi
      let iter=iter+1
   done
   echo -n "$f "
done < $1
```

Parsing the line

Loop on file lines

Solution 3

Using the while loop
(reads three values at a time)

```
#!/bin/bash

while read x y z
do
  f=3*x*x+4*y+5*z
  echo -n "$f "
done < $1

exit 0
```

Loop on file lines

Exercise

- Write a bash script that
  - Takes a filename from command line
  - Displays the file content
    - A line at a time, prepending the line number
    - A string at a time, prepending the string number

Solution

```
#!/bin/bash
n=1
while read line         # read a line
do
  echo "$n: $line"
  let n=n+1
done < $1               # Redirection !
n=1
for str in `cat $1`     # read a word
do
  echo "$n: $str"
  let n=n+1
done
```

Exercise

- Write a bash script that
  - Takes a filename from command line
  - Reads a sequence of integer number from the file
  - Each number represents a histogram bin value
  - Displays a horizontal histogram using '*'
  - Example

File content	Output
1	*
3	***
5	*****
4	****
2	**

Solution

```
#!/bin/bash

for n in $(cat $1)
do
  i=1
  while [ $i -le $n ]
  do
    echo -n "*"
    let i=i+1
  done
  echo
done

exit 0
```

Reads a number at a time

Prints without newline

Prints a newline

Exercise

- Write a bash script that displays
  - All files of the current directory
  - With "`.c`" extension
  - That include string "**`POSIX`**"
- Example of execution
  - `./myScript .`
  - `./myScript /home/bin`

Solution

```
#!/bin/bash
for file in $(ls *.c); do
  grep --quiet "POSIX" $file
  if [ $? -eq 0 ]
  then
    more $file
  fi
done
exit 0

# Alternative (single command):
# more $(grep  POSIX *.c -l)
# Notice the difference !!:
# grep POSIX *.c -l | more
```

grep
1) -q, --quiet, avoids printing the found line
2) If a file is found, returns (echo $?) 0 i.e., the condition is TRUE

grep -l
displays only the filenames matching the string POSIX

Exercise

- Write a bash script that
  - Takes a filename from command line
  - The file contains two columns of data
  - Example

    7 3

    2 23

    5 0
  - The script must overwrite the file swapping the two columns

Solution

```
#!/bin/bash

file="tmp"

while read var1 var2
do
  echo $var2 $var1
done <$1 >$file

mv $file $1

exit 0
```

Uses a temporary file ...

... renamed at the end of the script

Exercise

- Write a bash script that
  - Takes a set of strings from command line
  - The first string is a directory name
  - The others are filenames
    - `$myScript dir file1 file2 ... filen`
- The script
  - Creates the directory if it does not exist
  - For each file, ask the user if the file should be copied in the destination directory
  - Copy only files confirmed by the user

Solution

```
#!/bin/bash

if [ $# -le 1 ]
then
  echo "Run: $0 dir file1 file2 ..."
  exit 1
fi

if [ ! -d $1 ]
then
  echo "Create directory $1"
  mkdir $1
fi
```

Solution

```
dir=$1
shift
for i in $*
do
  echo -n "$i in $dir (y/n)? "
  read choice
  if ["$choice" = "y" ] ; then
    if cp $i $dir
    then
      echo "Copy done for $dir/$i"
    else
      echo "Error copying $i"
    fi
  fi
done
exit 0
```

The command line arguments are shifted to the left

Exercise

- Write a bash script that
  - Takes a filename (of a text file) from command line
  - Copy the file with the same filename, but with extension **xyx**
  - Modifies the original file
    - Adding at the beginning of each line the number of words in the line, and the total number of lines of the file
    - Sorting the lines according to their number of words

basename command

- **Syntax:**
  - `basename NAME [SUFFIX]`
  - Prints **NAME** with **any leading directory** components **removed**. If specified, it will also remove a trailing **SUFFIX** (typically a file extension)

```
> name='basename /home/user/current/file.txt .txt'
> echo $name
file
```

Solution

```
#!/bin/bash
if [ $# -ne 1 ]
then
  echo "usage $0 file.txt"
  exit 1
fi
newfilename=$(basename $1 txt)
newfilename=$newfilename“ xyz"
cp $1 $newfilename
nlines=$(cat $1 | wc –l)
rm -f tmp1.txt
while read line
do
  nwords=$(echo $line | wc –w)
  echo $nwords $nlines $line  >> tmp1.txt
done < $1
cat tmp1.txt | sort -k 1 -n > $1
rm tmp1.txt
exit 0
```

Filename without extension

Better:
`nlines=$(wc –l < $1)`

Add information on a temporary file

Sort and overwrite the original file

Clean-up

Exercise

- Write a bash script that
  - Takes 4 arguments (**`dir1`**, **`dir2`** e **`dir3`**, directory names, and **`n`**, an integer number)
  - Finds in **`dir1`** and **`dir2`** all files that have the same name, extension **`txt`** and more than **`n`** lines
  - Creates in directory **`dir3`** a version of these files with extension
    - **`eq`** save the lines that are equal in both files
    - **`dif`** save the lines that differ in the two files
    - **`cat`** concatenates the content of the two files

Solution

```
#!/bin/bash

if [ $# -ne 4 ]
then
  echo "usage: $0 dir1 dir2 dir3 n"
  exit 1
fi
if [ ! -d $3 ]
then
  mkdir $3
fi

for file in $(ls $1/*.txt); do
  name=$(basename $file ".txt")
  if [ -f "$2/$name.txt" ]; then
    n1=$(wc –l < $file)
    n2=$(wc –l < "$2/${name}.txt")
    if [ $n1 -gt $4 -a $n2 -gt $4 ]; then
```

find rather than ls

`find $1 -maxdepth 1 -type f -name "*.txt"`

Counts and controls the number of lines

Solution

```
      while read line; do
        grep -q -e "^$line$" "$2/$name.txt"
        if [ $? -eq 0 ]; then
          echo $line >> "$3/${name}.eq"
        else
          echo $line >> "$3/${name}.dif"
        fi
      done < $file
      while read line; do
        grep -q -e "^$line$" "$3/${name}.eq"
        if [ $? -eq 1 ]; then
          echo $line >> "$3/${name}.dif"
        fi
      done < "$2/$name.txt"
      cat $file "$2/${name}.txt" > "$3/${name}.cat"
    fi
  fi
done
```

Looks for this line in the second file

Looks for lines that are different in the second file

File concatenation