Il sistema operativo UNIX/Linux

Soluzione di problemi mediante script

Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

Esercizio

- Si scriva uno script in grado di calcolare i valori di una funzione f(x) per tutte le terne di valori intere memorizzate in un file
  - $f(x) = 3 \cdot x^2 + 4 \cdot y + 5 \cdot z$
  - Esempio

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

Contenuto del file

Valori da calcolare e visualizzare

- Il nome del file sia dato sulla riga di comando

Si risolva l'esercizio con il costrutto while e il costrutto for

Soluzione 1

Ciclo for

Valori letti da file uno alla volta !

```
#!/bin/bash

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

exit 0
```

Soluzione 2

Ciclo while
(lettura di righe intere e successiva parsificazione)

Tutto il contenuto di una riga è memorizzaato in row

```
#!/bin/bash

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

Parsificazione della riga

Soluzione 3

Ciclo while
(lettura di tre elementi alla volta)

Valori letti da file
tre alla volta !

```
#!/bin/bash

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

exit 0
```

Ciclo sulle righe

Esercizio

- Scrivere uno script bash in grado di visualizzare
  - Tutti i file del direttorio corrente
  - Con estensione ".c"
  - Che contengono almeno una volta la stringa "POSIX"

Soluzione

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

# Alternativa (commando unico):
# more $(grep -l POSIX *.c)
# Diverso da:
# grep -l POSIX *.c | more
```

grep
1) -q, --quiet, --silent, evita l'output della riga trovata
2) se trova file ritorna (echo $?) 0 quindi la condizione è vera

Grep
-l significa output soppresso tranne nome dei file

Esercizio

- Un file contiene due colonne di dati
  - Esempio

    7 3
    2 23
    5 0
- Scrivere uno script bash in grado di scambiare tra di loro tali colonne
- Lo script riceva il nome del file sulla riga di commando
- Si osservi che file di uscita e di file ingresso coincidono

Soluzione

```
#!/bin/bash

file="tmp"

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

mv $file $1

exit 0
```

Utilizza un file temporaneo

Che al termine del procedimento ridenomina correttamente

Esercizio

- Uno script riceve un insieme di stringhe sulla riga di comando
  - La prima stringa è il nome di un direttorio
  - Le stringhe successive sono nomi di file
    - $myScript dir file1 file2 ... filen
- Lo script deve
  - Creare il direttorio in caso esso non esista
  - Chiedere all'utente per ciascun file se è necessario effettuarne la copia nel direttorio specificato
  - In caso affermativo effettuare la copia del file

Soluzione

```
#!/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
```

Soluzione

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

exit 0
```

N.B. $* non include il nome del programma

Salta il primo parametro

Esame del 30.01.2018

Esercizio

- Il comando **`df file`** mostra lo spazio su disco disponibile sul file system contenente file
- Esempio

```
df /data/backup
Fifesystem  1K-blocks    Used  Avaifable Use% Mounted on
/dev/sda7    39056088 5881472   33174616  16% /data
```

  - Il secondo, terzo e quarto campo mostrano lo spazio totale, usato e disponibile sul flle system contenente /data/backup
  - I campi sono suddivisi da spazi
  - Si supponga nessun altro carattere di separazione sia usato e che lo spazio non compaia in nessun altro punto

Esame del 30.01.2018

Esercizio

- Si scriva uno script che riceva il percorso di un file sorgente e un percorso destinazione e
  - Controlli il corretto passaggio dei parametri
  - Effettui la copia in background del file sorgente nel percorso destinazione
  - Analizzi a intervalli regolari di un secondo lo spazio occupato sul file system destinazione, visualizzando a video la percentuale di avanzamento dell'operazione di copia
    - Si supponga questa sia ì'unica operazione in corso su tale file system

il comando **sleep n** può essere utilizzato per mettere in pausa lo script per **n** secondi

Soluzione

```
#!/bin/bash
if [ $# -ne 2 ]; then
  echo "Usage $0 <source> <destination>"
  exit 1
fi
if [ ! -f $1 ]; then
  echo "Source is not a valid file."
  exit 1
fi
if [ ! -d $2 ]; then
  echo "Destination is not a valid directory."
  exit 1
fi
source=$1
destination=$2
```

Verifica numero di parametri

Verifica validità parametri

Soluzione

```
size=$(ls -l $source | cut -d " " -f 5)
let "size=size/1024"

startUsed = $(df $destination | \
              tail -n 1 | \
              tr -s " " | \
              cut -d " " -f 3)

cp $source $destination &

transferred=0
percentage=0
```

Calcola dimensione file in blocchi da 1KB

Calcola dimensione file system destinazione

Copia in background

Soluzione

Verifica stato della copia in background

```
while [ $transferred -lt $size ]; do
  currentUsed = $(df $destination | \
                  tail -n 1 | \
                  tr -s " " | \
                  cut -d " " -f 3)
  let "transferred=currentUsed-startUsed"
  let "percentage=transferred*100/size"
  echo "Progress: $percentage%"

  sleep 1
done
```

Esame del 22.02.2018

Esercizio

- Uno script riceve quali parametri
  - Il nome di un file (nf) e tre interi (n1, n2 e n3)
  - Il file specifica un path su ciascuna riga
- Lo script deve
  - Verilicare che i 4 parametri siano specificati correttamente, che i valori siano interi siano positivi, e che sia $n1 \leq n2$
  - Per ciascuna riga del file
    - Verificare tale stringa individui un file regolare

Esame del 22.02.2018

Esercizio

- Se la dimensione del file è
  - Inferiore a n1 byte, cancellarlo
  - Compresa tra n1 e n2 byte, ignorarlo
  - Maggiore di n2 byte, comprimerlo. Comprimere un file significa
    - Farne una copia in un file con stesso path ma con aggiunta una ulteriore estensione compresso
    - Modificarne il contenuto copiando solo una stringa ogni n3 stringhe (ovvero occorre copiare solo le stringhe di posizione 1, 1*n3, 2 * n3, etc.). Si considerino le stringhe separate da spazi o da caratteri di "a capo"

Soluzione

```
#!/bin/bash
if [ $# -ne 4 ]; then
  echo "Usage $0 <list> <n1> <n2> <n3>"
  exit 1
fi
if [ ! -f $1 ]; then
  echo "List is not a valid file."
  exit 1
fi
if [ $2 -lt 0 ] || [ $3 -lt 0 ] || [ $4 -lt 0 ]; then
  echo "Values n1, n2 and n3 should be non-negative integers."
  exit 1
fi
if [ $2 -gt $3 ]; then
  echo "Values n1 should be non-greater than n2."
  exit 1
fi
```

Verifica numero di parametri

Verifica validità parametri

Soluzione

```
while read file; do

  if [ ! -f "$file" ]; then
    echo "Invalid file: $file"
    continue
  fi

  size=$(cat $file | wc -c)

  if [ $size -lt $2 ]; then
    rm -f $file
```

Per ogni path letto da file

Salta path non a file regolari

Calcola dimensione del file

Rimuovi file piccoli

Soluzione

```
  elif [ $size -gt $3 ]; then
    i=1
    for word in $(cat $file); do
      let "i--"
      if [ $i -eq 0 ]; then
        echo $word >> $file".compresso"
        i=$4
      fi
    done
  fi
done < $1
```

Comprimi file grossi

Esame del 28.01.2013

Esercizio

- Realizzare uno script che ricevuto come unico argomento un file di testo sia in grado di
  - Effettuare una **copia** del file in un file con lo stesso nome ma con estensione **xyx**
  - Modificare il file **originario**
    - Aggiungendo all'inizio di ogni riga il numero di parole della riga e il numero di righe totali del file
    - Ordinando le righe in ordine crescente in base al numero di parole

Comando **`basename`**:
elimina il path e eventualmente l'estensione dal nome di un file
`basename /home/quer/current/file.txt → file.txt`
`basename /home/quer/current/nome.txt ".txt" → nome`

Soluzione

```
#!/bin/bash
if [ $# -ne 1 ]
then
  echo "usage $0 file.txt"
  exit 1
fi
newfilename=$(basename $1 ".txt")
newfilename=$newfilename".xyz"
cat $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
```

".txt" = "*.txt" = .txt

Crea nuovo nome

Copia file. Anche:
`cp $1 $newfilename`

Aggiunge le informazioni su un file temporaneo

Ordina numericamente sul primo campo e sovrascrive

Clean-up

Esame del 03.02.2014

Esercizio

- Uno script ricevuti 4 parametri (**dir1**, **dir2** e **dir3**, nomi di direttori, e **n**, intero) deve
  - Trovare in **dir1** e **dir2** tutti i file che hanno lo stesso nome, estensione txt e più di n righe
  - Creare nel direttorio **dir3** una versione dei file con estensione
    - **eq** in cui vengono memorizzate le righe uguali dei due file originari
    - **dif** che memorizza solo le righe diverse dei due file
    - **cat** che memorizza la concatenazione dei due file

Controllare il numero di parametri
Creare il direttorio dir3 se non esiste

Soluzione

```
#!/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=$(cat $file | wc -l)
    n2=$(cat "$2/${name}.txt" | wc -l)
    if [ $n1 -gt $4 -a $n2 -gt $4 ]; then
```

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

Bastava eliminare il path

Per ogni file .txt del primo direttorio genero il corrispondente nome nel secondo direttorio

Conto e controllo il numero di righe

Soluzione

```
        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
```

Le righe in file1 anche in file2 fanno in eq quelle non in file2 vanno in dif

Controllo sul risultato della grep $?=0 (true o trovato)

Controllo inverso per le righe potenzialmente in diff

Concatenazione dei due file