MasterIT - Scripting Es1

Sh, Csh, Grep, Sed, Awk

Sandro.Angius@lnf.infn.it

24/10/2002

Shell

- Redirezione dell'I/O
  - Differenze Bourne Shell e CSH
  - Esempi Bourne:
    - » ps -ef > /tmp/listaprocessi
    - » ps -ef >> /tmp/listaprocessi
    - » more < /tmp/listaprocessi
    - » cat << FINE > /tmp/testo
> prima
> ultima
> FINE
    - » cat /tmp/listaprocessi | more
    - » ls xyz mnp *.krn 2>/dev/null
    - » ls xyz mnp *.krn > /tmp/lista 2>&1
    - » ls xyz mnp *.krn > /tmp/lista 2>/tmp/lista.err

Shell

- Redirezione dell'I/O
  - Esempi CSH:
    - » ps -ef > /tmp/listaprocessi
    - » ps -ef >! /tmp/listaprocessi
    - » ps -ef >> /tmp/listaprocessi
    - » more < /tmp/listaprocessi
    - » cat << FINE > /tmp/testo
> prima
> ultima
> FINE
    - » cat /tmp/listaprocessi | more
    - » ls xyz mnp *.krn >& /tmp/lista
    - » ls xyz mnp *.krn >>& /tmp/lista
    - » ls xyz mnp *.krn >&! /tmp/lista

Shell

- Variabili: Definizione, Uso e Quoting
  - Bourne:
    - pippo=pluto
      echo $pippo
      sh
      echo $pippo
      ^D
    - export pippo
      sh
      echo $pippo
      ^D
    - unset pippo
      echo $pippo

Shell

- Variabili: Definizione, Uso e Quoting
  - Bourne:
    - echo ${pippo:-var non definita}
    - echo ${pippo:=predefinito}
      echo $pippo
    - echo ${pippo:+valore ok}
    - echo ${pippo:?Errore variabile non definita}

Shell

- Variabili: Definizione, Uso e Quoting
  - CSH:
    - set pippo=pluto
      echo $pippo
      csh
      echo $pippo
      exit
    - setenv pippo pluto
      csh
      echo $pippo
      exit
    - unset pippo
    - unsetenv pippo

Shell

- Variabili: Definizione, Uso e Quoting
  - CSH:
    - echo $?myvar
    - set RGB=(red green blue)
      echo $RGB
      echo $#RGB
      echo $RGB[2]
      echo $RGB [1-2]
      set RGB=($RGB black brown)
      echo $#RGB
      echo $RGB[2-5]
      set RGB=($RGB[2-4])

Shell

- Variabili: Definizione, Uso e Quoting
  - Quoting CSH e Bourne:
    - echo $HOME
    - echo “La home directory e’ $HOME”
    - echo ‘$HOME contiene la home directory’

Shell

- Alcune variabili speciali:
  - echo “Il pid di $0 e’ $$”
  - cp /bin/sh mycmd
    ./mycmd
    echo “Il pid di $0 e’ $$”
    exit
  - CSH:
    - » ls xyz
      echo $status
      echo $status
  - Bourne:
    - » ls xyz
      echo $?
      echo $?

Shell

- Espressioni in Bourne:
  - test 1 -eq 1
    echo $?
  - [1 -eq 2]
    echo $?
  - mfl=/etc/passwd
    [-f $mfl] && echo "Il file $mfl esiste" || echo "Il file $mfl non esiste”
    mfl=/etc/passwd.xyz
    [-f $mfl] && echo "Il file $mfl esiste" || echo "Il file $mfl non esiste”
- Espressioni in CSH:
  - @ res = 1 + 2 * `date "+%S”` ; echo $res
  - @ res *= 3 ; echo $res
  - @ res = ($res << 2) ; echo $res

Shell

- If then else in Bourne:
  - sec=`date "+%S"`
    if [`expr $sec % 2` -eq 0]
    > then
    > echo Numero Pari
    > else
    > echo Numero Dispari
    > fi
- If then else in CSH:
  - if (3 * 2 == 5 + 1) echo sono uguali
  - csh << END
    ? if (`date "+%S"` % 2 == 0) then
    ? echo Numero Pari
    ? else
    ? echo Numero Dispari
    ? endif
    ? END

Shell

- Iterazioni in Bourne:
  - for cfile in `ls /bin`
    > do
    > test ! -x /bin/$cfile && echo “/bin/$cfile non e' eseguibile”
    > done
  - while test `date "+%S"` -lt 55
    > do
    > echo Waiting…
    > sleep 1
    > done
- Iterazioni in CSH:
  - foreach cfile (`ls /bin`)
    ? if (! -x /bin/$cfile) echo "/bin/$cfile non e' eseguibile”
    ? end
  - @ bpot =1
    while ($bpot < 2 << 20)
    ? echo $bpot
    ? @ bpot = ($bpot << 1)
    ? end

Regular Expr: Operatori Fondamentali

	BRE POSIX	ERE POSIX	BRE GNU	ERE GNU	Perl
escape	\	\	\	\	\
ancora	^	^	^	^	^
ancora	$	$	$	$	$
alternativa		|	|	|	|
raggruppamento	\(\)	( )	\(\)	( )	( )
elenco	[ ]	[ ]	[ ]	[ ]	[ ]
riferimento	\n		\n	\n	\n

Regular Expression: insiemi [ ] e [[: :]]

- Un insieme di caratteri es.: [aed]
- L'opposto e' [^aed]
- Possibili abbreviazioni es.: [A-G0-4]
- Le classi di caratteri [[: :]]

Classe di caratteri	Descrizione
upper	Collezione alfabetica delle lettere maiuscole.
lower	Collezione alfabetica delle lettere minuscole.
alpha	Lettere alfabetiche: di solito l'unione di '`upper`' e '`lower`'.
digit	Cifre numeriche.
alnum	Cifre alfanumeriche: di solito l'unione di '`alpha`' e '`digit`'.
punct	I caratteri di punteggiatura.
space	I caratteri definiti come «spazi bianchi» per qualche motivo.
blank	Di solito comprende solo '`<space>`' e '`<tab>`'.
cntrl	I caratteri di controllo che non possono essere rappresentati.
graph	Caratteri grafici: di solito l'unione di '`alnum`' e '`punct`'.
print	Caratteri stampabili: di solito l'insieme di '`alnum`', '`punct`' e di '`<space>`'.
xdigit	Cifre numeriche e alfabetiche per rappresentare numeri esadecimali.

grep: Global Regular Expression Parser

- ps -ef | grep sh
- ps -ef | grep -c sh
- ps -ef | grep -i sh
- ps -ef | grep -n sh
- ps -ef | grep -v sh
- ps -ef | grep -v -c sh
- grep -i -e imap -e post /etc/services
- grep -E "^p" /etc/services
- grep -E “[[:space:]][[:digit:]]{2}/tcp” /etc/services
- grep -E “[[:space:]][0-9]{2}/tcp” /etc/services
- grep -n -E '([ijk]-esim[io])|(jn)+|^z{5}$’
- grep -n '\(^[[:digit:]]\{1,\}\) = \1$’
- grep -n -E 'i(per|po)[[:alpha:]]+'

sed: Stream EDitor

- sed “” /etc/passwd
- sed p /etc/passwd
- sed -n 1,5p /etc/passwd
- sed -n ‘5,$p’ /etc/passwd
- sort /etc/services | sed -n '/^imap/,/^pop/p' | more
- sort /etc/services | sed '/^imap/,/^pop/d' | more
- sed s/:/@/ /etc/passwd
- sed s#:#@#g /etc/passwd
- man man | sed y/aeiou/UOIEA/ | more
- man man | tr aeiou UOIEA | more
- man man | sed 's@\([abc][[:alpha:]]*[aeoiu] \)@ ###\1### @g' | more
- (echo 'i\' ; echo 'SEPARAZIONE RIGA') > sed.test
  sed -f sed.test /etc/passwd

awk: Aho Weinberger Kernighan

- awk '{print $0}' < /etc/passwd
- awk -F: '{print $3, $NF}' < /etc/passwd
- awk 'BEGIN{print "---INIZIO---"}{print $0}END{print "---FINE---"}' < /etc/passwd
- awk -v k=10 ‘BEGIN{for (k--; k<100; k++) print int(rand()*1000)}’ > /tmp/lista.rnd

- awk 'BEGIN{tot=0; dispari=0; pari=0}\
  ? {tot+=$1; if ($1 % 2) {dispari++} else pari++}\
  ? END{printf "Somma %d record = %d\nRec Dispari = %d\nRec Pari = %d\n",\
  ? NR, tot, dispari, pari}' < /tmp/lista.rnd

- awk -F: -v base=2 -v inc=12300 '{if ($4<base) { print $0 }\
  ? else printf "%s:%s:%d:%d:%s:%s:%s\n", $1, $2, $3, $4+inc, $5, $6, $7}'\
  ? < /etc/passwd

- ypcat passwd | awk -F'[:,]' '$4 == 3385 {sub(" ","."$5); gsub(" ","",$5); \
  ? printf "%s:%s\n", $5, $1}’

- ypcat passwd | awk -F'[:,]' '$1 ~ /^a/ {sub(" ","."$5); gsub(" ","",$5); \
  ? printf "%s:%s\n", $5, $1}’

Script Shell

- Script Shell
  - Shebang: #!<path>/<pgm> Es.: #!/bin/sh
  - vi ese1.sh
  - #!/bin/sh
    echo Il comando si chiama $0
    echo “Il pid e’ $$”
    echo I parametri sono $#
    for param in $@
    do
        echo “Parametro: $param”
    done
  - chmod +x ese1.sh
  - ./ese1.sh 123 qwe ok now
  - sh -x ./ese1.sh other run

Script Shell

- vi ese2.csh
- #!/bin/csh

```
# Definizione nomi mesi
set mesi=(Gennaio Febbraio Marzo Aprile Maggio Giugno Luglio \
Agosto Settembre Ottobre Novembre Dicembre)

# Stampa nome mese per ogno parametro
# esce con messaggio al primo parametro errato
foreach par ($argv)
  if ( $par !~ [1-9] && $par !~ 1[0-2] ) then
      echo "'$par' non e' un mese valido..."
      break
  endif
  echo $mesi[$par]
end
```

- chmod +x ese2.csh
- ./ese2.csh 1 2 8 12
- csh -x ./ese1.sh 6 9 14

ese3.sh

- vi ese3.sh
- #!/bin/sh
  sort | awk ’
  BEGIN{ ctot=0 }
  {
  if (NR > 1)
  if ($0 == last)
  ctot++
  else {
  printf("%8d %s\n", ctot, last)
  last=$0
  ctot=1
  }
  else {
  last=$0
  ctot=1
  }
  }
  END{ if (ctot != 0) printf("%8d %s\n", ctot, last)}
  ' | sort -r
- chmod +x ese3.sh
- ps -e | awk '{print $NF}' | ./ese3.sh

Shell - Uso delle “trap”

- vi trap.sh
- **#!/bin/sh**

```
trap "echo Fine dello script" 0
trap "echo '<SEGNALE IGNORATO>'" 1 2 3

cnt=0
while [ $cnt -lt 10 ]
do
        echo "Counter = $cnt”
        cnt=`expr $cnt + 1`
        sleep 1
done

trap 1 2 3

echo
echo "Segnali riabilitati al default tranne 0”
echo

while [ $cnt -gt 0 ]
do
        echo "Counter = $cnt”
        cnt=`expr $cnt - 1`
        sleep 1
done
```

ese4.sh

- banner “Bye Bye” (su axcalc o dxcalc)

```
######                          ######
#     #   #   #  ######        #     #   #   #  ######
#     #    # #   #             #     #    # #   #
######      #    #####         ######      #    #####
#     #     #    #             #     #     #    #
#     #     #    #             #     #     #    #
######      #    ######        ######      #    ######
```

- ese4.sh “Bye Bye”

```
BBBBBB                          BBBBBB
B     B   Y   Y  EEEEEE        B     B   Y   Y  EEEEEE
B     B    Y Y   E             B     B    Y Y   E
BBBBBB      Y    EEEEE         BBBBBB      Y    EEEEE
B     B     Y    E             B     B     Y    E
B     B     Y    E             B     B     Y    E
BBBBBB      Y    EEEEEE        BBBBBB      Y    EEEEEE
```

- Su linux banner ~ /usr/libexec/filters/lpbanner -L"Bye Bye" | grep -E 'X|^ *$'
- Utilizzare awk; Soluzioni alternative?

ese5.sh

- Dato un numero IP valido e il numero di bit della netmask, riportare l'identificativo di network, esempio:
  - ./ese5.sh 193.206.84.219 21
    193.206.84.219/21 ===> 193.206.80.0
  - ./ese5.csh 192.168.160.14 24
    192.168.160.14/24 ===> 192.168.160.0
  - ./ese5.awk 10.199.213.87 12
    10.199.213.87/12 ===> 10.192.0.0
- Codifica con sh, csh, awk

Bibliografia

- man sh
- man csh
- man regexp
- man grep
- man sed
- man awk
- Appunti di informatica libera (© Daniele Giacomini)
  Esiste una copia in: http://www.lnf.infn.it/computing/doc/AppuntiLinux.pdf