

Tipi di Shell

- sh – Bourne shell
- csh – C shell
- ksh – Korn shell
- bash – Bourne Again (Bash) shell
- tcsh – TENEX C shell
- zsh – Z shell
- rc – rc shell
- es – es shell
- → csh e bash sono le più comuni.
- Per cambiare il tipo di shell basta digitare il suo nome nel prompt dei comandi:
 - *csh* (*cambia il tipo di shell*)

Shell: variabili di sistema

- **PATH**
- **HOME**
- **USER**
- **PWD**
- **SHELL**
- **HOSTNAME**
- **HOSTTYPE**

Per visualizzare il valore di tutte le variabili d'ambiente si usa il comando “env”

Shell: variabili di sistema

Per visualizzare il valore di una specifica variabile d'ambiente si può filtrare l'output di "set" con una "grep".

Ad esempio: "set | grep PATH"

Programmazione della Shell

Variabili definite dall'utente:

z=3

echo "z vale \$z" # stampa "z vale 3"

read y # legge da tastiera una stringa e la assegna a y

x=0

**let x=\$x+1 # assegna alla variabile x il valore x+1
cioè 1**

!!! Usare le virgolette se si mettono degli spazi con let

Programmazione della Shell

Assegnare il risultato di un comando a una variabile

Il risultato di tutto quello che è fra \$(....) è assegnato a una variabile.

Esempio:

x=\$(ls -l | wc -l) #questo comando conta un file in più

let x="\$x-1"

echo Sono presenti \$x files e/o cartelle nella cartella \$PWD

Il file .bashrc

Alias dell'utente

alias rm='rm -i'

alias cp='cp -i'

alias mv='mv -i'

#PATH

export PATH=\$PATH:/home/folino/psb

(adesso tutti i file contenuti nella cartella psb verranno eseguiti dovunque mi trovi)

export PATH=\$PATH:.

(Aggiungo la cartella corrente al PATH; Posso eseguire gli script nella mia cartella senza usare la notazione ./programma)

source .bashrc (attivo le modifiche senza dover riavviare la shell)

Programmazione della Shell

Variabili passate al comando:

\$1 è il primo argomento, \$2 il secondo, etc.

\$0 contiene il nome del comando

**\$# contiene il numero di argomenti passati al comando
(\$0 escluso)**

**S@ contiene tutti gli argomenti della linea di comando
(\$1, \$2, \$3, etc.); si usa nelle ripetizioni enumerative
(for)**

Programmazione della Shell

Istruzione condizionale if

if <lista-comandi> ; **then**

<comandi>

[**else** <comandi>]

fi

Le parole chiave (do, then, fi, etc.) devono essere

o a capo o dopo il separatore ;

Programmazione della Shell

Esempio:

```
if [ $1 -gt 10 ]
```

```
then
```

```
    echo "il parametro è maggiore di 10"
```

```
else
```

```
    echo "il parametro non è maggiore di 10"
```

```
fi
```

Programmazione della Shell

Valutazione di una espressione:

[**-<opzioni> <nomefile>**] ritorna uno stato uguale o diverso da zero

[**-f <nomefile>**] esistenza di file

[**-d <nomefile>**] esistenza di directory

[**-r <nomefile>**] diritto di lettura sul file (-w e -x)

[**<stringa1> = <stringa2>**] valuta se due stringhe sono uguali # o diverse

[**-z <stringa1>**] valuta se la stringa è nulla

[**<stringa1>**] valuta se la stringa non è nulla

[**<numero1> [-eq -ne -gt -ge -lt -le] <numero2>**] confronta tra loro due stringhe numeriche, usando uno degli operatori relazionali indicati

! not

-a and

-o or

Scrivere un programma che riceva come argomento (da linea di comando) il nome di un file ed il nome di una directory, e sposti il file nella directory:

Ad esempio:

sposta fileA directoryB

```
if [ $# -ne 2 ]; then  
    echo sintassi: $0 nomefile nomedirectory  
    exit 1  
fi  
  
if [ -f $1 -a -d $2 ]; then  
    mv $1 $2  
  
else  
    echo file e/o directory inesistenti  
fi
```

Modificare il precedente programma in modo che il nome del file e della directory non siano letti da linea di comando, ma da input.

Ad esempio:

sposta

Nome del file:

fileA

Nome della directory:

directoryB

echo Nome del file: ; read file

if [! -f \$file]; then

echo file inesistente

exit 1

fi

echo Nome della directory: ; read directory

if [! -d \$directory]; then

echo directory inesistente

exit 2

fi

mv \$file \$directory

Scrivere un programma che riceva due argomenti: il nome di un file F, ed un numero N. Il programma crea il file F (usando il comando “touch”) ed appende (usando il comando >>) un carattere (nell’esempio seguente il carattere B) al file F per un numero di volte pari ad N.

Ad esempio:

crea nomefile dimensione

Programmazione della Shell

Istruzione ripetitiva while

while <lista-comandi>

do

<comandi>

done


```
if [ $# -ne 2 ]; then
    echo sintassi: $0 nomefile numero
else
    touch $1
    x=0
    while [ $x -lt $2 ]; do
        echo -n B >> $1
        let x=$x+1
    done
fi
```

Modificare il programma precedente in modo che riceva un ulteriore argomento, che specifica qual è la stringa che deve essere appesa per N volte al file F.

Ad esempio:

crea nomefile dimensione ciao

```
if [ $# -ne 3 ]; then
    echo sintassi: $0 nomefile numero stringa
else
    touch $1
    x=0
    while [ $x -lt $2 ]; do
        echo -n $3 >> $1
        let x=$x+1
    done
fi
```

Programmazione della Shell

Istruzione ripetitiva for

for <variabile> **in** \$<variabile>

do

<comandi>

done

Scrivere un programma che riceve come argomenti una parola e un file e verifica quante volte la parola è presente in tale file

Esempio: conta_parole casa file_testo

Restituisce quante volte la parola casa è presente in file testo

```
if [ $# -ne 2 ]; then
    echo sintassi: $0 parola nomefile
else
    par=$(more $2)
    cont=0
    for i in $par
    do
        if [ $1 = $i ]; then
            let cont=$cont+1
        fi
    done
    echo "trovate $cont occorrenze della parola $1"
fi
```

Programmazione della Shell

Istruzione case

```
case $<variabile> in
```

```
Caso1 ) comandi1;;
```

```
Caso2 ) comandi2;;
```

```
.....
```

```
* ) comandi default
```

```
esac
```

Scrivere un programma che riceve come argomenti tre nomi di cartelle e copia tutti i file con estensione .c contenuti nella prima cartella nella seconda cartella e quelli con estensione .java nella terza cartella

Esempio: copiafile programmi prog_c prog_java


```
if [ $# -ne 3 ]; then
    echo sintassi: $0 cartella_base cartella1 cartella2
else
    files=$(ls $1)
    for i in $files
    do
        case $i in
            *.c ) copy $1/$i $2;;
            *.java ) copy $1/$i $3;;
            * ) echo "File $i non copiato";;
        esac
    done
fi
```

Shell: Più comandi in una linea

```
cp pippo.txt backup/ ; rm pippo.txt  
// eseguo entrambi
```

```
cp pippo.txt backup/ && rm pippo.txt  
// se fallisce il primo non eseguo il secondo
```

```
cp pippo.txt backup/ || cp pippo.txt backup2/  
// se riesce il primo non eseguo il secondo
```

Shell: Espressioni Regolari

Esempio: `grep ^public *.java` (trova tutte le righe che iniziano con public)

| Operator | Effect |
|----------|---|
| . | Matches any single character. |
| ? | The preceding item is optional and will be matched, at most, once. |
| * | The preceding item will be matched zero or more times. |
| + | The preceding item will be matched one or more times. |
| {N} | The preceding item is matched exactly N times. |
| {N,} | The preceding item is matched N or more times. |
| {N,M} | The preceding item is matched at least N times, but not more than M times. |
| - | represents the range if it's not first or last in a list or the ending point of a range in a list. |
| ^ | Matches the empty string at the beginning of a line; also represents the characters not in the range of a list. |
| \$ | Matches the empty string at the end of a line. |
| \b | Matches the empty string at the edge of a word. |
| \B | Matches the empty string provided it's not at the edge of a word. |
| \< | Match the empty string at the beginning of word. |
| \> | Match the empty string at the end of word. |

Shell: sed (1)

Comandi di Editing di Sed

| Command | Result |
|---------|--|
| a\ | Append text below current line. |
| c\ | Change text in the current line with new text. |
| d | Delete text. |
| i\ | Insert text above current line. |
| p | Print text. |
| r | Read a file. |
| s | Search and replace text. |
| w | Write to a file. |

| Option | Effect |
|-----------|---|
| -e SCRIPT | Add the commands in SCRIPT to the set of commands to be run while processing the input. |
| -f | Add the commands contained in the file SCRIPT-FILE to the set of commands to be run while processing the input. |
| -n | Silent mode. |
| -V | Print version information and exit. |

Opzioni di sed

Shell: sed (2)

```
sandy ~> cat -n example
 1 This is the first line of an example text.
 2 It is a text with errors.
 3 Lots of errors.
 4 So much errors, all these errors are making me sick.
 5 This is a line not containing any errors.
 6 This is the last line.

sandy ~>
```

1) **sed -n '/^This.*errors.\$/p'** example

This is a line not containing any errors.

2) **sed '3,\$d'** example

This is the first line of an example text.

It is a text with errors.

3) **sed 's/errors/errors/'** example

This is the first line of an example text.

It is a text with errors.

Lots of errors.

So much errors, all these errors are making me sick.

This is a line not containing any errors.

This is the last line.

Shell: sed (3)

4) sed 's/errors/errors/g' example

This is the first line of an example text.

It is a text with errors.

Lots of errors. So much errors, all these errors are making me sick.

This is a line not containing any errors.

This is the last line.

5) sed 's/^/>/' example

This is the first line of an example text.

> It is a text with errors.

> Lots of errors.

> So much errors, all these errors are making me sick.

> This is a line not containing any errors.

> This is the last line.

6) sed -e 's/errors/errors/g' -e 's/last/final/g' example

This is the first line of an example text.

It is a text with errors. Lots of errors.

So much errors, all these errors are making me sick.

This is a line not containing any errors.

This is the final line.