NOZIONI BASE

SHELL E SCRIPT LINUX

Aggiornato al 11 gennaio 2006

Ermes ZANNONI
(ermes@zannoni.to.it)
(http://www.zannoni.to.it)

Indice :

1. Introduzione

Dispensa introduttiva alla shell di LINUX. Per chi è alle prime armi e non vuole istallarlo vi consiglio di cercare una distribuzione a vostro piacimento che faccia il boot da CD-ROM (linux live).
Questa dispensa è di distribuzione gratuita per cui non può essere messa in vendita o utilizzata per scopi commerciali.

2. La Shell

Avviata la shell da utente e non amministratore (root) ci appare una schermata nera con una scritto:

[nome@percorso host]$

il simbolo $ indica che siamo entrato con l'accesso di utente, per cui non avremo il completo controllo del PC. Se al posto del simbolo $ abbiamo il simbolo #

[nome@percorso host]#

vuol dire che siamo entrati come amministratori (root).

Per passare da utente a amministratore bisogna digitare **su** e successivamente viene richiesta di digitare la password di amministratore

[nome@percorso host]$ **su**
password:
[nome@percorso host]#

Attenzione, per chi utilizza le distribuzioni di linux live cercate su internet la password per accedere al sistema come amministratori.

Se eventualmente non ci si ricorda esattamente come si chiama il file da cercare, abbiamo a disposizione dei caratteri jolly, che sono * e ?.

Ad esempio se vogliamo cercare il file Pippo.txt possiamo scrivere nel seguente modo :

P?ppo.txt
*.txt
Pi*.txt
…

Se si lancia qualche comando che effettua un lungo ciclo, con la sequenza di tasti CONTROL + D e possibile arrestarlo.

2.1 Comandi da Shell

In seguito troverete l'elenco con relativa spiegazione dei comandi principali.

Per sapere le numerose opzioni di un determinato comando vedere il manuale, digitare:

man *comando* -> man *cd*

Per uscire dal man basta premere q.

2.1.1 File e directory

Comando	Note
ls	Elenca il contenuto di una directory
ls –a	Elenca il contenuto di una directory e i file nascosti
ls –l	Elenco dettagliato dei file e directory
cd *dir*	Per passare da una directory all'altra
cd ..	Per tornare indietro di una directory
cp *file1 file2*	Copia il contenuto del file1 nel file2
mv *file1 file2*	Rinomina il file1 in file2
rm *file*	Elimina un file
mkdir *dir*	Crea una nuova directory
rmdir *dir*	Eliminare una directory
pwd	Mostra la corrente directory di lavoro
diff *file1 file2*	Mostra la differenza tra due file di testo
file *file*	Per sapere il formato del file
find *dir* –name *file* –print	Per trovare un file in una directory

2.1.2 Processi (programmi in esecuzione)

Comando	Note
ps	Per visualizzare i programmi in esecuzione
ps u	Per avere un report più dettagliato
kill *pid*	Per terminare un processo
kill –STOP *pid*	Per sospendere un processo
kill –CONT *pid*	Per riavviare un processo
comando &	Aggiungendo & alla fine del comando, possiamo far eseguire un processo in background, in questo modo possiamo continuare a fare nuove operazioni

Per verificare lo stato di un programma, digitando il comando **ps**, bisogna guardare sulla colonna STAT se il processo è su S (sospeso) o R (in esecuzione).

2.1.3 Password

Comando	Note
passwd	Per cambiare la password

2.1.4 Autorizzazioni sui file e directory

Ogni file e ogni directory ha delle autorizzazioni per leggere, scrivere e eseguire. Per visualizzare le seguenti autorizzazioni dei file utilizzare il comando **`ls -l`**.
Le autorizzazioni verranno visualizzate nel seguente modo:

```
- rwx r-- rw-
```

Il primo trattino indica che è un file senza utilizzi speciali, se fosse una directory al posto del trattino avremmo una **d**.
I successivi tre gruppi da tre caratteri indicano le autorizzazioni, suddivisi in:

1° gruppo	Autorizzazioni utente
2° gruppo	Autorizzazioni di gruppo
3° gruppo	Altre autorizzazioni

Ogni gruppo e ulteriormente suddiviso in:

r	lettura
w	scrittura
x	esecuzione
-	nulla

Comando	Note
chmod *numero file* *esempio:* chmod 600 testo.txt	Per modificare le autorizzazioni sui file e le directory Numero: 644 utente(r/w), gruppo e altro(r) 600 utente(r/w), gruppo e altro(-) 755 utente(r/w/x), gruppo e altro(r/x) 700 utente(r/w/x), gruppo e altro(-) 711 utente(r/w/x), gruppo e altro(x)

2.2 Script della Shell

Uno script è semplicemente una serie di comandi scritti all'interno di un file testo. Ricordarsi di controllare le autorizzazioni del file che si crea. Una volta scritto il nostro programma possiamo avviarlo tramite la shell utilizzando il nome del file preceduto dalla sintassi ./, esempio:

```
./nome_file
```

2.2.1 Intestazione

La prima riga deve essere obbligatoriamente la seguente, scritta senza spazzi:

```
#!/bin/sh
```

Successivamente a questa consiglio sempre di scrivere qualche commento preceduto dal simbolo #, esempio:

```
#
# ----------------------------------------
# Stampa sul video la scritta CIAO A TUTTI
# ----------------------------------------
#
```

2.2.2 Stampare una scritta su video

Per stampare una scritta su video utilizziamo il comando echo, vedi esempio:

```
echo "CIAO A TUTTI"
```

2.2.3 Variabili

Nell'esempio seguente vediamo l'utilizzo di una variabile all'interno di un programma:

```
variabile_1=verde
variabile_2=rosso
echo $variabile_1 $variabile_2
```

eseguendo questo script, otteniamo sullo schermo:

```
verde rosso
```

Oltre a questo scopo, abbiamo delle variabili speciali che ci permettono di avere dei parametri tramite la riga di comando, per essere più chiari facciamo alcuni esempi di utilizzo:

1° Esempio

```
#!/bin/sh
variabile_1=$1
variabile_2=$3
echo $variabile_1 $variabile_2
```

Se questo script lo lanciamo con il seguente comando:

```
./nome_file rosso verde blu
```

Otterremo stampato nello schermo:

```
rosso
blu
```

Sono collegati nel seguente modo:

$1 -> rosso
$2 -> verde
$3 -> blu
…

2° Esempio

```
#!/bin/sh
echo $1
shift
echo $1
shift
echo $1
shift
```

Se questo script lo lanciamo come nell'esempio precedente:

```
./nome_file rosso verde blu
```

Otterremo stampato nello schermo:

```
rosso
verde
blu
```

In pratica inserendo il comando shift ci permette di scalare gli argomenti, da $3 a $2, da $2 a $1.

3° Esempio

```
#!/bin/sh
echo $1
shift
echo $1
shift
echo $#
```

Se questo script lo lanciamo come nell'esempio precedente:

```
./nome_file rosso verde blu
```

Otterremo stampato nello schermo:

```
rosso
verde
1
```

La variabile $# contiene il numero di argomenti passati allo script, nel nostro caso utilizzando l'istruzione shift ci consente di sapere quanti elementi rimangono.

4° Esempio

```
#!/bin/sh
echo $@
```

Se questo script lo lanciamo come nell'esempio precedente:

```
./nome_file rosso verde blu
```

Otterremo stampato nello schermo:

```
rosso verde blu
```

La variabile $@ consente di passare tutti gli argomenti allo script.

5° Esempio

```
#!/bin/sh
echo $0 Script non valido
```

Se questo script lo lanciamo come nell'esempio precedente:

```
./nome_file rosso verde blu
```

Otterremo stampato nello schermo:

```
nome_file non valido
```

La variabile $0 consente di ottenere il nome dello script.

2.2.4 Diramazioni

2.2.4.1 if … then … e if… then… else…

```
#!/bin/sh
if [ $1 = verde ]; then
     echo il colore è verde
fi
```

```
#!/bin/sh
if [ $1 = verde ]; then
     echo il colore è verde
else
     echo il colore non è verde
fi
```

Quando viene eseguito uno dei due script precedenti, l'istruzione if valuta se l'argomento $1 = verde sia vera o falsa (valore booleano), se questa valutazione risultasse vera esegue la prima istruzione (echo il colore è verde), se nel caso risultasse falsa la seconda (echo il colore non è verde).

Volendo possiamo fare tanti altri tipi di controlli, ve ne elenco alcuni:

&& (and) o || (or)

funzione and e or

```
#!/bin/sh
if [ $1 = verde ] || [ $1 = giallo ]; then
     echo il colore è verde
fi
```

-e

Restituisce true se un file è esistente

```
#!/bin/sh
if [ -e file.txt ]; then
     echo il file è esistente
else
     echo il file non è esistente
fi
```

-s

Restituisce true se un file non è vuoto

```
#!/bin/sh
if [ -s file.txt ]; then
     echo il file è scritto
else
     echo il file è vuoto
fi
```

test aritmetici

```
-eq  ->   uguale
-ne  ->   non uguale
-lt  ->   minore
-gt  ->   maggiore
-le  ->   minore o uguale
-ge  ->   maggiore o uguale
```

esempio:

```
#!/bin/sh
if [ 13 -eq 13 ]; then
     echo il valore è 13
fi
```

2.2.4.2 case

Se si vuole una diramazione multipla, utilizziamo il comando case che prende di prendere percorsi diversi in base al valore della stringa.

```
#!/bin/sh
case $1 in
      verde)
            echo colore verde
            ;;
      rosso)
            echo colore rosso
            ;;
      giallo)
            echo colore giallo
            ;;
      *)
            echo altri
            ;;
esac
```

Questo script confronta $1 con ogni case delimitato dal carattere), se trova un valore case uguale a $1, esegue i comandi successivi fino a ;;.
Terminate le istruzioni salta al termine dell'istruzione esac.
Nell'esempio precedente se $1 non trova un case uguale a esso esegue il l'ultimo caso *)

2.2.5 Cicli

Se dobbiamo ripetere alcune volte le stesse istruzioni variando solamente alcune variabili conviene utilizare il ciclo for:

```
#!/bin/sh
for stringa in verde rosso giallo; do
     echo $stringa
done
```

Ottenendo:

verde
rosso
giallo

La shell imposta la variabile stringa con il valore del primo elemento (verde), successivamente stampa su video il suo risultato arrivando all'istruzione done che fa iniziare il processo dal for, impostando nuovamente la variabile stringa con il successivo elemento (rosso), cosi via.