

Esercitazione Script Shell

Esercizio 1

Scrivere uno script shell che controlli se l'utente `UserName` sia collegato oppure no.

Lo script viene lanciato nel modo seguente:

```
$ testlogin UserName
```

Lo script deve:

1. stampare un errore se manca il parametro `UserName`;
2. stampare il messaggio "L'utente `UserName` è collegato" nel caso l'utente sia collegato;
3. stampare il messaggio "L'utente `UserName` non è collegato" nel caso l'utente non sia collegato.

Suggerimenti: Per risolvere l'esercizio è necessario utilizzare:

- le variabili predefinite della shell (in particolare, in questo esercizio, la variabile che contiene il numero dei parametri dello script e quella che contiene il primo argomento);
- la struttura di controllo `if`;
- i comandi `who`, `test`, `grep`, `echo`.

Nota: A volte risulta comodo "buttare via" l'output di un comando che non viene utilizzato; in questo caso si adotta la tecnica di ridirigere il suo output su `/dev/null` (file speciale usato per questo scopo).

Esercizio 1-bis

Utilizzando lo script precedente come base, scrivere uno script shell che controlla se l'utente `UserName` esiste (quindi il suo `UserName` è contenuto all'interno del file `/etc/passwd`) e nel caso che esista controlla se è collegato oppure no.

Lo script deve:

1. stampare un errore se manca il parametro `UserName`;
2. stampare il messaggio "L'utente `UserName` non esiste" nel caso l'utente non esista;
3. stampare il messaggio "L'utente `UserName` esiste" nel caso l'utente esista; in questo caso stampare se è collegato oppure no.

Nota: La macchina Unix sulla quale vi collegate non effettua l'autenticazione dei vostri utenti dal file `/etc/passwd` ma tramite il dominio Windows. Di conseguenza la vostra login non esiste in quel file. Per verificare gli utenti 'riconosciuti' direttamente dal sistema potete digitare il comando `cat /etc/passwd`.

Esercizio 2

Scrivere uno script shell che prende come parametri una sequenza di caratteri, separati tra loro da uno spazio, e li trasforma in numeri tramite la seguente regola (simile a quella dei cellulari):

```
0=ABC, 1=DEF, 2=GHI,  
3=JKL, 4=MNO, 5=PQR,  
6=STU, 7=VWX, 8=YZ
```

Lo script deve controllare la correttezza di ciascun parametro (deve essere una lettera AZ), nel caso trovi un parametro errato deve stampare un messaggio appropriato ed uscire. Si DEVE usare la struttura di controllo `Case` (e NON l'`if`).

Esempio:

```
$ ./trasformaInNumero A S T N O B
```

0
6
6
4
4
0

Suggerimenti: È consigliato l'utilizzo della struttura di controllo iterativa `for`.

Nota: La variabile `$*` contiene una stringa composta dalla concatenazione di tutti i parametri della linea di comando separati da uno spazio.

Esercizio 2-bis

Utilizzando lo script precedente come base, modificarlo in modo tale che stampi il numero su una linea unica. Cioè:

```
$ ./trasformaInNumeroUnito A S T N O B  
066440
```

Suggerimenti: È conveniente, quasi indispensabile, fare uso di una variabile di appoggio alla quale assegnare il numero via via costruito e concatenato con il nuovo carattere/numero. Si consiglia di fare delle prove direttamente dalla shell (cioè dalla linea di comando) per trovare il modo.

Esercizio 3

Scrivere uno script shell che calcola il numero di file delle sotto-directory di una certa directory, passata come parametro, e stampa a video per ogni directory il messaggio "La sotto-directory *NomeDir* contiene *N file*" (è molto importante riuscire a comporre esattamente questo messaggio). Gli eventuali file contenuti nella directory *NomeDir* devono essere saltati.

Lo script deve inoltre effettuare i seguenti controlli:

1. stampare un errore se manca il parametro `NomeDir`;
2. stampare un errore se `NomeDir` non è una directory.

Suggerimenti: In questo esercizio è necessario fare uso del Command Substitution (sostituzione di comandi) per l'assegnamento del risultato di un comando ad una variabile di appoggio. I vari controlli richiesti dallo script sono tutti realizzabili tramite appropriate opzioni del comando `test`.