

# Lab. di Sistemi Operativi

## Esercitazioni proposte per le lezioni del 11 e 12/03/10

Utilizzando la shell di Linux e disponendosi in gruppi di due persone per ogni PC del laboratorio.

1. Elencare i file nella directory corrente
2. Elencare i file nella directory corrente visualizzando i permessi associati
3. Visualizzare il nome assoluto della directory corrente
4. Spostare la directory corrente alla radice del filesystem
5. Visualizzare il nome assoluto della directory corrente
6. Ritornare alla propria home indicandone il nome in modo assoluto; verificare l'operazione
7. Spostare la directory corrente alla radice del filesystem
8. Ritornare alla propria home indicandone il nome in modo relativo; verificare l'operazione
9. Spostare la directory corrente alla radice del filesystem
10. Ritornare alla propria home utilizzando il comando `cd` senza argomenti; verificare l'operazione
11. Copiare il file `/usr/share/vim/vimrc` nella directory corrente
12. Copiare il file `/usr/share/vim/vimrc` nella directory corrente attribuendo al nuovo file il nome "file-copia"
13. Visualizzare i file nella directory corrente, comprendendo il file che iniziano con il punto
14. Cancellare il file "vimrc" dalla home
15. Cancellare il file `filecopia` e `.vimrc` con una unica command line
16. Copiare `/usr/share/vim/vimrc` in `.vimrc` nella propria home e modifichiamolo con l'editor vim
17. Creiamo un file di nome prova con il vim, inserendo al suo interno la linea "Questo e' un file di prova"
18. Rendiamo il file prova illeggibile da tutti
19. Rendiamo il file leggibile dal solo proprietario
20. Verificare cosa avviene al file prova settando a 1 e 0 i 12 bit dei permessi
21. Invocare `cat` per visualizzare un file, ma utilizzando la ridirezione della shell
22. Utilizzare `cat` per creare in file, sempre mediante ridirezione della shell; cosa avviene se si esegue la ridirezione con due *maggiori* affiancati?
23. Creare e visualizzare una variabile d'ambiente
24. Creare e visualizzare una variabile d'ambiente contenente spazi
25. Combinare i comandi `ps` e `grep` per individuare i processi che eseguono bash; utilizzare un file temporaneo. Cosa fanno `ps` e `grep`? Utilizzare **man ps** e **man grep** per scoprirlo
26. Combinare i comandi `ps` e `grep` utilizzando il carattere di pipe
27. Cosa e' cambiato? il S.O. ha creato un file temporaneo per voi o no?

28. Creiamo con il vim un file di nome script.sh ed inseriamo al suo interno una command line
29. Rendere il file eseguibile
30. Mettiamo in esecuzione il file prova.sh (con e senza l'ausilio di source)
31. Modificare lo script in modo che venga visualizzato il nome dell'interprete. Utilizzare "ls -l /proc/\$\$/exe". Chi esegue lo script?
32. Modifichiamo lo script in modo che la prima linea contenga solo i seguenti caratteri, senza virgolette: "#!/bin/csh". Chi esegue ora lo script?
33. Creare uno script per la creazione automatica di uno script. Si salvi il file con il nome **crea\_script.sh**. Il nome del file da creare deve essere creato partendo da una stringa fornita dall'utente appendendo i caratteri ".sh". Lo script creato deve avere lo sha-bang in linea 1 e il bit di eseguibilità attivo per tutti gli utenti. Lettura e modifica consentiti solo al proprietario.

# Soluzione

## Esercitazioni proposte per le lezioni del 11 e 12/03/10

Utilizzando la shell di Linux e disponendosi in gruppi di due persone per ogni PC del laboratorio.

1. Elencare i file nella directory corrente

**Soluzione:**

```
ls
```

2. Elencare i file nella directory corrente visualizzando i permessi associati

**Soluzione:**

```
ls -l
```

3. Visualizzare il nome assoluto della directory corrente

**Soluzione:**

```
pwd
```

4. Spostare la directory corrente alla radice del filesystem

**Soluzione:**

```
cd /
```

5. Visualizzare il nome assoluto della directory corrente

**Soluzione:**

```
pwd
```

6. Ritornare alla propria home indicandone il nome in modo assoluto; verificare l'operazione

**Soluzione:**

```
cd /home/nXXXXX  
pwd
```

7. Spostare la directory corrente alla radice del filesystem

**Soluzione:**

```
cd /
```

8. Ritornare alla propria home indicandone il nome in modo relativo; verificare l'operazione

**Soluzione:**

```
cd home/nXXXX  
pwd
```

9. Spostare la directory corrente alla radice del filesystem

**Soluzione:**

```
cd /
```

10. Ritornare alla propria home utilizzando il comando cd senza argomenti; verificare l'operazione

**Soluzione:**

```
cd
pwd
```

11. Copiare il file /usr/share/vim/vimrc nella directory corrente

**Soluzione:**

```
# Copia di un file: comando cp sorgente destinazione: se la destinazione e' una
# directory, il nome del file viene mantenuto:
cp /usr/share/vim/vimrc .
ls
```

12. Copiare il file /usr/share/vim/vimrc nella directory corrente attribuendo al nuovo file il nome "filecopia"

**Soluzione:**

```
# Copia di un file indicando il nome della copia:
cp /usr/share/vim/vimrc filecopia
ls
```

13. Visualizzare i file nella directory corrente, comprendendo il file che iniziano con il punto

**Soluzione:**

```
# Nomi di file "particolari": se il nome inizia con un punto il file non e'
# normalmente visibile
cp /usr/share/vim/vimrc .vimrc
ls
# Visualizzare i file "nascosti" con ls
ls -a
```

14. Cancellare il file "vimrc" dalla home

**Soluzione:**

```
# Cancellazione di file: il comando rm
rm vimrc
```

15. Cancellare il file filecopia e .vimrc con una unica command line

**Soluzione:**

```
rm filecopia .vimrc
```

16. Copiare /usr/share/vim/vimrc in .vimrc nella propria home e modifichiamolo con l'editor vim

**Soluzione:**

```
# .vimrc sara' il file di configurazione di vim per il nostro "utente".
# Molti programmi consentono una configurazione personalizzata tramite un file
# nella home che inizia con il punto e, di conseguenza, risulta normalmente invisibile.
cp /usr/share/vim/vimrc .vimrc
vim .vimrc
```

17. Creiamo un file di nome prova con il vim, inserendo al suo interno la linea "Questo e' un file di prova"

**Soluzione:**

```
vim prova
# da vim, comando iQuesto e' un file di prova<ESC>
# da vim, comando :wq
```

18. Rendiamo il file prova illeggibile da tutti

**Soluzione:**

```
chmod a-r prova
ls -l prova
vim prova
# Il vim non mostrera' il contenuto del file: non e' leggibile
```

19. Rendiamo il file leggibile dal solo proprietario

**Soluzione:**

```
chmod u+r prova
ls -l prova
vim prova
```

20. Verificare cosa avviene al file prova settando a 1 e 0 i 12 bit dei permessi

**Soluzione:**

```
chmod a-rwx prova
ls -l prova
chmod 00777 prova
ls -l prova
```

21. Invocare cat per visualizzare un file, ma utilizzando la ridirezione della shell

**Soluzione:**

```
cat < prova
```

22. Utilizzare cat per creare in file, sempre mediante ridirezione della shell; cosa avviene se si esegue la ridirezione con due *maggiori* affiancati?

**Soluzione:**

```
cat > prova
cat >> prova
# La seconda command line esegue una ridirezione in append, quindi il contenuto
# precedente del file viene mantenuto.
# Ricordare il CTRL-D per indicare la fine del file
```

23. Creare e visualizzare una variabile d'ambiente

**Soluzione:**

```
variabile=valore
echo $variabile
```

24. Creare e visualizzare una variabile d'ambiente contenente spazi

**Soluzione:**

```
# Doppi apici:
variabile=un valore
echo $variabile
variabile="un valore"
echo $variabile
```

25. Combinare i comandi ps e grep per individuare i processi che eseguono bash; utilizzare un file temporaneo. Cosa fanno ps e grep? Utilizzare **man ps** e **man grep** per scoprirlo

**Soluzione:**

```
# Combinazione di comandi:
ps -A > tempfile
grep bash < tempfile
rm tempfile
# Cosa fa ps?
man ps
# Per uscire dal man, tasto q
```

26. Combinare i comandi ps e grep utilizzando il carattere di pipe

**Soluzione:**

```
ps -A | grep bash
```

27. Cosa e' cambiato? il S.O. ha creato un file temporaneo per voi o no?

**Soluzione:**

```
No! Il sistema operativo ha creato un canale di comunicazione fra i due
processi ed ha provveduto alla sincronizzazione reciproca in modo che:
1) Il processo produttore (ps) venga interrotto quando ci sono troppi caratteri
in attesa nel canale
2) Il processo utilizzatore (grep) venga svegliato quando ci sono caratteri
da processare nel canale
```

28. Creiamo con il vim un file di nome script.sh ed inseriamo al suo interno una command line

**Soluzione:**

```
vim script.sh
# Come convenzione, terminiamo il nome dei file in linguaggio shell con .sh
# inseriamo un comando shell, ad esempio:
echo "Ciao Mondo"
```

29. Rendere il file eseguibile

**Soluzione:**

```
chmod 00744 script.sh
```

30. Mettiamo in esecuzione il file prova.sh (con e senza l'ausilio di source)

**Soluzione:**

```
source script.sh
./script.sh
```

31. Modificare lo script in modo che venga visualizzato il nome dell'interprete. Utilizzare "ls -l /proc/\$\$/exe". Chi esegue lo script?

**Soluzione:**

```
# Chi esegue il file per voi?
# Aggiungiamo nello script la linea:
ls -l /proc/$$/exe
```

32. Modifichiamo lo script in modo che la prima linea contenga solo i seguenti caratteri, senza virgolette: `#!/bin/csh`. Chi esegue ora lo script?

**Soluzione:**

Inseriamo un commento in prima linea:

```
#!/bin/csh
echo "Ciao Mondo"
```

Il programma (shell) che esegue `script.sh` ora non è `bash` ma `tcsh` (sintassi simile ma non compatibile). Fortunatamente la command line contenuta nello script viene interpretata allo stesso modo ma, tipicamente, uno script scritto in linguaggio `bash` non può essere interpretato da `csh`.

Il commento in linea 1, quindi, è fondamentale per garantire che l'interprete sia quello giusto. Se il commento fosse assente, la selezione dell'interprete sarebbe dipendente dalla configurazione del sistema e dalle preferenze dell'utente. Uno script ben progettato deve, pertanto, dichiarare l'interprete da utilizzare (esistono tanti linguaggi di script oltre alle varie shell, come il `perl` o il `php` ad esempio).

33. Creare uno script per la creazione automatica di uno script. Si salvi il file con il nome **`crea_script.sh`**. Il nome del file da creare deve essere creato partendo da una stringa fornita dall'utente appendendo i caratteri `.sh`. Lo script creato deve avere lo sha-bang in linea 1 e il bit di eseguibilità attivo per tutti gli utenti. Lettura e modifica consentiti solo al proprietario.

**Soluzione:**

```
#!/bin/bash
# crea_script: script per creare altri script
#
# Richiesta interattiva del nome del file da creare:
echo -n "Nome dello script da creare, senza .sh"; read nomefile

# Aggiunta dell'estensione .sh:
nomefile=${nomefile}.sh

# Creazione del file (vuoto)
>${nomefile}

# Modifica dei permessi: tutti lo possono eseguire, solo il proprietario può
# leggerlo o modificarlo
chmod 00711 $nomefile
```