

CORSO DI SISTEMI OPERATIVI A - ESERCITAZIONE 1

1 Accesso al Sistema

Questa prima esercitazione ha lo scopo di fornire le indicazioni essenziali per familiarizzare con il sistema operativo Linux.

Come attivare una sessione :

1. avviare la macchina virtuale Linux Suse 11 cliccando l'icona omonima sul desktop di Windows ;
2. attendere il completamento della fase di boot del sistema operativo ;
3. alla richiesta di *Username* e *Password*, inserire i dati del proprio account ;
4. se il login non ha successo, riprovare almeno una volta e in caso di nuovo insuccesso verificare con i tecnici i dati del proprio account.

Al termine della sessione di lavoro, all'interno della macchina virtuale selezionare "Computer → Arresta → Arresta" dal menu e attendere lo spegnimento della macchina virtuale, prima di fare logout da Windows.

Chi sono? Utilizzare il comando *whoami*.

NOTA 1: ogni comando di sistema ha il suo manuale di riferimento, che può essere consultato con il comando *man <nomecomando>* (ad es. *man ls*).

NOTA 2: ogni volta che si richiede ad uno shell l'esecuzione di un comando, viene ricercato un file eseguibile con lo stesso nome del comando all'interno dei direttori specificati nella variabile d'ambiente PATH. Per visualizzare il percorso assoluto del file eseguibile individuato, utilizzare il comando *which <nomecomando>* (ad es. *which ls*).

2 Gestione del File System

2.1 File e Direttori

Esercizio guidato:

1. Esistono vari modi per creare un file. Il modo più semplice per creare un file vuoto è quello di il comando *touch*: *touch pippo*
Dopo aver utilizzato il comando *touch* per creare il file pippo non si ottiene alcuna conferma dell'avvenuta esecuzione dell'operazione. Questo atteggiamento è tipico dei sistemi Unix i cui comandi tendono a non manifestare il successo delle operazioni eseguite. Si può comunque verificare: *ls -l pippo*
2. rimuovere il file *pippo*: *rm pippo*

3. creare il direttorio *prova*: `mkdir prova`
4. copiare il file *pippo.txt* da `/temi_esame/00Unix/soa/eserc1` (direttorio condiviso accessibile in sola lettura) a *prova*: `cp /temi_esame/00Unix/soa/eserc1/pippo.txt prova`
5. portarsi nel direttorio *prova*: `cd prova`
6. verificare la posizione del direttorio corrente nel File System: `pwd`
7. visualizzare il contenuto del direttorio corrente (*prova*): `ls`
8. identificare il tipo del file *pippo.txt*: `file pippo.txt`
9. visualizzare il contenuto del file *pippo.txt*: `cat pippo.txt`
10. tornare nel direttorio a monte: `cd ..`
11. spostare il file `~/prova/pippo.txt` nel direttorio corrente: `mv ~/prova/pippo.txt .` . N.B. non dimenticare il punto . finale !
12. rimuovere il direttorio *prova*: `rmdir prova`
13. portarsi nel direttorio `/usr/local`: `cd /usr/local`
portarsi in `/usr/bin` effettuando uno spostamento relativo: `cd ../bin`
verificare la posizione del direttorio corrente nel File System: `pwd`
14. portarsi in `/var/tmp` effettuando uno spostamento relativo cumulato: `cd ../../var/tmp`
verificare la posizione del direttorio corrente nel File System: `pwd`
15. tornare nel proprio direttorio principale: `cd ~`
16. creare una struttura articolata di direttori:
`mkdir carbonio`
`mkdir carbonio/idrogeno`
`mkdir carbonio/idrogeno/elio`
17. provare a rimuovere il direttorio *carbonio*: `rmdir carbonio`
il comando precedente non funziona perchè *carbonio* contiene dei direttori; per rimuoverlo occorre utilizzare la ricorsione: `rmdir -p carbonio/idrogeno/elio`
NOTA: per ottenere lo stesso risultato si può utilizzare `rm -r carbonio`, ma `rmdir -p` permette di cancellare singoli rami di un albero di direttori.

Esercizio proposto:

creare un nuovo direttorio e copiare al suo interno tutti i file di `/temi_esame/00Unix/soa/eserc1` che contengono nel nome la stringa *pippo* (suggerimento: utilizzare il metacarattere *).

2.2 Linking

Esercizio guidato:

1. creare un link simbolico al direttorio `/temi_esame/00Unix/soa/eserc1`: `ln -s /temi_esame/00Unix/soa/eserc1 es1`
2. verificare che il link è stato creato correttamente: `ls -l`
3. creare un hard link al file `es1/pluto.txt`: `ln es1/pluto.txt pluto2.txt`

4. verificare la condivisione dell'i-node tra il file originale e l'hard-link, eseguendo `ls -l` (si osservi il numero nella seconda colonna)
5. visualizzare il contenuto del file `pluto2.txt` (l'hard link): `cat pluto2.txt`
6. cancellare `pluto2.txt`, rispondendo `y` (yes) alla richiesta di autorizzazione.

2.3 Protezione

Esercizio guidato:

1. copiare il file `/temi_esame/00Unix/soa/eserc1/pippo.txt` nel proprio direttorio principale
2. visualizzare il contenuto dettagliato del direttorio corrente: `ls -l`
3. cambiare i diritti di `pippo.txt` in modo che sia tolto il permesso in lettura a tutti gli utenti, anche al proprietario: `chmod -r pippo.txt`
4. visualizzare nuovamente il contenuto dettagliato del direttorio (cosa è cambiato?)
5. verificare che il file non è leggibile: `cat pippo.txt`
6. ripristinare il permesso in lettura e togliere quello in scrittura: `chmod +r-w pippo.txt`
7. verificare che il file non è scrivibile: `cp /temi_esame/00Unix/soa/eserc1/pluto.txt pippo.txt`
Un tentativo di sovrascrittura genera una segnalazione di errore, come nell'esempio appena visto, così come qualunque altro tentativo di modificare il suo contenuto.
8. fare in modo che `pippo.txt` sia scrivibile solo dal proprietario: `chmod u+w pippo.txt`
NOTA: oltre all'opzione `u` ci sono anche `g` (concessione del permesso al gruppo) e `o` (concessione del permesso a tutti gli altri);
9. Il comando `chmod` può operare anche sui direttori; creare una cartella `prova` nel proprio direttorio principale e poi togliere il permesso in esecuzione a `prova` con il comando: `chmod -x prova`

Esercizio proposto:

ripristinare i diritti di esecuzione del direttorio `prova` solo per il proprietario.

3 Altri comandi di sistema

3.1 Comandi di stato

Provare i comandi visti a lezione: `date`, `time <nomecomando>`, `who`, `ps [-elf]`, `top`, `free`, `du`.

3.2 Ricerca

Provare l'esempio visto a lezione relativo al comando `grep`.

Provare anche il seguente comando: `find / -name bash -print`

Questo comando esegue una ricerca per i file e le directory denominati `bash` all'interno di

tutte le directory che si articolano a partire dalla radice. Il file viene trovato, ma tutte le volte che *find* tenta di attraversare directory per cui non si ha il permesso, si ottiene una segnalazione di errore.

4 Utilizzo dello Shell

4.1 Completamento automatico

Il completamento automatico è un modo attraverso cui lo Shell aiuta l'utente a completare un comando. La richiesta di completamento viene fatta attraverso l'uso del tasto [Tab].

Si preparano alcuni file di esempio (i nomi utilizzati sono volutamente lunghi):

```
touch microcontrollore  
touch microfono  
touch ballatoio
```

Supponendo di voler utilizzare questi nomi all'interno di una riga di comando, si può essere un po' infastiditi dalla loro lunghezza. Utilizzando il completamento automatico si risolve il problema:

```
ls bal[Tab]
```

Dopo avere scritto solo *bal*, premendo il tasto [Tab] si ottiene il completamento del nome, dal momento che non esiste alcun file o direttorio (nella posizione corrente) che inizi nello stesso modo.

Il completamento automatico dei nomi potrebbe essere impossibile. Infatti, potrebbe non esistere alcun nome che coincida con la parte iniziale già inserita, oppure potrebbero esistere più nomi composti con lo stesso prefisso. In quest'ultimo caso, il completamento si ferma al punto in cui i nomi iniziano a distinguersi:

```
ls mic[Tab]ro
```

In questo caso, il completamento si spinge fino a *micro* che è la parte comune dei nomi *microcontrollore* e *microscopio*. Per poter proseguire occorre aggiungere un'indicazione che permetta di distinguere tra i due nomi. Volendo selezionare il primo di questi nomi, basta aggiungere la lettera *c* e premere nuovamente il tasto [Tab]:

```
ls mic[Tab]roc[Tab]ontrollore
```

4.2 I metacaratteri

L'utilizzo di metacaratteri rappresenta una forma alternativa di completamento dei nomi. Infatti è compito dello Shell trasformare i simboli utilizzati per questo scopo.

Per questo esercizio si utilizzano i file creati nella sezione precedente: *microcontrollore*, *microscopio* e *ballatoio*.

L'asterisco rappresenta una sequenza indefinita di caratteri di qualunque tipo, esclusa la barra di separazione tra le directory. Per cui l'asterisco utilizzato da solo rappresenta tutti i nomi di file disponibili nella directory corrente.

```
ls micro*
```

Questo comando fa in modo che lo Shell elenchi tutti i file il cui nome inizia per *micro*. Il punto interrogativo rappresenta esattamente un carattere qualsiasi. Con il comando:

```
ls ??????o?o
```

si ottiene il seguente output:

```
ballatoio microfono
```

Le parentesi quadre vengono utilizzate per delimitare un elenco o un intervallo di caratteri. Rappresentano un solo carattere tra quelli contenuti, o tra quelli appartenenti all'intervallo indicato.

Osservare cosa fa il comando: `ls ??????[abdf]*`

Osservare anche cosa fa: `ls ??????[c-f]*`

Il fatto che lo Shell sostituisca alcuni caratteri impedisce di fatto il loro utilizzo nei nomi di file e direttori. Se esiste la necessità, è possibile evitare la sostituzione di questi facendoli precedere dal carattere `\`, che funge da carattere di "escape". Ad esempio, per creare il file (vuoto) *sei*otto*, si usa:

```
touch sei\*otto
```

4.3 Redirezione dell'I/O

Esercizio guidato:

1. scrivere in un file il contenuto di `/temi_esame/00Unix/soa/eserc1`, ordinato in base alla data di creazione: `ls -t /temi_esame/00Unix/soa/eserc1 > paperino.txt`
2. visualizzare il contenuto del file generato: `cat paperino.txt`
3. è possibile indirizzare l'input e l'output contemporaneamente; come prova, scrivere in un nuovo file il contenuto, questa volta ordinato alfabeticamente, del file creato precedentemente: `sort < paperino.txt > topolino.txt`

4.4 Piping dei comandi

Visualizzare il numero dei file contenuti nel direttorio *bin*: `ls /bin | wc -l`

Esercizio proposto:

utilizzando il piping dei comandi e la redirezione dell'output, scrivere in un file di testo i 10 file più recenti del direttorio `/temi_esame/00Unix—/soa/eserc1` (suggerimenti: utilizzare il comando `head` e leggere il manuale del comando `ls`, con `man ls`).

4.5 Modalità di esecuzione

In tutti gli esercizi svolti finora i comandi sono stati eseguiti in modalità *foreground*. Come visto a lezione esiste una seconda modalità di esecuzione detta *background*, che si ottiene

ponendo alla fine della linea di comando il simbolo `&`. In quest'ultima modalità lo shell padre non attende il completamento dell'esecuzione del comando.

Esercizio guidato:

1. digitare il seguente comando: `ls -lR > prova.txt &`
2. lo shell è immediatamente disponibile, quindi digitare: `date`

4.6 Controllo della modalità di esecuzione dei comandi

Il comando `jobs` fornisce l'elenco dei comandi attualmente eseguiti in background, indicandone il job number (tra parentesi quadre), lo stato e il nome.

Esercizio guidato:

1. eseguire Emacs (programma per l'editing di testi) in background: `emacs &`
2. eseguire firefox (web browser) in background: `firefox &`
3. eseguire il comando: `jobs`

E' possibile portare da background a foreground l'esecuzione di un comando utilizzando il comando `fg`. Se si sono più esecuzioni in background, bisogna specificare quale portare in foreground indicandone il job number.

E' altresì possibile effettuare l'operazione inversa, cioè portare un'esecuzione da foreground a background; per fare ciò bisogna anzitutto sospendere l'esecuzione del comando premendo contemporaneamente i tasti `[Ctrl^z]`, e poi utilizzare il comando `bg`.

Esercizio guidato:

1. portare in foreground l'esecuzione di firefox: `fg %2`
2. sospendere l'esecuzione di firefox: `[Ctrl^z]`
3. portare in background l'esecuzione di firefox: `bg`