Gestione degli spazi

- Se possibile, è più semplice sostituire uno spazio con un altro carattere, come ad esempio un *under score* _ o un trattino -
  - Quindi, *mio cugino* diventerebbe *mio_cugino*, che è sintatticamente una sola parola
  - Purtroppo, non è sempre possibile: a volte ci troviamo a lavorare con file o directory creati da altri
  - E comunque vogliamo avere la libertà di poterlo fare!

```
corso@sistemi-operativi:~/tutorial$ mkdir "folder 1"
corso@sistemi-operativi:~/tutorial$ mkdir 'folder 2'
corso@sistemi-operativi:~/tutorial$ mkdir folder\ 3
corso@sistemi-operativi:~/tutorial$ mkdir "folder 4" "folder 5"
corso@sistemi-operativi:~/tutorial$ mkdir -p "folder 6"/"folder 7"
corso@sistemi-operativi:~/tutorial$
```

Creazione di file

- Finora ci siamo occupati soltanto di directory, ma possiamo anche creare dei file
- Ad esempio, supponiamo di voler creare un file avente come contenuto l'output del comando *ls*, eseguito all'interno di */home/corso/tutorial*

```
corso@sistemi-operativi:~/tutorial$ ls
 cugino   dir1   dir2   dir3  'folder 1'  'folder 2'  'folder 3'  'folder 4'  'folder 5'  'folder 6'   mio
corso@sistemi-operativi:~/tutorial$
```

- E' possibile redirigere ***l'output*** di un comando su file tramite l'operatore *>*

```
corso@sistemi-operativi:~/tutorial$ ls > output.txt
corso@sistemi-operativi:~/tutorial$ ls
 cugino   dir1   dir2   dir3  'folder 1'  'folder 2'  'folder 3'  'folder 4'  'folder 5'  'folder 6'   mio   output.txt
corso@sistemi-operativi:~/tutorial$
```

Lettura di un file

- Possiamo leggere il contenuto di un file utilizzando il comando ***cat*** (con**cat**enate)

```
corso@sistemi-operativi:~/tutorial$ cat output.txt
cugino
dir1
dir2
dir3
folder 1
folder 2
folder 3
folder 4
folder 5
folder 6
mio
output.txt
corso@sistemi-operativi:~/tutorial$
```

- Possiamo creare ulteriori file con contenuti testuali arbitrari con il comando ***echo***
  - Proviamo ad eseguire *echo "This is a test"*. Cosa succede?
  - Come possiamo creare un file che contenga la frase *This is a test* ?

Più files, più letture

- Esercizio: creiamo tre file, chiamati *test_1.txt, test_2.txt, test_3.txt,* che contengano rispettivamente i seguenti testi:
  - This is a test
  - This is a second test
  - This is a third test

```
corso@sistemi-operativi:~/tutorial$ echo "This is a test" > test_1.txt
corso@sistemi-operativi:~/tutorial$ echo "This is a second test" > test_2.txt
corso@sistemi-operativi:~/tutorial$ echo "This is a third test" > test_3.txt
corso@sistemi-operativi:~/tutorial$
```

- Ora proviamo a leggere in modo concatenato il contenuto dei tre file

```
corso@sistemi-operativi:~/tutorial$ cat test_1.txt test_2.txt test_3.txt
This is a test
This is a second test
This is a third test
corso@sistemi-operativi:~/tutorial$
```

Creiamo un nuovo file che contenga il contenuto dei vari test

- Come possiamo creare un nuovo file con all'interno il contenuto dei vari file di test che abbiamo scritto prima?
  - *cat t* > combined.txt*
- Cosa succede se ripetiamo questo stesso comando?

- E se invece volessimo effettivamente "appendere" un testo ad un file esistente?
  - Utilizziamo l'operatore *>>*
- Proviamo:
  - *cat t* >> combined.txt*
  - *echo "ho appeso una linea" >> combined.txt*
  - *cat combined.txt*
- Ripetiamo i comandi un po' di volte (provate ad utilizzare la freccia in su), fino ad ottenere un file più lungo della dimensione dello schermo. Come possiamo leggere tale file?
  - *less combined.txt*

Spostare file

- E' possibile spostare un file o una cartella utilizzando il comando *mv* (**m**o**v**e)
  - Il comando *mv combined.txt dir1* sposta il file *combined.txt* all'interno della cartella *dir1*
- Verifichiamo se il file è stato effettivamente spostato

```
corso@corso-os:~$ mv  combined.txt dir1
corso@corso-os:~$ ls dir1
combined.txt
corso@corso-os:~$
```

- Ora supponiamo di aver cambiato idea e di non volere il file in *dir1*, ma nella directory dove si trovava inizialmente. Come facciamo? Diverse strade, proviamone alcune:
  - Posso specificare come percorsi di partenza e destinazione dei percorsi assoluti
  - Posso spostarmi nella cartella che attualmente contiene combined.txt ed utilizzare dei percorsi relativi per spostare il file una directory sopra
  - Posso rimanere dove mi trovo ed utilizzare i percorsi relativi per spostare il file nella directory corrente
  - ...

Spostamenti multipli di file e cartelle

- *mv* consente di spostare con un solo comando diversi elementi in una cartella di destinazione
  - Esempio: *mv combined.txt test_* dir3 dir2* sposta all'interno di *dir2* gli elementi *combined.txt, test_*, dir3*
  - Notare che gli elementi oggetto dello spostamento possono essere sia file che cartelle, mentre l'elemento di destinazione deve necessariamente essere una cartella (non avrebbe senso altrimenti)

```
corso@corso-os:~$ mv combined.txt test_* dir3 dir2
corso@corso-os:~$ ls dir2
combined.txt  dir3  test_1.txt  test_2.txt  test_3.txt
corso@corso-os:~$
```

- Esercizio: spostare ancora una volta il file *combined.txt* da *dir2* a *dir4/dir5/dir6*

Copiare e rinominare

- Il comando per la copia di un file è molto simile a quello per lo spostamento:
  - *cp <percorso di origine> <percorso di destinazione>*
- Come copiamo il file combined.txt che sta dentro dir4/dir5/dir6 all'interno della directory corrente?
  - *cp dir4/dir5/dir6/combined.txt .*
- A questo punto, supponiamo di voler creare una copia di backup di *combined.txt* all'interno della stessa cartella, chiamata *backup_combined.txt.* Come facciamo?
  - *cp combined.txt backup_combined.txt*

- E se il nome che abbiamo scelto non dovesse piacerci? Supponiamo di voler rinominare il file *backup_combined.txt* in *combined_backup.txt.*
  - *mv backup_combined.txt combined_backup.txt*

- Esercizio: rinominare tutte le cartelle con lo spazio che avevamo creato prima.

Cancellare file e cartelle

- Attenzione! La cancellazione di un file o di una cartella da shell non utilizza il meccanismo del cestino
  - Questo vuol dire che un elemento, una volta cancellato, è perso.
- La cancellazione di un file avviene tramite il comando *rm* (**re**m**o**ve), seguito da una lista di file da rimuovere. Esempio:
  - *rm dir4/dir5/dir6/combined.txt combined_backup.txt*
- Proviamo ora a rimuovere anche le cartelle *folder_** che avevamo creato precedentemente.

```
rm: cannot remove 'folder_1': Is a directory
rm: cannot remove 'folder_2': Is a directory
rm: cannot remove 'folder_3': Is a directory
rm: cannot remove 'folder_4': Is a directory
rm: cannot remove 'folder_5': Is a directory
rm: cannot remove 'folder_6': Is a directory
```

Cancellare cartelle

- Per la cancellazione di cartelle dobbiamo utilizzare il comando *rmdir*.
  - Proviamo di nuovo ad eliminare le cartelle *folder_ **

```
rmdir: failed to remove 'folder_6': Directory not empty
```

- *rmdir* richiede che la cartella da rimuovere sia vuota. Non deve contenere ne' sottocartelle ne' files.

- Esiste un modo per rimuovere una cartella e tutto il suo contenuto, incluse le sue sottocartelle e, a loro volta, le loro sottocartelle, e così via
  - Si utilizza l'opzione *–r* di *rm* per indicare una rimozione ricorsiva.