Sistemi Operativi, Secondo Modulo, Canale M–Z
Riassunto della lezione del 29/02/2016

Igor Melatti

Il filesystem ed i file

- Altro comando importante: `cp [-r] [-i] [-a] [-u] {filesorgenti} filedestinazione`
  - permette di copiare file e directory
  - 2 modalità basilari:
    * con 2 argomenti, `filedestinazione` può essere un file (in questo caso, ovviamente, il sorgente dev'essere a sua volta un file...)
    * con 2 argomenti, la destinazione può non esistere, e verrà creata (un file se la sorgente è un file, una directory se la sorgente è una directory)
    * con più di 2 argomenti, `filedestinazione` dev'essere una directory (esistente)
    * differentemente dall'omologo comando MS-DOS, non è possibile avere un solo argomento: se si vuole copiare un file che si trova in un'altra directory nella cwd (mantenendone il nome), il secondo argomento sarà `.`
  - tra i `filesorgenti` ci possono essere file e/o directory
  - da notare che se la destinazione è una directory, i sorgenti vengono copiati dentro la directory destinazione
  - se ci sono directory, allora occorre dare l'opzione `-r`, altrimenti quelle directory non verranno copiate (ma i file sì)
  - se la copia avviene su file esistenti, verranno sovrascritti; con l'opzione `-i`, prima di sovrascrivere viene chiesta conferma
  - con l'opzione `-u`, la sovrascrittura avviene solo se l'mtime del sorgente è più recente di quello della destinazione (o quello di destinazione non esiste)
  - i permessi del file sorgente potrebbero non venire preservati: sono soggetti alla dura legge dell'`umask`
  - per forzare a mantenere i permessi, c'è l'opzione `-a` (questa opzione serve anche per altri motivi, che verranno discussi in seguito)

- **esercizio:** verificare il funzionamento di ciascuna delle opzioni mostrate. Per esempio, per l'opzione `-i`, provare a copiare un file sorgente in un file destinazione esistente e vedere che effettivamente viene chiesta conferma. Per l'opzione `-a`, è sufficiente creare un file ed aggiungere qualche permesso, poi fare la copia con e senza `-a`...
- **esercizio:** verificare il funzionamento del comando `cp` con diverse tipologie di argomenti: i) 2 file (tutti e 4 i casi tra esistenti e non); ii) 2 directory (tutti e 4 i casi tra esistenti e non); iii) 3 file (l'ultimo sia esistente che non); iv) 3 directory (idem); v) 2 file e 2 directory...

- Altro comando: `mv [-i] [-u] [-f] {filesorgenti} filedestinazione`
  - come `cp`, ma serve a *spostare* anziché *copiare*: quindi, i sorgenti risulteranno *cancellati* dopo l'esecuzione del comando, ed esisteranno solo nella destinazione
  - con 2 argomenti omologhi (2 file o 2 directory) effettua in pratica una ridenominazione
  - le opzioni `-i` e `-u` hanno lo stesso significato di `cp`; `-f` è il contrario di `-i` (ed è l'opzione di default)
  - `-r` e `-a` non servono: è come se fossero sempre abilitate
  - **esercizio:** rifare entrambi gli esercizi visti per `cp`
- Per cancellare e basta, c'è il comando `rm [-f] [-i] [-r] {file}`
  - cancella definitivamente i file e le directory indicati (non completamente vero nel caso di hard link, vedere più avanti)
  - le opzioni `-f` e `-i` sono come quelle della `mv` (ma stavolta il default è `-i`); l'opzione `-r` è come quella della `cp`
  - **esercizio:** creare una directory, poi crearci dentro un file, togliergli il permesso di scrittura a quest'ultimo e cancellarlo. Lo lascia fare? Perché? Provare a fare la stessa dentro la directory predefinita `(/tmp`
- Script per "giocare" con i permessi: `create_dirs_and_files.script`
  - creare una nuova directory e copiarci (o spostarci) dentro `create_dirs_and_files.script`
  - eseguire con il seguente comando: `bash create_dirs_and_files.script`
  - dopodiché, eseguire `ls -lR` (nota a margine: le opzioni si possono sempre mettere a fattor comune scrivendo una sola volta il dash e poi tutte le opzioni che si vogliono; attenzione, funziona solo per le opzioni con un solo dash, quelle con 2 dash vanno sempre staccate)
  - come si può vedere, ora ci sono 8 file, uno per ogni permesso, e 8 directory, una per ogni permesso

  - all'interno di ciascuna directory, si ripete lo schema: 8 file e 8 directory, ciascuna di queste ultime con dentro 8 file
  - **esercizio:** testare i vari permessi, come sono riportati nelle tabelle della lezione 2. Ad esempio, provare a leggere un file solo eseguibile, o ad appendere testo ad un file solo leggibile, o ad attraversare una directory senza permesso di esecuzione...

- Comando `ln [-s] sorgente [destinazione]`
  - soft (o symbolic) ed hard link
  - in pratica: copiare un file, senza copiare l'intero file (che potrebbe essere grande)
  - inoltre, successive modifiche al *contenuto* della sorgente si rifletteranno sulla destinazione, e viceversa
  - il comando realizza un soft link (se c'è l'opzione `-s`) o un hard link (altrimenti)
  - se c'è un solo argomento, allora la sorgente dev'essere un file in un'altra directory, e il link destinazione avrà lo stesso nome di questo file
  - soft link: viene creato un nuovo file (destinazione), il contenuto del quale coincide con `sorgente` (lo si può intuire guardando la dimensione del file con `ls -l` o con `stat`); da notare che spesso questo contenuto è direttamente negli attributi del file
  - se si cancella il file sorgente, il link diventa "morto", e provare a visualizzare il file porta ad un errore
  - hard link: aumenta il link count della sorgente, e crea una destinazione con lo stesso link count
  - se si cancella la sorgente, il link count decresce, ma la destinazione continua a mantenere il contenuto del file
  - se si fa un hard link di un hard link, il link count cresce ancora (per tutti i file coinvolti)
  - cancellare un file non vuol dire più automaticamente rimuovere il suo contenuto dal disco: potrebbero esserci hard links...
  - non si può vedere a cosa "punta" un hard link
  - non si possono fare hard links a directory (tranne quelli predefiniti `..` e `.`)
  - **esercizio:** verificare con esempi tutte le cose dette sopra; provare anche a fare l'hard link di un soft link e viceversa

- Qualche dettaglio in più su `touch [-a] [-m] [-t timestamp] {file}`
  - il suo "vero" uso è quelli di cambiare i timestamps negli attributi dei file dati (lo può fare)

  - li cambia normalmente tutti e 3, a meno che non si sia data l'opzione `-a` (solo atime) o `-m` (solo mtime)
  - come effetto collaterale, se un file dato non esiste lo crea
  - può essere usato anche su una directory (ma solo se esiste: come mai?)
  - con `-t timestamp` setta i timestamp del file al timestamp dato, anziché al timestamp del tempo attuale
  - **esercizio:** immaginare almeno due casi in cui touch non riesce a creare un file

- Comando `du [-c] [-s] [-a] [-h] [--exclude=PATTERN] [files...]`: fa il conto di tutte le dimensioni dei file e/o directories dati come argomento
  - se non ci sono argomenti, dà la dimensione della cwd
  - per le directory, considera tutti i file nel corrispondente sottoalbero
  - `-s` o `-a`: mostra solo il totale o tutti i files (di default, fa una cosa intermedia, mostrando tutte le eventuali sottodirectory)
  - `-c`, utile solo se c'è più di 1 argomento: fa la somma tra le dimensioni di tutti gli argomenti
  - `-h` (*human readable*), anziché stampare in bytes lo mostra arrotondato al kB, al MB, al GB
  - `--exclude=PATTERN`, toglie dal conto i file il cui nome soddisfa l'espressione regolare `PATTERN`: ci ritorneremo
  - **esercizio:** verificare le opzioni sopra riportate su alcune directory standard di Linux (ad es., `/etc`)

- Comando `df [-h] [-l] [file]`: mostra la dimensione e l'attuale uso dei filesystem
  - senza argomenti, li mostra tutti
  - altrimenti, mostra solo quello che contiene `file`
  - `-h`: human readable
  - `-l`: solo filesystem locali (ad es.: niente filesystem di rete)
  - **esercizio:** vedere le statistiche di uso di tutti i filesystem, e poi solo del filesystem contenente i seguenti files: `/etc/passwd` e `/proc`

- Comando `dd [opzioni]`
  - crea file in modo elaborato
  - parte da un file e ne crea un altro tramite "conversioni"
  - modo diverso di dare le opzioni per questo (e qualche altro) comando

- si tratta di una sequenza di assegnamenti `variabile=valore`
- le variabili più importanti sono:
  * `bs` dimensione di un singolo blocco in lettura/scrittura
  * `count` numero di blocchi da copiare
  * `convert`, in questo caso, il valore specifica una conversione, per esempio di codifica (da minuscolo a maiuscolo e viceversa, più svariate altre cose; vedere il man)
  * `if` file di input (se non dato, legge da tastiera)
  * `of` file di output (se non dato, scrive su schermo)
- a parte che per le conversioni, si usa soprattutto nei casi in cui la copia tramite `cp` non funzionerebbe
  * `/dev/zero`: se si prova a farsi stampare questo file, vengono fuori un numero infinito di zeri
  * è un file *speciale*, o meglio *character special file*, e non si trova su un disco, ma è connesso direttamente al kernel, che risponde appunto con un numero infinito di zeri quando si cerca di leggerlo; altro file simile: `/dev/urandom`, che risponde con un numero infinito di numeri pseudo-casuali
  * grazie a `dd` e alle sue opzioni, si può creare un file con un certo numero di zeri in modo semplice e veloce; ad es.: `dd if=/dev/zero of=test_file.zeri bs=1M count=10` crea un file da 10MB, tutto fatto di zeri
  * **esercizio:** provare ad aprire il file appena creato con `gedit`: contiene quello che ci si aspetta? se no, perché?
  * perché uno dovrebbe fare una cosa del genere, apparentemente stupida?
  * ad esempio, per cancellare completamente dati da un supporto di memoria
  * o per preparare un supporto di memoria (o meglio ancora un file) ad essere *formattato*
  * altro uso: copiare solo una parte di un file, grazie a `skip=`n, che salta n blocchi (quanti bytes ci sono in un blocco è dato dalla variabile `bs`)
  * analogamente, `seek=`m permette di non modificare i primi m blocchi del file destinazione (nel caso il file destinazione non sia vuoto, o abbia almeno quel numero di blocchi)
  * **esercizio:** creare un file di testo usando `gedit`, con il seguente contenuto:

    ```
    ciao1
    addio2
    via3
    ehila4
    dove vai5
    ```

e poi copiarlo in un altro file in modo tale che quest'ultimo contenga solo i bytes che vanno dal decimo al ventesimo; usare sia 1 che 10 come valori per `bs`; rifare nuovamente la copiatura sullo stesso file destinazione, ma questa volta fare in modo che il contenuto del file destinazione sia duplicato (ovvero, contenga per 2 volte consecutive i bytes che vanno dal decimo al ventesimo)

- Comando `mkfs [-t type fsoptions] device`
  - crea un filesystem su un device
  - spesso si parla di questa operazione come di "formattazione", perché implica preparare il device a memorizzare files secondo il particolare formato
  - `type` può essere ext2, ext3, ext4, vfat, ntfs, ... (vedere Tabella 1 nella lezione 2)
  - `fsoptions`: si può specificare se si vuole che sia in sola lettura (`ro`) e anche in scrittura (`rw`)
  - device può essere o un file speciale chiamato appunto *device file* (uno di quelli contenuto nella directory `/dev`) oppure anche un file regolare
    - se si tratta di un file regolare, spesso è creato con `dd` come descritto sopra
    - se si tratta di un device, allora non dev'essere stato già montato (vedere più avanti)
    - device tipici: `/dev/hda` è spesso l'hard disk primario; `/dev/hda2` una partizione dell'hard disk primario; `/dev/sdb1` una penna USB, ...
  - spesso mkfs non è facilmente disponibile per tutti gli utenti (di solito, viene usato solo dall'amministratore); in caso, si può usare `/sbin/mkfs`
  - per esempio, volendo creare un filesystem ext3 di 100 MB su un file, si può fare così:

    ```
    dd if=/dev/zero bs=1M count=100 of=fs-virtuale
    ls -l fs-virtuale
    /sbin/mkfs -t ext3 fs-virtuale
    ```

  - Per poter usare un filesystem formattato occorre "ancorarlo" ad una qualche directory: comandi `mount [-t fstype] [-o opzioni] device mountpoint` e `umount [-a] mountpoint|device`
    - per il device, vale la stessa cosa detta per `mkfs`
    - mountpoint è una directory del filesystem
    - bisogna sempre anteporre `sudo` ai comandi di mount ed umount
    - la pipe `|` sta per "oppure"

– provare uno alla volta i seguenti comandi, stando ad esempio nella home:

```
dd if=/dev/zero bs=1M count=100 of=fs-virtuale
ls -l fs-virtuale
/sbin/mkfs -t ext3 fs-virtuale
mkdir dir
sudo mount -t ext3 fs-virtuale dir
cd dir
ls -l
dd if=/dev/zero bs=1M count=2 of=file1
cd ..
ls -l
rm -rf dir
cd dir
dd if=/dev/zero bs=1M count=98 of=file2
cd ..
umount dir
ls -l dir
rm -rf dir
sudo mount -t ext3 fs-virtuale dir
cd dir
ls -l
umount .
cd ..
umount dir
```