Dati

mer 01/03/2017
Le informazioni sugli utenti sono nel file `/etc/passwd`, scritte con il campo `utente:x:userId:groupId:description:userDirectory`
La `x` è un campo storico, perchè prima c'era scritta la password. Ora invece le password vengono tenute nel file `/etc/shadow`, disponibile solo a root.
Le informazioni dei gruppi sono nel file `/etc/group`.
Il **link simbolico** è un file, dove ci sta scritto l'indirizzo di un altro file (o directory). Se io cancello il file puntato, non succede nulla; il link continua a puntare a un file non esistente.
Un **link hardware** invece è una situazione particolare in cui ho due path che puntano allo stesso file con path diversi ma con lo stesso inode. Quando cancello uno di questi file, il link count diminuisce di uno. Se è minore o uguale a zero, allora il file non esiste più nel sistema. In pratica l'hardlink sarebbe un file con nome diverso ma con inode uguale rispetto a un altro file. Ogni cartella ha due hardlink standard, cioè `"."` e `".."`.
Con i link software, potrebbe succere che il file puntato non esiste. Non può succedere con i link hardware.
Ogni file ha 4 terne per desciverne l'accesso: `sSt,rwx,rwx,rwx`. Il primo bit, il `setuid` bit, significa che quando viene eseguito il processo che lo esegue, quel processo ha i privilegi del propietario del file. Questo consente all'utente normale di eseguire operazione di root controllate attraverso binari. Esempio: `/usr/bin/passwd`. Invece il secondo, il `groudid` bit, permette l'accesso al file agli utenti che appartengono agli stessi gruppi secondari del propietario. Il terzo lo `sticky` bit, significa che quel file deve essere lasciato in memoria una volta caricato. Nei sistemi moderni non viene quasi più usato. Le altre tre terne sono i permessi del propietario, del gruppo del propietario e degli altri. L'ultimo bit, l'esecuzione di "altri", può essere usato per mappare anche la terna sSt: se è s, allora significa suid o sgid + eseguibile. Se è t, allora significa sticky + eseguibile. S significa suid o sgid - eseguibile; T significa sticky - eseguibile.
La directory `tmp` è una directory speciale: tutti hanno permessi in quella directory. (Cioè possono cancellare e inserire file, ma non modificarli, perchè ogni file ha i suoi permessi.)
Quando faccio `ls -la` ho anche il tipo di file, che può essere -(file normale), d(directory), p(file a pipe), l(link), c(boh), b(file a blocchi)
Un comando utile per vedere tutte le informazione su un file è `stat`, che mi dice tutte le cose esplicitamente.
Alcuni comandi molto comuni non hanno la man page. Questo è perchè sono funzionalità interne al terminale (bash). Quindi dobbiamo fare `man bash` per vedere le cose relative ad essi.

06/03/2017
Usare il comando `stat -f` per avere informazioni sul filysistem invece che sul file singolo.
Imparare a usare gli alias.
Con `mkdir -p` si posso creare alberi di cartelle.
Con il comando `touch -t` si possono cambiare le date di accesso / modifica dei singoli file.
Una directory eseguibile significa che è possibile accedere(leggere) il suo contenuto.
Il comando `cp -u` copia solo i file la cui data di sorgente è più recente della data di destinazione (utile per sincronizzazioni). Il comando `cp -a` serve a preservare le modalità di accesso del file sorgente in quello creato.

13/03/2017
Certi comandi vengono riconosciuti dalla shell (bash) e eseguiti come builtin, come `cd`, cioè non sono programmi presenti dentro /bin. Questo implica che non viene creato un nuovo processo, ma è lo stesso processo della shell che esegue quel comando. Per capire se un comando è builtin oppure no, allora basta fare `type cd`.
Quando faccio `top`, ogni processo ha tre coppie di informazioni di userID: real (l'utente che ha fatto partire il programma), effective (l'utente che il programma sta impersonando, cioè per esempio è root se eseguo passwd), saved(non ho capito bene.). Inoltre ho anche le informazinoi `cwd`, cioè la cartella su cui sta lavorando, `umask`, ?, e `niceness`, un valore indirettamente proporzionale alla priorità del processo. Ogni utente può solo aumentare la niceness dei suoi processi, non la può abbassare. (quindi ogni processo può solo cedere priorità.). Inoltre top fornisce anche lo stato del processo, che può essere (running, runnable, sleeping, zombie e stopped).
`init` diventa il padre di tutti i processi orfani.
I segnali sono lo strumento rudimentale di coordinazione fra processi. Ogni programma definisce all'inizio il comportamento che assumerà all'arrivo di ogni segnale. Questa parte si chiama installazione dei segnali. Tuttavia, non posso ridefinire tutti i segnali, come SIGKILL, che mi chiude perforza il processo..
Il comando `ps` lista i processi del terminale corrente. Il comando `ps -e` lista tutti i processi del sistema. Il comando `ps -efl` lista tutti i processi del sistema dando tantissime informazioni per ogni processo.
Possiamo eseguire un comando in background mettendo la `&` alla fine del comando.