Sistemi Operativi

AAF - Secondo anno - 3CFU

A.A. 2020/2021

Corso di Laurea in Matematica

Linux, bash e comandi

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

Argomenti trattati

1. Per cominciare
   - Informazioni iniziali
   - Comandi di base
2. Utenti, filesystem e file
   - Amministrazione degli utenti
   - Il filesystem
   - Comandi per la gestione di file e directory
   - Filesystem e comandi per la gestione

Per cominciare

Informazioni iniziali

Terminale e shell

- Per dare comandi dalla riga di comando si può usare il **terminale**
- In questo modo viene utilizzata la **shell**, che funziona appunto su un terminale
- Quando si chiama il terminale viene subito avviata la **shell standard** (ad esempio la *Bourne again shell*, *Bash*)
- La *shell* è un programma che svolge la funzione di interfaccia tra sistema e utente: comprende un interprete della riga di comando, accoglie gli input utente tramite tastiera (cioè nella riga di comando) e li analizza, avvia i programmi e restituisce all'utente il risultato sotto forma di testo
- Ogni shell possiede un proprio linguaggio di programmazione che consente di scrivere *script shell*

Standard input, output ed error

- In Linux si usano i concetti di standard input, output ed error, cioè `stdin, stdout e stederr`
- Quando viene eseguito un comando vengono generati i tre *flussi* (o stream) standard, dove per *flusso* si intende un meccanismo usato per trasferire dati, in questo caso testo
- `stdin` è lo standard input stream e accetta testo come input
- `stdout` è lo standard output stream ed è il testo prodotto in output dal comando eseguito, di solito scritto sullo schermo
- `stderr` è lo standard error stream ed è visualizzato sullo schermo tramite messaggi di errore
- Come per i file, agli standard stream vengono associati degli identificatori: 0 per `stdin`, 1 per `stdout` e 2 per `stederr`

Utenti, filesystem e file

Comandi di base

Comando `man`

- `man [opzione] nomecomando` apre le pagine del manuale (man pages) nel terminale
- Le *man pages* di Linux sono suddivise in 10 tematiche:
  - (1) Comandi utente
  - (2) Collegamenti del sistema
  - (3) Funzioni del linguaggio di programmazione C
  - (4) Formati dei file
  - (5) File di configurazione
  - (6) Giochi
  - (7) Varie
  - (8) Comandi per l'amministrazione del sistema
  - (9) Funzioni del kernel
  - (10) Nuovi comandi
- Ad esempio, sia usando `man clear` che (restringendo la ricerca) usando `man 1 clear`, si apre la pagina del manuale riguardante il comando `clear`

Comandi `whatis` e `apropos`

- `whatis [opzioni] parolachiave` cerca le parole chiave nel manuale (o meglio nel database whatis)
- Se la parola cercata è presente nel manuale, `whatis` ne fornisce una breve descrizione nel terminale
- Sono visualizzate solo le corrispondenze con parole intere
- `apropos stringa` cerca una o più stringhe nel database whatis
- A differenza di whatis, sono visualizzate tutte le corrispondenze
- Ad esempio `apropos keyboard` visualizza le righe del database whatis contenenti la stringa keyboard

Comandi `history` e `clear`

- `history` mostra i comandi eseguiti
- Su Bash vengono memorizzati nella cronologia (*history*) gli ultimi 500 comandi inseriti nella riga di comando.
- Consente di ricercare nella lista dei comandi precedenti con i tasti freccia ed eseguirli di nuovo confermando con il tasto di invio

- `clear` serve a rimuovere il contenuto dello schermo
- Si ottiene un terminale vuoto con aperta solo la finestra della riga di comando
- Gli input immessi precedentemente rimangono comunque memorizzati nello *scrollback buffer*

Comandi `help` e `info`

- `help` mostra una lista dei comandi shell integrati (comandi built-in)
- `help comandoshell` fornisce una descrizione del corrispettivo comando
- Molti comandi accettano anche l'opzione -h (o - -help) che fornisce una breve descrizione sull'utilizzo del comando e delle sue opzioni

- `info comando` fornisce informazioni estese sul comando
- Nella maggior parte dei casi si hanno le informazioni che si possono richiamare tramite man, ma con collegamenti che agevolano la navigazione nel manuale

Utenti, filesystem e file

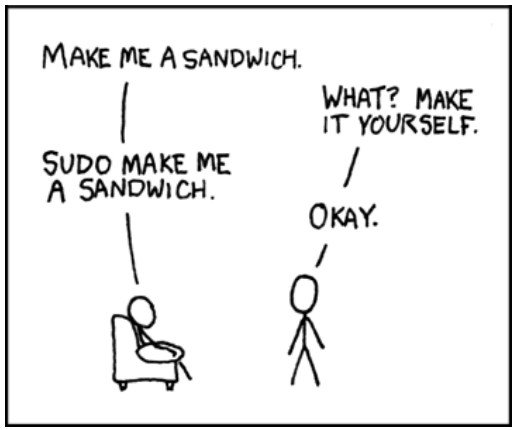
Amministrazione degli utenti

Utenti

- Durante l'installazione di un qualsiasi Linux, è necessario specificare (almeno) un **utente**
  - non tutti gli utenti possono fare login (per esempio, in molti sistemi Linux non può farlo l'utente `root`)
  - l'installazione di Linux crea (almeno una decina di) utenti, che servono solo per scopi interni del sistema operativo
- Ogni utente appartiene ad un **gruppo** (insieme di utenti)
  - viene tipicamente creato un gruppo con lo stesso nome dell'utente *principale* specificato in fase di installazione
- Esistono molti gruppi e uno stesso utente può appartenere a più gruppi: per sapere quali, si può usare il comando `groups [nomeutente]`
  - molti gruppi servono solo a scopi interni al sistema operativo

Utenti

- Nel caso di Ubuntu, l'utente creato a tempo di installazione è un **sudoer**, cioè un *utente con privilegi di amministratore di sistema*, e appartiene al gruppo predefinito `sudo`
  - quindi può eseguire comandi da *superutente* semplicemente preponendo il comando `sudo`, derivato da **s***uper* **u***ser* **do**: per esempio, può installare nuovi pacchetti
  - `sudo comando` è un comando particolare, che prende come argomento un altro comando, che può avere svariati argomenti



Utenti

- È possibile creare un altro utente usando il comando `adduser nuovoutente`
  - di default, questo crea un utente *non* sudoer
  - però il comando va dato da un utente sudoer, che deve anche aggiungere `sudo` all'inizio
  - infatti creare utenti rientra nelle prerogative di un amministratore di sistema

- **Esercizio** creare due nuovi utenti non-sudoer, con nome `utente1` ed `utente2`

Utenti

- È possibile aggiungere un utente già creato ad un gruppo, usando il comando `adduser utente gruppo`
  - dalla pagina di manuale: If called with two non-option arguments, adduser will add an existing user to an existing group.
  - di nuovo, va fatto dall'utente sudoer, con `sudo` davanti
- È possibile cambiare utente usando il comando `su -l nomeutente`

Utenti

- **Esercizio**:
  - Con l'utente sudoer, provare ad eseguire il comando `apt-get update` e il comando `sudo apt-get update`
  - Con l'utente `utente1`, provare ad eseguire `apt-get update` e il comando `sudo apt-get update`
- APT (Advanced Package Tool), sistema di gestione dei pacchetti predefinito in Ubuntu
- Il comando `apt-get update` aggiorna la lista dei pacchetti disponibili dai repository; andrebbe eseguito periodicamente per verificare che la propria lista di pacchetti sia aggiornata
- Il comando `apt-get install` installa il pacchetto specificato

Utenti

- È possibile cambiare o impostare la propria password o la password di un utente usando il comando `passwd`
  - `sudo passwd nuovoutente` consente di impostare la password dell'utente nuovoutente
  - `passwd` consente di cambiare la propria password
- È possibile cancellare un utente usando il comando `deluser utente`
  - di nuovo, va fatto dall'utente sudoer, con `sudo` davanti

Utenti, filesystem e file

Il filesystem

Filesystem in Linux

- Un **filesystem** è un'organizzazione di un'area di memoria (tipicamente di massa, come il disco), basata sul concetto di *file* e di *directory*
  - una directory serve a contenere al suo interno altre directory oppure file
  - induce naturalmente una struttura gerarchica, ad albero, dove ogni nodo è una directory o un file
  - solo le directory possono avere figli
  - i file *regolari* contengono sequenze di bit dell'area di memoria sulla quale c'è il filesystem e possono essere testi, dati, programmi sorgente, eseguibili
  - file *speciali* sono directory, device (dispositivi hardware collegati visti come file), pipe (file per lo scambio di dati sincrono tra due processi concorrenti, link (riferimento ad un altro file o directory)

Filesystem in Linux

- Linux ha un solo filesystem principale, che ha come radice la directory / (cioè la directory *root*)
  - tutti i file e le directory sono contenuti, direttamente o indirettamente, in tale directory
  - le foglie dell'albero possono essere directory vuote oppure file
  - all'interno della stessa directory non ci possono essere due file, due directory oppure un file e una directory con lo stesso nome
  - cambiare le maiuscole/minuscole è sufficiente a distinguere tra due files o directory: `nomeFile` è diverso da `nomefile`

Filesystem in Linux

- Ogni file o directory è raggiungibile dalla directory radice attraverso un *path assoluto*
  - una sequenza di directory separate da slash e avente slash come primo carattere
  - (quindi, il carattere slash non può essere usato per dare un nome ad una directory o ad un file)
  - esempio `/home/utente1/dir1/dir3/dir7/file.png`
  - come parziale eccezione, è un path assoluto anche quello che comincia con una tilde $\sim$
  - infatti, come vedremo, la tilde è una scorciatoia per la directory home dell'utente corrente x: `/home/`x

Directory principali

- Alcune delle principali directory e loro utilizzo su Ubuntu
  - `/bin` Contiene i programmi basilari per la gestione del sistema, cioè buona parte dei comandi base utilizzabili dalla riga di comando da qualsiasi utente senza dover utilizzare i privilegi dell'amministratore
  - `/boot` Contiene le immagini del kernel e i file indispensabili al bootstrap del sistema
  - `/dev` È la directory che individua sotto forma di file le periferiche hardware
  - `/etc` Contiene i file di configurazione del sistema. Ad esempio `/etc/apt` file di configurazione dei repository
  - `/home` Contiene tutte le directory personali degli utenti del sistema
  - `/lib` Contiene tutte le librerie condivise del sistema

Directory principali

- Alcune delle principali directory e loro utilizzo su Ubuntu
  - `/root` Contiene la directory home dell'amministratore del sistema ed è esplorabile solo utilizzando i privilegi da super utente. Il contenuto è analogo a quello delle singole home-utente descritte nel capitolo della directory /home
  - `/tmp` Contiene file temporanei
  - `/usr` È la directory che contiene la maggior parte dei programmi installati sul sistema
    - `/usr/bin` file eseguibili delle applicazioni accessibili a tutti gli utenti, cioè i programmi normalmente avviabili dal menù delle applicazioni.
    - `/usr/sbin` file eseguibili delle applicazioni di sistema accessibili solo all'amministratore
    - `/usr/share` file di vario genere (configurazione, documenti di testo, ecc..) indipendenti dall'architettura del sistema (i386, amd64). Ad esempio, le cartelle backgrounds, icons e themes contengono sfondi, icone e temi del desktop

Directory

- Concetto di *current working directory* (cwd)
  - vale per ogni processo, quindi anche per le shell, che la mostrano nel prompt
  - per sapere qual è la cwd, usare il comando `pwd`
  - per cambiare la cwd, usare il comando `cd [path]` (se non si specifica il `path`, la nuova directory sarà la home)
  - all'interno di `path` può essere usato
    - sia `..` (directory *parent*, che contiene quella attuale; se fatto sulla root, ritorna la stessa root),
    - oppure anche `.` (la directory stessa)
    - il path `/home/utente1/dir1/dir3/dir7/file.png` può essere equivalentemente scritto, ad esempio, `/home/./utente1/dir1/./dir3/dir7/file.png` oppure `/home/utente1/../utente1/dir1/dir3/dir7/file.png` oppure `/home/utente1/dir1/../dir1/dir3/dir7/file.png`
- **Esercizio:** posizionarsi nella directory `/lib` e controllare come cambia il path nel prompt

Directory

- A partire dalla cwd, si possono usare i *path relativi*
  - sono quelli *non* assoluti; pertanto, non cominciano con uno slash (né con la tilde)
  - per esempio: se la cwd è la home dell'utente `utente1`, allora lo stesso file di sopra è raggiungibile con il path relativo `dir1/dir3/dir7/file.png`, o anche `./dir1/dir3/dir7/file.png`, o anche `../utente1/dir1/dir3/dir7/file.png`
  - a seguito di un `cd dir1/dir3` (o equivalentemente, `cd /home/utente1/dir1/dir3/`), lo stesso file di sopra è raggiungibile con il path relativo `dir7/file.png`, o anche `./dir7/file.png`, o anche `../dir3/dir7/file.png`
- La differenza tra un path assoluto ed uno relativo sta nel fatto che il path assoluto è valido qualsiasi sia la cwd, mentre il path relativo può non essere valido quando si cambia la cwd

Utenti, filesystem e file

Comandi per la gestione di file e directory

Comando `ls`

- Il comando `ls [-la] [-R] [nomedir]` mostra il contenuto della directory `nomedir` (o della cwd, se `nomedir` non è dato)
  - `ls -l` mostra informazioni relative al contenuto della directory
  - `ls -a` mostra i file considerati *nascosti* (*hidden*), cioè i file con nomi che cominciano con il punto
  - `ls -R` mostra tutto il sottoalbero con radice in `nomedir`
  - `ls -s` mostra la dimensione dei file

Comando `mkdir`

- Il comando `mkdir [-p] nomedir` crea la directory `nomedir` (vuota)
  - Se si usa l'opzione `-p` e se `nomedir` è un path con più di una directory, allora crea tutte le directory nel path (se non esistono)
  - ad esempio, `mkdir -p dir11/dir13/dir15`, supponendo che `dir11` esista già, creerà la directory `dir13` dentro `dir11`, e poi `dir15` dentro `dir13`
  - **Esercizio** provare a vedere cosa succede, nella stessa situazione, senza l'opzione `-p`

- **Esercizio**
  - creare l'intero albero di directory dato sopra (ovvero, `/home/utente1/dir1/dir3/dir7/`), posizionarsi dentro `dir1` e poi in `dir7` sia usando che non usando la directory parent `..`
  - controllare il risultato usare il comando `ls`;
  - controllare come cambia il path riportato nel prompt

Comando `touch`

- Il comando `touch nomefile` crea il file `nomefile` (vuoto)
- Per modificare un file, scrivendoci un testo dentro, dare il seguente comando `geany nomefile &`; si aprirà un editor grafico standard, dove è possibile scrivere e salvare le modifiche (il carattere `&` serve per esecuzione in background)
- Alternativamente, si possono usare editor che si interfacciano direttamente con il terminale (ma non sempre sono installati): `nano nomefile` oppure `pico nomefile` (attenzione: *non usare* il carattere `&`). Più complicato: `vi nomefile`
  - se avete provato ad eseguire `vi nomefile` e non sapete come uscire, digitate i caratteri `:q` seguiti da invio)

Altri file

- Per creare file non di testo, occorre usare opportuni altri programmi (a seconda di cosa serve)
  - ad esempio, i vari applicativi di LibreOffice possono essere usati per creare file contenenti documenti formattati (DOC, DOCX, ODT, etc), fogli di calcolo (XLS, XLSX, ODS, etc)
  - basta usare il comando `libreoffice nomefile`, che però non è installato sul disco virtuale fornito (ma per installarlo basta scrivere `sudo apt-get install libreoffice`)
  - attenzione: `nomefile` deve già esistere; altrimenti si può eseguire `libreoffice` senza argomenti e poi usare l'interfaccia grafica per creare un nuovo documento del tipo desiderato

Comando `tree`

- Si può visualizzare un intero albero di directory con il comando `tree [-a] [-L maxdepth] [-d] [-x] [nomedir]`
  - potrebbe non essere installato: `sudo apt-get install tree`
  - in generale: se si dà un comando sbagliato, e l'output sembra non finire mai, provare a premere CTRL+c
  - usare l'opzione `-L` per limitare la profondità dell'albero mostrato
  - attenzione: l'output contiene anche caratteri ASCII non-standard, per visualizzare la struttura dell'albero. Si tratta di caratteri UTF-8 a 3 bytes per carattere (vedere `http://www.fileformat.info/info/unicode/utf8.htm`)

Utenti, filesystem e file

Filesystem e comandi per la gestione

Filesystem di Linux e altri filesystem

- Il filesystem di Linux è unico, ma può contenere elementi eterogenei
  - il disco, ovviamente
  - potrebbe esserci però più di un disco, ad esempio uno tradizionale e uno a stato solido
  - potrebbe esserci un disco solo, ma partizionato; in questo caso, ogni partizione può trovarsi in punti diversi del file system
  - filesystem virtuali, montati dal kernel per gestire le risorse
  - filesystem di rete
  - filesystem in memoria principale (RAM)

Comando `mount`

- Il comando `mount` permette di *aggiungere* (o *montare*) un filesystem (di un dispositivo esterno o di un'altra partizione del disco rigido) agganciandolo a una directory - stratagemma del **mounting**
- Serve a rendere accessibili ai programmi e agli utenti del sistema, file e directory contenuti nel *nuovo* filesystem
- Una qualsiasi directory dell'albero gerarchico può diventare il punto di **mount** per il *nuovo* filesystem

Comando `mount`

- Più formalmente:
  - una directory x del filesystem di Linux si dice *punto di mount* per un altro (nuovo) filesystem F se e solo se la directory root di F diventa accessibile a partire da x
  - così, il comando `ls` x mostrerà il contenuto della directory root di F
  - leggere o modificare un file dentro il sottoalbero radicato in x avrà l'effetto di leggerlo o modificarlo nella corrispondente directory di F
  - quindi, un path assoluto p dentro il filesystem F diventa il path assoluto x/p dentro Linux (supponendo che x sia anch'esso un path assoluto)

Comando `mount`

- È meglio scegliere una directory x vuota, altrimenti il suo *vecchio* contenuto non sarà più visibile fino all'`umount`
  - **ma** il contenuto di x sul disco non viene cancellato, semplicemente non è più accessibile (né in lettura né in scrittura) tramite `ls` ed altri comandi per il filesystem
  - più precisamente: se x si trovava, prima del mount, nel filesystem $G \neq F$, allora i contenuti di G relativi alla directory di x vengono nascosti (ma non cancellati), e si mostrano quelli di F a partire dalla sua root

Comando `mount`

- **Ad esempio**:
  - Sulla directory `/proc`, durante il boot, viene montato un filesystem virtuale (non corrisponde ad alcun disco)
  - I filesystem di rete potrebbero essere montati su una directory `/nfs`, creata appositamente
  - Un filesystem in memoria principale può essere montato su una directory all'interno della home di un utente
  - Il disco principale, contenente l'installazione del sistema operativo, sarà montato su `/`
  - Un disco secondario, senza l'installazione del sistema operativo (o con installato un altro sistema operativo!), può essere montato su `/windows`

Filesystem, directory e dischi

- Anche se c'è un solo disco, è possibile *partizionarlo*: una parte si prende il sistema operativo (e viene montato su `/`) e una parte si prende la directory home (e viene montato su `/home`)
  - utile se poi si vuole installare un altro sistema operativo da capo: si installa sulla partizione che era montata su `/`, si monta la home così com'è su una qualche directory, e ci si risparmia di dover copiare i vecchi dati della home: sono già lì
    - se si tratta semplicemente di un'altra distribuzione Linux non c'è problema
    - se invece si passasse a Windows, usare il filesystem `home` sarebbe complicato (Windows non riconosce nativamente i tipi di filesystem di Linux, come `ext2` o `ext3`)

Filesystem, directory e dischi

- **Attenzione** partizionare un disco implica cancellare i dati precedenti
- Per essere più precisi: partizionare ulteriormente una partizione di un disco cancella i dati presenti in quella partizione
- Programmi per partizionare dischi: `gparted` (grafico), `parted` ed altri (testuali)
- Nei sistemi Linux ci sono sempre almeno due partizioni: una montata su `/` e una di tipo particolare usata per lo *swap* (memoria virtuale)

Tipi di filesystem

- Principali caratteristiche del **tipo** di filesystem
  - Dal punto di vista dell'**utente**: dimensione massima di una partizione, dimensione massima di un file, lunghezza massima di un nome di file, se è journal o meno
  - *journal* vuol dire che ogni modifica viene scritta su un file speciale, e applicata su disco in un secondo tempo (meglio come prestazioni e come resistenza ad alcuni tipi di fault)
  - Dal punto di vista del **progettista-programmatore** di sistemi operativi, il tipo definisce anche la metodologia con cui i dati vengono letti/scritti sul disco

Tipi di filesystem

- I tipi di filesystem sono relativamente pochi: ci sono quelli di Windows (NTFS, MSDOS, FAT32, FAT64) e svariati per Linux
- Per ognuno di questi tipi, ci sono le caratteristiche dette sopra
- Ogni disco, o meglio ogni partizione, va inizialmente *formattata* con uno di questi tipi: ovvero, occorre dichiarare con quale tpo di filesystem si vuole usare quella partizione
- quando un disco, o una sua partizione, viene montata, occorre specificare il giusto filesystem (quello con cui è stato formattato)

Tipi di filesystem

- I principali tipi di filesystem di un sistema Linux

Nome	Journal	Partiz (TB)	File (TB)	Nome file (bytes)
Ext2	No	32	2	255
Ext3	Sì	32	2	255
Ext4	Sì	1000	16	255
ReiserFS	Sì	16	8	4032

Comando `cat`

- Per sapere quali filesystem sono montati e dove: comando `cat /etc/mtab`, oppure anche `mount` (senza argomenti)
- Il comando `cat [nomefile]`: scrive a schermo il contenuto di `nomefile`
  - senza argomenti, resta in attesa: se si scrive qualcosa e poi si preme invio, ripete quanto scritto, finché non si preme CTRL+d, che è il carattere EOF (*end-of-file*)
  - funziona bene solo se il file è di testo (e se usa la codifica riconosciuta da `cat`), altrimenti scrive caratteri incomprensibili
  - comando `mount`, è quello da usare per fare il *mounting* descritto prima
- Per sapere quali filesystem vengono montati a tempo di boot (esclusi quelli gestiti dal kernel): `cat /etc/fstab`

- Tabella delle directory di primo livello di un sistema Linux

Directory	**Spiegazione**	**Montata**
/boot	Kernel e file di boot	NO
/bin	Binari (programmi eseguibili) di base	NO
/dev	Devices (periferiche) hardware e virtuali	boot
/etc	File di configurazione di sistema	NO
/proc	Dati e statistiche dei processi e parametri del kernel	boot
/sys	Informazioni e statistiche di device di sistema	boot
/media	Mountpoint per device di I/O (es: CD, DVD, USB pen)	quando necessario
/mnt	(come /media)	quando necessario
/sbin	Binari di sistema	NO
/var	File variabili (log file, code di stampa, mail ...)	NO
/tmp	File temporanei	NO
/lib	Librerie	NO

File di info per utenti e gruppi

- Alcuni file importanti: `/etc/passwd` e `/etc/group`
  - il primo file elenca tutti gli utenti, il secondo tutti gruppi
  - entrambi i file sono un esempio della filosofia Linux: si usano file di testo (con codifica ASCII a 8 bit) con una struttura definita e conosciuta dai programmi che devono interagire con quei file
  - ad esempio, `adduser` conosce la struttura di entrambi i file
  - questi due file, come molti altri, sono organizzati a *righe*, dove per *riga* si intende una sequenza di caratteri terminati dall'andata a capo LF (*line feed*, carattere 0x0A ASCII)

File di info per utenti e gruppi

- **Esercizio** far scrivere a schermo il contenuto di tali file
  - Le righe che iniziano con il carattere # sono da intendersi come commenti, e vengono ignorate dai programmi che leggono/scrivono tali file
  - Ogni riga è formata da campi separati dal caratteri speciale : (che, quindi, non può essere usato per definire un nome utente)
  - per `/etc/passwd`, i campi sono i seguenti: `username:password:uid:gid:gecos:homedir:shell`
  - per `/etc/group`, i campi sono i seguenti: `groupname:password:groupID:lista_utenti` (dove la lista degli utenti è separata da virgole, e quindi neanche le virgole possono comparire in un nome utente)
  - **Esercizio** verificare che `utente3` non sia presente in `/etc/passwd`, crearlo, e poi controllare che sia presente
  - **Esercizio** verificare che `utente3` non sia nel gruppo `sudo`, aggiungerlo al gruppo `sudo` e poi controllare che sia presente