Sistemi Operativi

AAF - Secondo anno - 3CFU

A.A. 2020/2021

Corso di Laurea in Matematica

Linux, bash e primi comandi

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

SAPIENZA UNIVERSITÀ DI ROMA

Argomenti trattati

Unix

Caratteristiche di Unix

Caratteristiche principali

Multiutente: più utenti contemporaneamente possono usare lo stesso computer grazie a Unix

Multiprocesso: lo stesso utente può lanciare contemporaneamente più di un processo

File system gerarchico: il file system è organizzato come un albero, dove ogni nodo interno è una directory e ogni foglia è un file o una directory

- è possibile aggiungere file system aggiuntivi (ad esempio, una chiave USB o un CD): viene "montato" su una qualche directory

Caratteristiche principali

Kernel: gestisce memoria (principale e secondaria), processi, I/O, risorse hardware in generale

System calls: funzioni C che possono essere chiamate se ci si vuole interfacciare con il kernel (ad es., per creare un file...)

Shell: programma interattivo che accetta comandi da "girare" al kernel (del tipo: mostra il contenuto di una directory)

Altro:

- *Ambienti di programmazione:* per permettere di scrivere programmi, tipicamente in C
  - compilatore, debugger, text editor
  - i programmi scritti in linguaggi interpretati (ad es., Python o Java) non vengono eseguiti direttamente, ma appunto tramite l'interprete
  - quindi è come se ci fosse un ulteriore "velo", che con il C è rimosso
  - pertanto, il linguaggio principe per "dialogare" direttamente con il kernel è il C

Caratteristiche principali

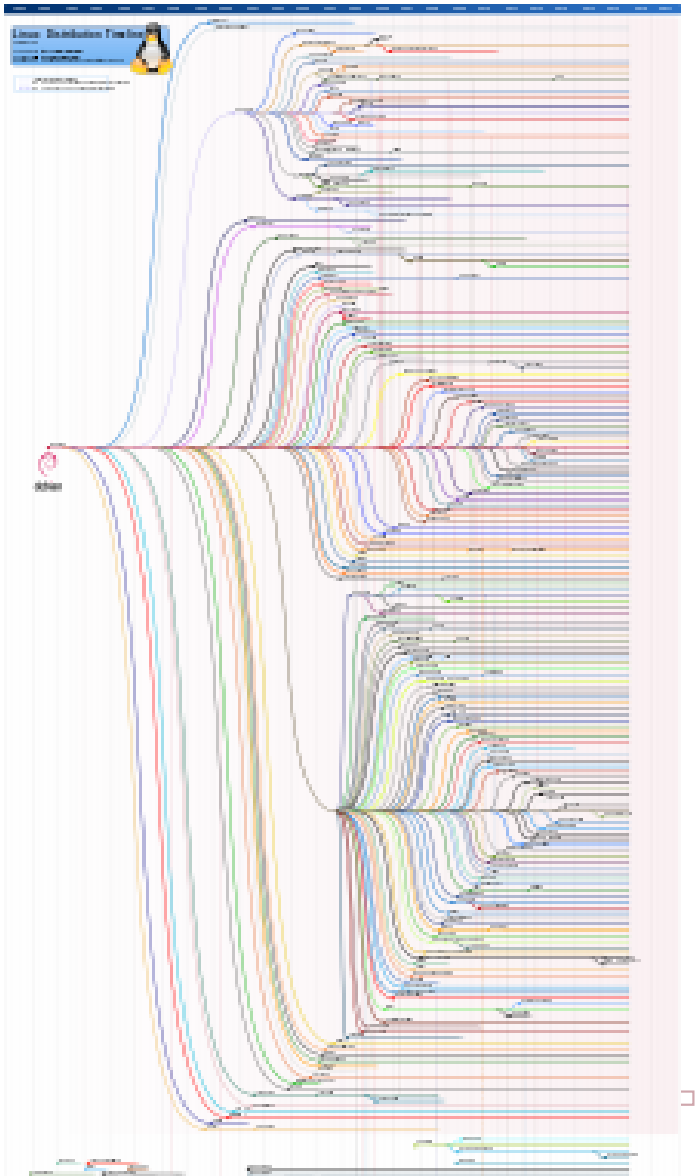
Altro:

- *Utilities:* altri programmi, per fare qualsiasi cosa che sia computabile
  - suite Office (OpenOffice, LibreOffice)
  - lettore PDF (Acrobat Reader, evince, ...)
  - browser (Firefox, Chrome, ...)
  - messaggistica (Skype, ...)
  - riproduttore audio/video (VLC, ...)
  - editor di immagini (xfig, gimp, ...)
  - giochi (semplici!)
  - ...
- *Modularità:* programmi installabili a pacchetti, moduli del sistema operativo attivabili e disattivabili

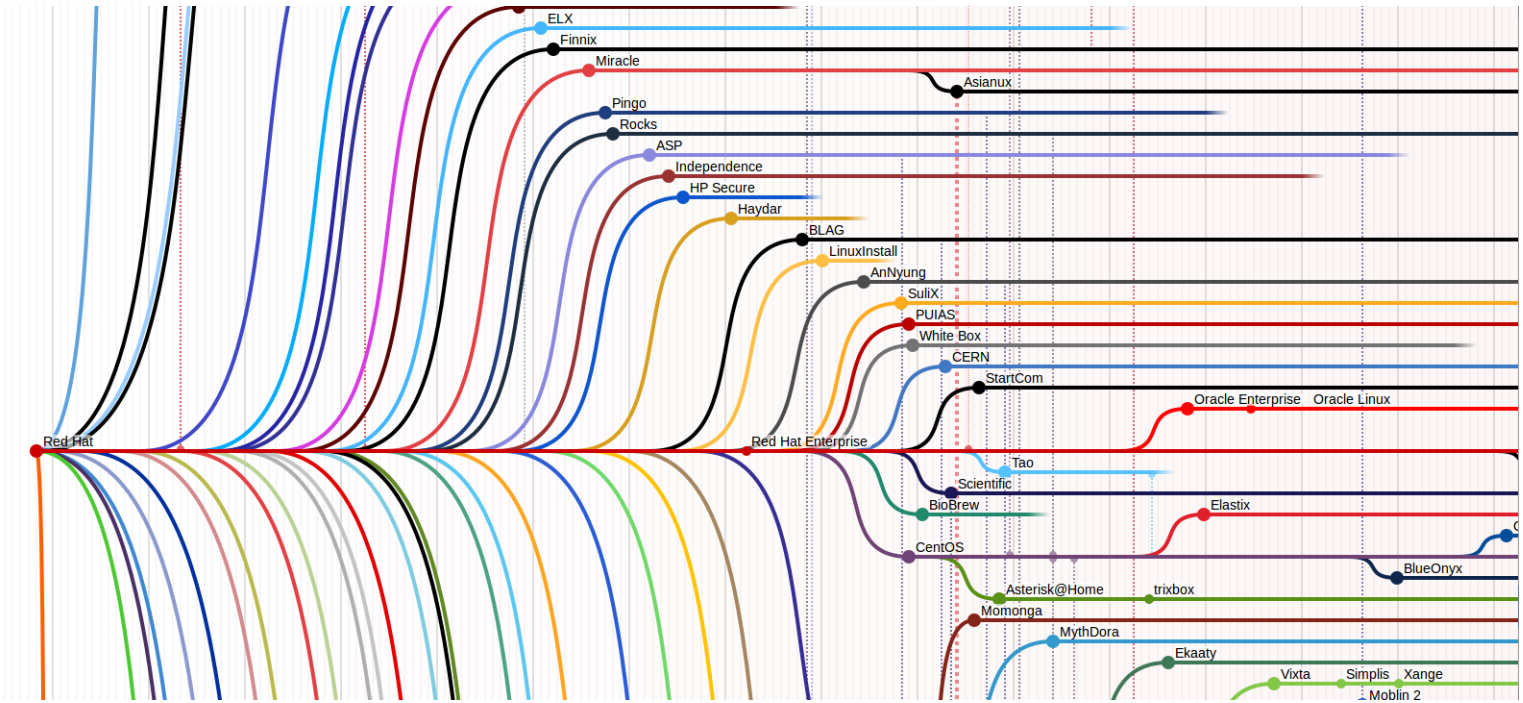
Linux

Le versioni di Linux

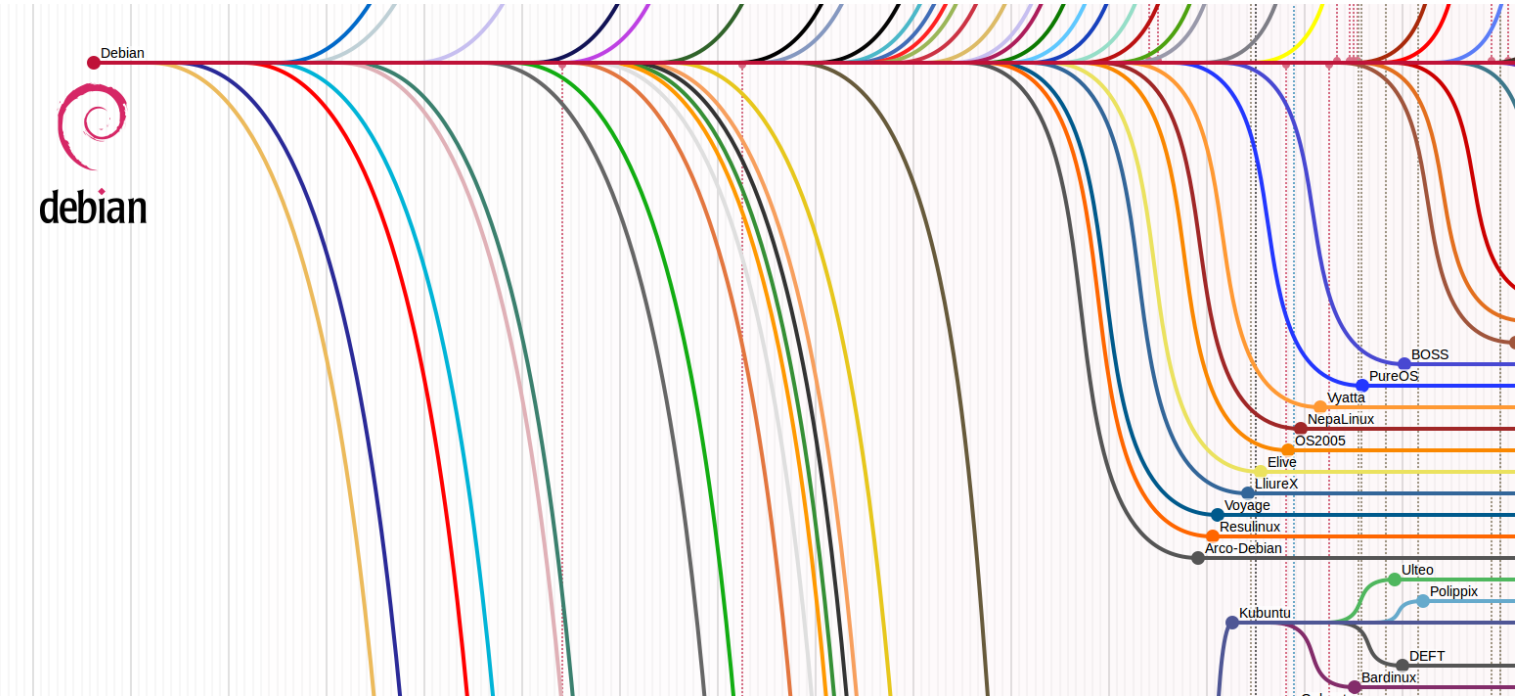
- Ci sono moltissime versioni
- Timeline delle distribuzioni di Linux, da Wikipedia



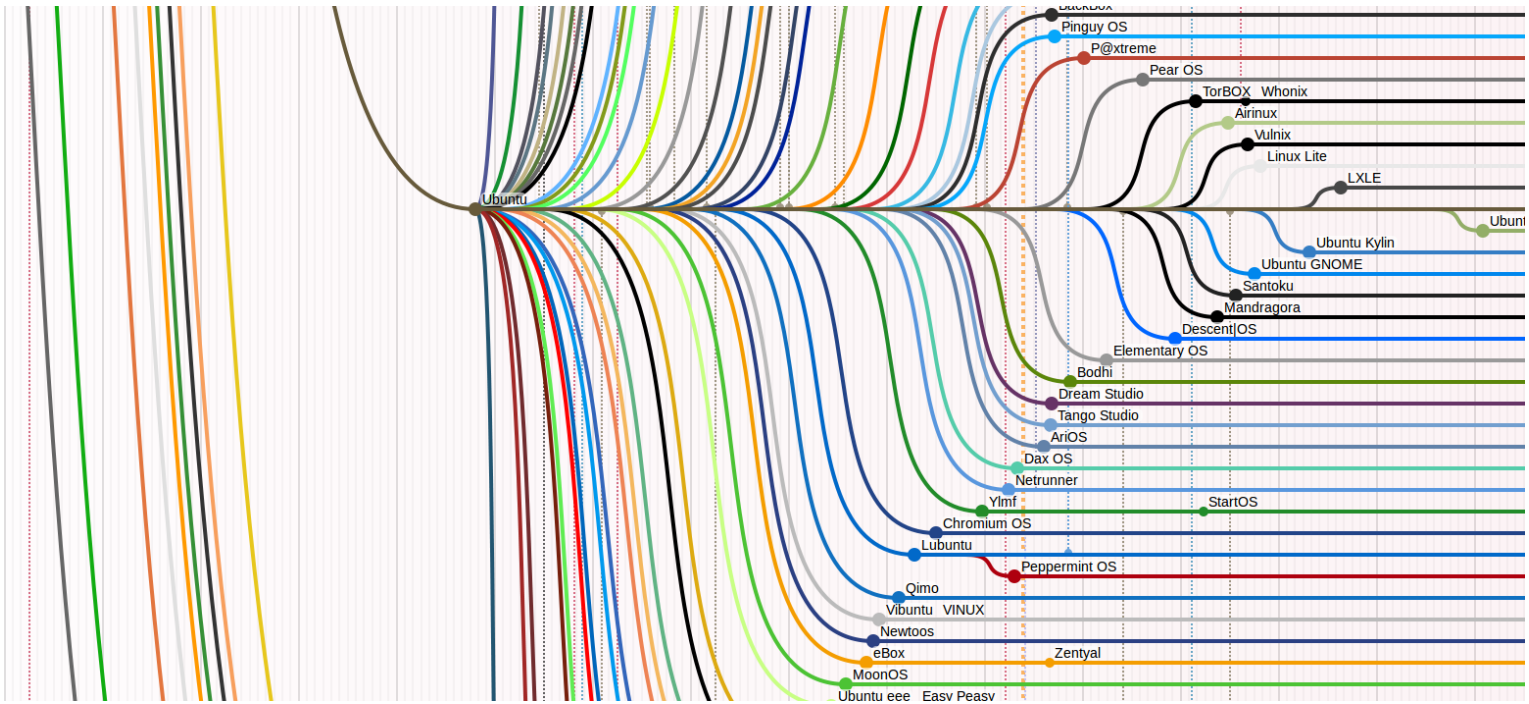
- Timeline delle distribuzioni di Linux, da Wikipedia (particolare)



- Timeline delle distribuzioni di Linux, da Wikipedia (particolare)



- Timeline delle distribuzioni di Linux, da Wikipedia (particolare)



- Possiamo installare Linux su una *macchina virtuale*, (come VirtualBox)
  - per *macchina virtuale* si intende un applicativo che replica in tutto e per tutto un computer, a partire dal tasto di accensione
  - è possibile far sì che la rete sia condivisa con il sistema operativo *ospitante*
    - per *ospitante* si intende il sistema operativo sul quale è in esecuzione VirtualBox
    - per *ospitato*, si intende il sistema operativo che viene eseguito all'interno di VirtualBox

Bash

Informazioni generali e comandi

Terminale e shell

- Per interagire con il sistema operativo si possono dare comandi dalla riga di comando usando il **terminale**
- In questo modo viene utilizzata la **shell**, che funziona appunto su un terminale
- Cioè, quando si chiama il terminale viene subito avviata la **shell standard**, come 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

Bash shell

- Ogni shell possiede un proprio linguaggio di programmazione che consente di scrivere *script shell*
- Da adesso in poi, assumiamo di avere un terminale aperto e che la shell sia la **bash** (Bourne Again Shell)
- Per sapere quale shell è in uso si possono usare due comandi:
  - `echo $0` oppure
  - `ps -p "$$" -ocmd -h`
- Un comando interessante della `bash` è `history`
- Si possono replicare comandi già dati usando i tasti cursore *freccia giù* e *freccia su*
  - una volta trovato il comando cercato, lo si può modificare: utile se si vuole rilanciare un programma già dato in precedenza con piccole modifiche
- Si possono anche ricercare parti di comandi dati in precedenza con CTRL+r (e una volta trovato il comando, lo si può modificare)

Bash shell

- La bash scrive un *prompt* ed attende che l'utente scriva un `comando`
  - *prompt* sta per *pronto*, e infatti la presenza del prompt indica, solitamente, che la shell è pronta ad accettare un nuovo comando
  - il prompt tipico è così costituito: `nomeutente@nomemacchina:~cammino$`, dove `cammino` è il path dalla directory home alla directory attuale
    - qui stiamo già parlando del filesystem, lo vedremo più avanti
  - Se si è semplicemente nella home (directory principale dell'utente), c'è solo la tilde $\sim$
  - se la directory corrente non si trova nel sottoalbero radicato nella home, allora `cammino` è il path assoluto (la tilde non c'è)

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`

Per cominciare

I comandi

Comandi

- Ogni comando verrà indicato come segue
  - `comando [opzioni] argomentiobbligatori`
  - tutto ciò che è tra parentesi quadre può essere omesso
  - se ci sono parentesi graffe sugli argomenti, allora ci dev'essere *almeno* un argomento (ma ce ne può essere anche più d'uno)
    - esempio: `cp [-r] [-i] [-a] [-u] {filesorgenti} filedestinazione`

Comandi

- Se ci sono le parentesi quadre e i puntini, allora ci possono essere 0, 1 o più argomenti (eventualmente separati dal carattere indicato)
  - esempio: `ps [opzioni] [pid...]`
  - altro esempio: `chmod mode[, mode...] filename`
  - talvolta, se necessario, potrà essere reso esplicito il numero di argomenti: `ps [opzioni] [pid1 ... pid`*n*`]`

Comandi

- Le opzioni sono tipicamente composte da uno o due *dash* (ovvero, il carattere -) seguiti da alcuni caratteri (senza spazi)
  - solitamente, dopo un dash c'è un solo carattere (versione *vecchia*), dopo 2 dash c'è una parola (versione *moderna*)
  - spesso, si possono usare 2 opzioni per dire la stessa cosa: per esempio le opzioni `-i` e `--interactive` del comando `cp` sono equivalenti
  - le opzioni sono sempre omissibili
  - le opzioni possono avere o no un argomento
    - esempi **senza** argomento: `-r`, `--recursive`:
    - esempi **con** argomento: `-k1`, `-k 1`, `--key=1`
    - le opzioni senza argomento con un trattino solo sono raggruppabili: `-b -r -c` è equivalente a `-brc`
  - gli argomenti sono solitamente (ma non necessariamente) nomi di file e/o directory

Comandi

- Primo esempio (*sinossi*) di comando:
  `man [sezione] comando`
  - dà informazioni complete su un comando
  - per esempio, si può (in un certo senso, ricorsivamente) digitare il comando `man man`
    - come risultato, si apre una pagina che illustra tutte le possibili opzioni che sono accettate dal comando `man`
  - considerando gli altri comandi visti sopra, si può anche eseguire: `man cp`, `man ps`, `man chmod`
  - si vede subito dalla *synopsis* che l'esempio dato sopra è molto semplificato, anche se l'uso tipico è quello

Comandi

- `man [sezione] comando`
  - si può notare che in alto a sinistra c'è scritto `MAN(1)`: vuol dire che la sezione è la `1`, quindi, lo stesso risultato si sarebbe ottenuto scrivendo `man 1 man`
  - si può navigare una pagina di manuale con le frecce cursore e con `PagUp`, `PagDown` (per sistemi in cui manca il programma `less`: si può solo premere la barra spaziatrice...)
  - si può ricercare una parola scrivendo prima lo *slash* (ovvero, il carattere `/`) e poi la parola da cercare (basta poi scrivere solo lo slash per cercarla ancora)
    - non tutto può essere cercato: provare a cercare il singolo carattere `[`
  - per uscire da una pagina di manuale, premere il tasto `q`
  - **Esercizio**: provare ad usare alcune delle opzioni di `man` riportate nella sinossi completa

Comando `man`

- `man [sezione] 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 comandi inseriti nella riga di comando (di solito 500)
- 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

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 aprire e scrivere in un file, si può usare un editor di testi
- Provare i comandi `geany nomefile &` o `gedit nomefile &` (se i corrispondenti editor sono installati)
- **Osservazione** il carattere `&` serve per l'esecuzione in background e permette di mantenere la finestra del terminale attiva (altrimenti il processo è bloccato)
- Si possono anche usare editor che si interfacciano direttamente con il terminale (se sono installati):
  - `nano nomefile` oppure `pico nomefile` (**senza** usare il carattere &)
  - oppure `vi nomefile` che è un editor un po' più complicato:
    - per uscire da `vi`, digitate i caratteri `:q` seguiti da invio

Comando `xdg-open`

- Il comando `xdg-open nomefile|url` apre un file o una url con l'applicazione selezionata come *preferita*
  - se l'argomento è un file sarà scelta l'applicazione preferita per il tipo di file
  - se l'argomento è una *url* sarà scelto il browser preferito
- Le opzioni sono:
  - `--help` mostra la sinossi del comando
  - `--manual` mostra la pagina del manuale relativa al comando
  - `--version` mostra le informazioni sulla versione di `xdg-utils`
- `xdg-open` fa parte del pacchetto `xdg-utils` che è un insieme di *tools* (o *utilities*) che permettono di integrare varie applicazioni con la distribuzione Linux utilizzata

Comando cat

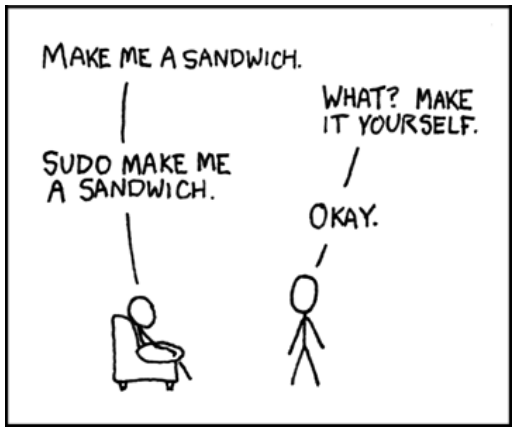
- Il comando `cat [nomefile]`: scrive a schermo il contenuto di `nomefile`
  - funziona bene solo se il file è di testo (e se usa la codifica riconosciuta da `cat`), altrimenti scrive caratteri incomprensibili
  - si può usare `cat` per leggere più di un file alla volta scrivendo `cat file1 file2... filen`:
    - in questo caso il comando `cat` stamperà prima il contenuto del primo file, poi quello del secondo, e così via
    - l'output sarà quindi la concatenazione del contenuto dei file specificati
  - 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*)

Creare altri file (non di testo)

- 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 (e va installato usando diritti da amministratore)
  - 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

Amministratore, utente e installazione di pacchetti

- 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`
  - 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



Comando `tree`

- Si può visualizzare un intero albero di directory con il comando `tree [-a] [-L maxdepth] [-d] [-x] [nomedir]`
  - potrebbe non essere installato e per installarlo si può scrivere: `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`)