

Sistemi Operativi, Secondo Modulo, Canale A–L
e Teledidattica
Riassunto della lezione del 22/02/2021

Igor Melatti

Qualche informazione generale sul corso

- Docente: Igor Melatti
- Dove potete trovare materiale ed informazioni generali sul corso:
<http://twiki.di.uniroma1.it/twiki/view/SO/S01213AL/SistemiOperativi12CFUModulo2CanaleAL20202021>
 - saranno presenti tutte le note di questo canale
 - le informazioni comuni con il primo modulo possono essere trovate alla pagina
<http://twiki.di.uniroma1.it/twiki/view/SO/S01213AL/SistemiOperativi12CFU>
- Libri di testo:
 - Glass, Ables, “Linux for Programmers and Users”, Prentice Hall
 - Johnson, “Pro Bash Programming”, Apress
 - B. Kernighan e D. Ritchie, “Il Linguaggio C”, Jackson Libri
- Regole per gli esami:
 - il secondo modulo di Sistemi Operativi **non va verbalizzato direttamente**; invece, confluisce in un’unica verbalizzazione con il primo modulo
 - questa verbalizzazione andrà sotto il nome di “Sistemi Operativi” (da 12 CFU)
 - * il voto finale è la media aritmetica tra i voti ottenuti ad entrambi i moduli
 - * per verbalizzare, occorre avere la sufficienza ad entrambi i moduli

- * inoltre, entrambe le sufficienze dovranno essere ottenute all'interno di un periodo di tempo così definito: sia x l'anno (solare, non accademico) nel quale si riesce a superare l'esame di uno dei due moduli; allora, occorrerà riuscire a superare l'altro modulo entro settembre (o novembre, se si hanno i requisiti per partecipare alla sessione straordinaria) dell'anno solare $x + 1$
- * ad esempio, se si è fatto il modulo 1 nel periodo da febbraio 2021 a settembre 2021, si ha tempo fino a settembre/novembre 2022 per fare anche il modulo 2
- * come ulteriore esempio, se si è fatto il modulo 2 a febbraio 2022, si ha tempo fino a settembre/novembre 2023 per fare anche il modulo 1
- qui di seguito, verrà trattato solo il secondo modulo; per informazioni sul primo modulo, vedere i link dati sopra
- ci sono 5 appelli d'esame nel corso dell'anno, più due “straordinari”
 - * uno a giugno (subito dopo la fine del corso), uno fine giugno / inizio luglio e uno a settembre nel 2021, e due a gennaio/febbraio 2022
 - * i due appelli straordinari (verso aprile il primo e verso novembre il secondo) sono riservati a studenti fuoricorso, ripetenti, part-time e lavoratori
 - * per partecipare agli appelli straordinari, occorre seguire le istruzioni pubblicate qui: <http://www.studiareinformatica.uniroma1.it/appelli-d-esame>
 - * verranno ammessi agli appelli straordinari tutti e soli gli studenti che la segreteria didattica comunicherà al docente
 - * per partecipare agli appelli occorrerà iscriversi su Infostud; onde districarsi tra i diversi verbali aperti, i quali avranno tutti la dicitura “Sistemi Operativi (12 CFU)” indipendentemente da canale e modulo, sul sito twiki comune ai 2 canali del secondo modulo verrà indicato, per ogni appello, il numero di verbale Infostud cui iscriversi
- ciascun appello d'esame è composto da uno scritto e da un orale
- è possibile partecipare a qualsiasi numero di esami (scritti e/o orali) nel corso dell'anno accademico
- per superare l'esame scritto occorre aver preso almeno 18
 - * fa fede il voto dell'*ultimo* esame scritto sostenuto
 - * nota bene: questo significa che non superare un esame scritto invalida eventuali esami scritti superati in precedenza
- il voto dell'esame scritto sarà al massimo 22
 - * sono possibili rare eccezioni a discrezione del docente

- chiunque abbia superato lo scritto può accettare direttamente il voto dello scritto stesso, senza un esame orale
 - * sono possibili eccezioni a discrezione del docente (in caso di dubbio di copiatura)
- chiunque abbia superato lo scritto può richiedere un esame orale per alzare il voto
 - * chi parte da 18 può arrivare a 30 e lode, e chi parte da 25 può essere bocciato
- chi venga bocciato all’orale (o rifiuti il voto) dovrà nuovamente superare un esame scritto
- l’orale può essere anche sostenuto (a discrezione dello studente) in un appello diverso da quello in cui ha superato lo scritto
- la modalità di svolgimento dell’esame dipenderà, ovviamente dalla situazione pandemica; sono pertanto possibili i seguenti casi:
 1. in presenza: tutti gli studenti fanno l’esame in un laboratorio della Sapienza attrezzato con computer
 2. a distanza: tutti gli studenti fanno l’esame a distanza, ciascuno a casa sua e con mezzi propri (possibile solo in caso di lockdown che coinvolga la Sapienza)
 3. in presenza con eccezioni: l’esame è in presenza, ma gli studenti che lo desiderano possono chiedere di farlo a distanza
 4. a distanza con eccezioni: l’esame è a distanza, ma gli studenti che lo desiderano possono chiedere di farlo in presenza
- per ogni appello, verrà chiaramente indicata la modalità scelta
- finché perdurerà l’emergenza CoViD-19, e potendo scegliere, il docente di questo corso effettuerà tutti gli esami in modalità 4
 - * in caso di lockdown che coinvolga la Sapienza, si passerà alla modalità 2
- le seguenti sono le dotazioni hardware minime richieste:
 - * PC (portatile o fisso) con webcam, microfono e altoparlanti (o cuffie);
 - * smartphone (o anche tablet);
 - * connessione Internet stabile attiva su entrambi i dispositivi;
 - * tutti i dispositivi devono essere collegati all’alimentazione elettrica durante l’esame;
 - * un documento di riconoscimento in corso di validità;
- i seguenti sono invece i prerequisiti software minimi:
 - * lo smartphone deve essere configurato, per tutta la durata dell’esame, in modo che non vada in sleep;

- si può fare o con app come wakey, o configurando opportunamente le impostazioni;
 - * un browser sul PC; accertarsi di essere in grado di eseguire GoogleMeet a partire dal proprio account di posta istituzionale (@studenti.uniroma1.it);
 - * l'applicazione Zoom sul PC (nota bene: per gli scopi dell'esame, non è possibile utilizzarla da browser);
 - * l'applicazione Zoom sullo smartphone;
 - * un terminale Linux sul PC; se non disponibile, è possibile eseguire una macchina virtuale dentro la quale eseguire il terminale
 - se non si ha neanche la macchina virtuale, si possono usare applicazioni del sistema operativo in uso: ad esempio Cyg-Win per Windows, mentre per macOS è presente un vero e proprio terminale nel software di sistema
 - * un'applicazione di posta elettronica (anche su browser) configurata sull'account istituzionale, sia sul PC che sullo smartphone.
- chi non soddisfa tali requisiti minimi, non può partecipare all'esame a distanza da casa
- più in particolare, per l'esame scritto, valgono le seguenti regole:
1. si tratta di un compito a quiz da fare direttamente al computer; è permesso l'uso del terminale di Linux, ma non è permesso portare appunti
 2. nel caso in cui ci siano molti iscritti, l'esame si farà in più turni, comunicati sul sito del corso
 - * se è in presenza, i turni sono determinati dal numero di posti in laboratorio
 - * se è da remoto, i turni sono determinati da quanti docenti fanno parte della commissione
 3. ci saranno 40 domande da fare in 25 minuti
 4. le domande saranno a risposta chiusa: tra quelle proposte, una e una sola opzione sarà esatta
 5. le domande verteranno sempre sull'intero programma del corso
 6. per ciascuno studente viene calcolato il punteggio come $2E - S$, con E numero di risposte esatte ed S numero di risposte sbagliate
 - * quindi il punteggio va da -40 a 80
 7. per assegnare un voto da insufficiente a 25 a ciascun punteggio, si valuteranno i punteggi di tutti i partecipanti all'esame, seguendo una distribuzione che assomiglia ad una campana di Gauss (*cum grano salis*)
 - * conseguenza: se ci sono copiatore, molti compiti avranno punteggi vicini, e quindi il voto si abbassa. Morale: non conviene copiare (altrimenti detto: chi permette copiatore regala parte del suo voto ad un altro)

- * non è necessario rispondere a tutte le domande
 - * chi fa il compito migliore rispetto agli altri ha il punteggio più alto
- più in particolare, per l'esame orale ci sono 2 modalità: con homework sincroni e con homework asincroni
1. con homework sincroni
 - * verranno proposti 2 homework: uno a circa metà corso (sul Bash scripting) e uno a fine corso (sul System Programming)
 - * il testo e la scadenza degli homework verranno pubblicati sulla pagina del corso
 - indicativamente, il primo homework andrà consegnato intorno all'inizio di aprile, il secondo intorno alla fine di maggio
 - passata la scadenza, non sarà più possibile consegnare gli homework
 - * ogni homework è *individuale*
 - è possibile consultarsi con altri studenti, ma poi ognuno deve scrivere la propria soluzione personale
 - * a tal proposito, verranno effettuati controlli anti-copia, sia automatici che manuali (a campione)
 - * in caso di copiatura di un homework, verranno convocati tutti gli studenti coinvolti, e in mancanza di spiegazioni esaurienti verranno cancellati i voti di entrambi gli homework a tutti gli studenti coinvolti
 - * chi sceglie di fare gli homework dovrà affrontare un orale di convalida, che avrà come unico argomento gli homework svolti
 - * per accedere all'orale di convalida, occorre aver superato lo scritto
 - * l'orale di convalida deve essere sostenuto entro febbraio 2022 (od aprile 2022 se si ha diritto all'appello straordinario), e può essere sostenuto *una volta sola*
 - * il voto di ciascun homework sarà *sommato* al voto dello scritto
 - * affinché il voto degli homework sia valido, è necessario superare ciascun homework con un punteggio strettamente maggiore di 2
 - * in caso di non superamento dell'orale di convalida, vengono cancellati i voti di entrambi gli homework, nonché il voto dello scritto
 - * in caso l'orale di convalida non venga sostenuto, vengono cancellati i voti di entrambi gli homework, ma il voto dello scritto (se conseguito con sufficienza) viene mantenuto

- * l'orale di convalida ha le stesse modalità (in presenza obbligatoria o facoltativa, da remoto obbligatorio o facoltativo) dell'esame scritto, ma si svolge singolarmente anziché in gruppo
 - * la correzione degli homework sarà automatica, e il codice del correttore verrà reso disponibile agli studenti
 - l'architettura di riferimento sarà la stessa distribuzione Debian presente sui computer del laboratorio
2. con homework asincroni
- * occorre mandare una settimana prima dell'appello cui si vuole partecipare una propria versione originale degli homework
 - * la data precisa verrà indicata sul sito del corso, unitamente al link da usare per la consegna
 - * verrà chiesto di effettuare una modifica ad entrambi gli homework
 - * anche in questo caso, le modalità (in presenza obbligatoria o facoltativa, da remoto obbligatorio o facoltativo) sono le stesse dell'orale di convalida
- di norma, l'esame orale inizia subito dopo l'esame scritto, con la comunicazione da parte del docente di un calendario di massima
 - in tale calendario di massima, a discrezione del docente, gli orali possono essere divisi su più giorni
- Obiettivo di questo corso: imparare alcuni rudimenti di programmazione ed uso del Sistema Operativo Linux
 - Organizzazione del corso
 - 27 lezioni in tutto (tenendo conto delle festività), se non ne salta nessuna
 - * dal 22 febbraio 2021 al 26 maggio 2021
 - ogni settimana 2 lezioni da 3 ore ciascuna
 - le lezioni possono essere seguite a distanza oppure in presenza
 - * ovviamente, con l'evolversi dell'attuale pandemia, potrebbe succedere che diventi obbligatorio seguire solo a distanza
 - In presenza: solo se siete stati esplicitamente autorizzati dall'applicativo PRODIGIT <https://prodigit.uniroma1.it/>
 - * ci sono controlli all'entrata della Sapienza
 - * gli studenti dovranno confermare la propria presenza in aula tramite il token
 - A distanza: tramite l'applicativo zoom

- * si consiglia l'uso dell'applicazione (scaricabile da <https://zoom.us/download>)
- * è comunque possibile usare la versione web (<https://zoom.us/>), che però ha limitazioni
- C'è un'unica conferenza zoom che verrà usata per tutto il corso, con ID **873 1883 6987**
- La password verrà comunicata tramite email (istituzionale) a tutti gli iscritti al link http://151.100.17.205/iscrizione/corso.php?id_corso=14
 - * l'iscrizione va fatta una volta sola ed è valida fino all'ultima lezione del corso
- C'è una waiting room: dovrete aspettare che il docente vi ammetta
 - * l'attesa potrà protrarsi al massimo fino a 10 minuti dopo l'orario stabilito
 - * waiting room comunque controllata ogni 20-30 minuti per ammettere eventuali ritardatari
 - * selezionare “join with computer audio”
- **Importante 1:** settare il proprio numero di matricola (e nient'altro) come nickname zoom
 - * vi viene chiesto sia dall'applicazione che dal sito web
 - * rischiate di non venire ammessi se non vi presentate in questo modo
- **Importante 2:** spegnere immediatamente sia il vostro video che il vostro microfono
 - * per ragioni di privacy, siete sconsigliati dal settare una vostra foto di profilo
- Per fare domande durante la lezione:
 - * se siete in presenza, chiedete e basta; parlate a voce non troppo alta e mantenendo la mascherina
 - * se siete a distanza, dovete usare la chat di zoom
 - o per scrivere direttamente la vostra domanda
 - o per chiedere il permesso di parola a voce: una volta ottenuto, attivare il microfono, porre la domanda, e disattivarlo nuovamente
 - * il docente risponderà sempre a voce, ripetendo la domanda
- Tutte le lezioni sono registrate
 - * copia della registrazione verrà mandata a tutti coloro che si sono prenotati su http://151.100.17.205/iscrizione/corso.php?id_corso=14 (già menzionato sopra)

Commenti per la prima lezione

- Esplorazione del sistema Linux installato in laboratorio: boot, scelta macchina virtuale, login, workspaces, applicazioni principali, gestore dei file, terminale, shortcuts predefinite, shortcuts customizzate, copia/incolla
- Lanciare un'applicazione sia da interfaccia grafica (non sempre possibile) che da terminale
- Logout, Reboot, Halt

Storia di Unix/Linux

Dalle origini di Unix al primo Linux

- In breve: Multics (1965, AT&T) → Unix (1970, Bell Labs della AT&T) → GNU/Linux (1983, Richard Stallman → 1991, Linus Torvalds)
- Prima parte: da Multics a Unix (1965 → 1974)
 - Multics: MULTiplexed Information and Computing System, nelle intenzioni multi-processo e multi-utente (time-sharing) con file system gerarchico
 - troppo mastodontico per i tempi, abbandonato dopo 5 anni, ma...
 - ... dalle ceneri risorge Unics (da “molto” a “uno”), poi rinominato in Unix
 - Unix sorprende tutti, e nel giro 3-4 anni viene “portato” su macchine diverse (inizialmente, i DEC-PDP della Digital)
 - * la primissima versione di Unix fu sviluppata da un ricercatore dei Bell Labs, Ken Thompson, che aveva a disposizione un piccolo computer, il PDP-7 (Figura 1), che aveva 9 kB di RAM (espandibili fino a 144..) e 512 kB di disco; valore commerciale: 70000 \$
 - il porting è reso possibile dal fatto che Unix viene riscritto in linguaggio ad alto livello (a parte poche parti di codice necessariamente in assembler): prima il B e poi il C
 - * il B si chiama così perché è un'evoluzione (semplificata) di un altro linguaggio, il BCPL
 - il BCPL non ebbe mai successo da solo, ma grazie alla discendenza BCPL → B → C, ha influenzato i linguaggi oggi più usati, come C, C++, Java
 - per dirne una: i blocchi con le parentesi graffe vengono da BCPL



Figure 1: PDP-7, da Wikipedia

- * il C si chiama così perché è l'evoluzione (miglioratoria) del B: si aggiungono strutture e tipi di dato (altrimenti, in B e BCPL c'era solo la word)
 - tutto il resto era già nel B: parentesi graffe (riprese da BCPL), sintassi per la dichiarazione di variabili e funzioni, sintassi dei costrutti `if`, `while`, `for` etc, assegnamenti e test di uguaglianza...
- * questi linguaggi vengono sviluppati da Thompson (B) e Ritchie (C)
- * il C viene completato all'inizio degli anni '70 e consente il porting di Unix su altre piattaforme hardware
- * il primissimo Unix del PDP-7 era in assembler
- Il bello è che AT&T non può lucrare su Unix: viene distribuito a tutti (soprattutto alle Università ed ai centri di ricerca, ma poi anche le aziende) con il codice sorgente
 - nascono moltissime versioni di Unix: ognuno modifica il codice sorgente come gli pare
 - le più importanti sono: System V (Unix della AT&T) e BSD (Berkeley Standard Distribution, della University of Berkeley)
 - nella versione del 1983 della BSD viene introdotto TCP/IP
 - verranno distribuite con ulteriori modifiche da svariate aziende, tra cui un'insospettabile Microsoft (ma anche IBM, HP e Sun Microsystems)
- Fin qui, Unix è praticamente usabile soltanto da personale specializzato di centri di ricerca ed Università: niente personal computer, che pure negli anni '80 cominciano ad essere costruiti

- ma vale la pena di ricordare che MS-DOS è basato su Unix (tolta ovviamente la parte del multi-utente e multi-programma, quindi essenzialmente resta solo il file system gerarchico)
- tutti i sistemi operativi della Apple sono basati sullo BSD Unix
- Per portare Unix a tutti, arriva Stallman
 - una mezza specie di santone, si inventa la licenza GPL: software scritto gratuitamente da una collaborazione mondiale di programmatori (Internet c'è già, anche se il Web ancora no...), tutti lo possono riusare, ma senza scopo di lucro
 - prendono Unix e lo riscrivono completamente (in pratica, lasciano solo le funzionalità)
 - implementazione di singoli pacchetti, alcuni dei quali importantissimi: gcc (GNU C Compiler) e make, varie shell, ...
- Ma manca il pezzo più importante: il kernel, in grado di dialogare con l'hardware, gestendo processore, RAM e I/O
 - ma è oramai tempo dei PC “standard” (o “IBM-compatibili”): Linus Torvalds programma il kernel del primo Linux (1991) per un 80386
 - per tutto il resto, si usano i pacchetti di GNU, quindi si chiama GNU/Linux

Dagli anni 90 ad oggi

- Nel 1994 la BSD introduce una differenziazione che talvolta si trova anche nei Linux odierni: c'è una versione libera e riusabile ed una con codice proprietario AT&T
 - quindi, occorre una licenza dell'AT&T per usare quella non libera (*encumbered*)
 - quella stessa licenza che negli anni '70 veniva concessa ad università e centri di ricerca, e poi ad aziende
 - l'altra la può usare chiunque (versione *lite*)
 - da quest'ultima discende MAC OS X
- Sempre nel 1994 viene definito lo standard per Unix: cosa deve avere un sistema per definirsi Unix
 - lo definisce il SUS (*Single UNIX Specification*)
 - chi si vuole fregiare del marchio UNIX®[®], oltre che a rientrare nelle specifiche, deve anche pagare le royalties (da 25000\$ all'anno per meno di 1000 istanze, a 110000\$ all'anno per più di 30000 istanze)
- Oggi ci sono 3 categorie di sistemi Unix (*famiglia Unix*):

Genetic Unix: Unix che provengono da quello dell'AT&T o da quelli da lui derivati (BSD). Rientra in questa categoria anche il BSD-lite, anche se non ha più nessun codice AT&T

Trademark Unix: Unix che pagano le royalties al SUS per essere definiti Unix

Functional Unix sistemi operativi che si “ispirano” a Unix. Qui rientra Linux (e anche altro, come Minix). Anche detti impropriamente *Unix-like*, o più propriamente *Unix system-based*

Caratteristiche di un Sistema Unix

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

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 esempio, 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, editor di testo
- 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

Utilities: altri programmi, per fare qualsiasi cosa che sia computabile

- suite Office (OpenOffice, LibreOffice)

- lettore PDF (Acrobat Reader, evince, ...)
- browser (Firefox, Google Chrome, ...)
- messaggistica (Skype, ...)
- riproduttore audio/video (VLC, mplayer, ...)
- editor di immagini (xfig, gimp, ...)
- giochi (semplici!)
- ...

Modularità: programmi installabili a pacchetti, e moduli del sistema operativo attivabili e disattivabili

Unix e la grafica

- Agli inizi, schermo nero con richiesta di login
- Dal 1984, XWindow System (il più usato è il cosiddetto X11, del MIT)
 - disegna lo schermo e gestisce gli eventi (battitura tasti, spostamento e/o click del mouse)
 - non è sempre presente (solitamente inutile per server dedicati)
- Desktop environment: dà istruzioni all'XWindow per disegnare effettivamente la schermata
 - ne esistono parecchi, e ognuno definisce un suo stile (solitamente personalizzabile)
 - Gnome, KDE, CDE, Xfce, Unity, ...
- Window Manager: fa parte del Desktop Environment, e definisce la modalità di interazione con le finestre aperte (workspaces etc)

Filosofia di Unix

1. Composto da una serie di piccoli programmi che eseguono un compito specifico, limitato, ma in maniera esatta e semplice.
2. I programmi sono silenziosi, il loro output è minimale e ridotto a: ciò che è stato esplicitamente richiesto e ciò che costituisce particolare interesse.
3. La definizione di lavori complessi può essere svolta come articolazione del lavoro svolto da più programmi semplici.
4. I programmi manipolano testo e non file binari.
5. Qualsiasi risorsa può essere rappresentata come file o come processo.

Le versioni di Linux

- Ce ne sono moltissime (Figura 2)
- Ultimamente si stanno imponendo le Debian-based, come la Ubuntu; per questo corso, useremo una versione modificata della Debian installata nei laboratori
- Da installare su una macchina virtuale, come VirtualBox (scaricabile da qui: <https://www.virtualbox.org/>)
 - 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”
 - * qui, 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
 - è possibile far sì che una particolare directory del sistema operativo ospitante venga vista come directory di rete dal sistema operativo ospitato
 - è possibile far finta di aver messo un CD (fornendo a VirtualBox un ISO), oppure mettere davvero un CD...
 - video se si vuole provare ad installare Linux: <https://www.youtube.com/watch?v=QpHIqI3Pyqg>
 - qui è disponibile direttamente il disco virtuale, non necessaria l’installazione; vedere le istruzioni su twiki

Cenni basilari sulla shell

- Da adesso in poi, si suppone di avere un terminale aperto, e che la shell sia la `bash` (Bourne Again Shell)
 - per lanciare la `bash` su Debian, cliccare in basso a sinistra, scegliere “Strumenti di Sistema” e poi “LXTerminal”
 - per sapere quale shell è in uso si possono usare due comandi: `echo $0` oppure `ps -p "$$" -ocmd -h`
 - una cosa interessante della `bash`: la *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

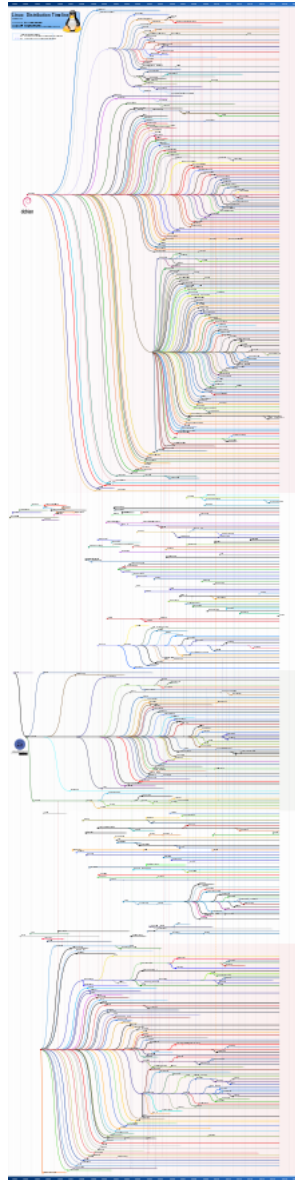


Figure 2: Timeline delle distribuzioni di Linux, da Wikipedia

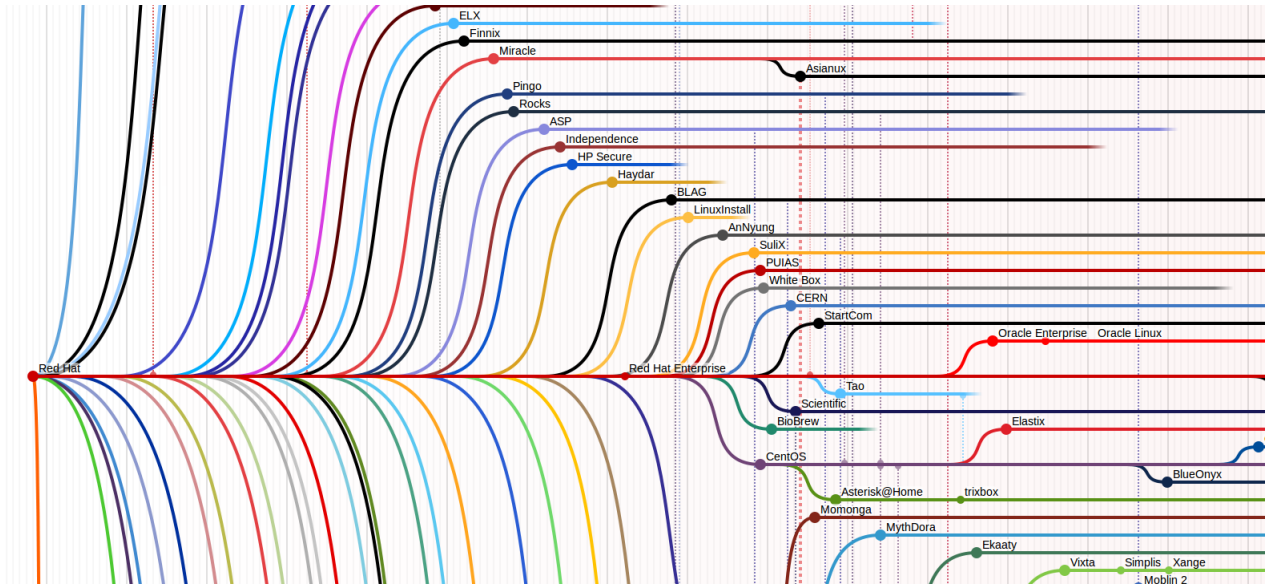


Figure 3: Timeline delle distribuzioni di Linux, da Wikipedia (particolare)

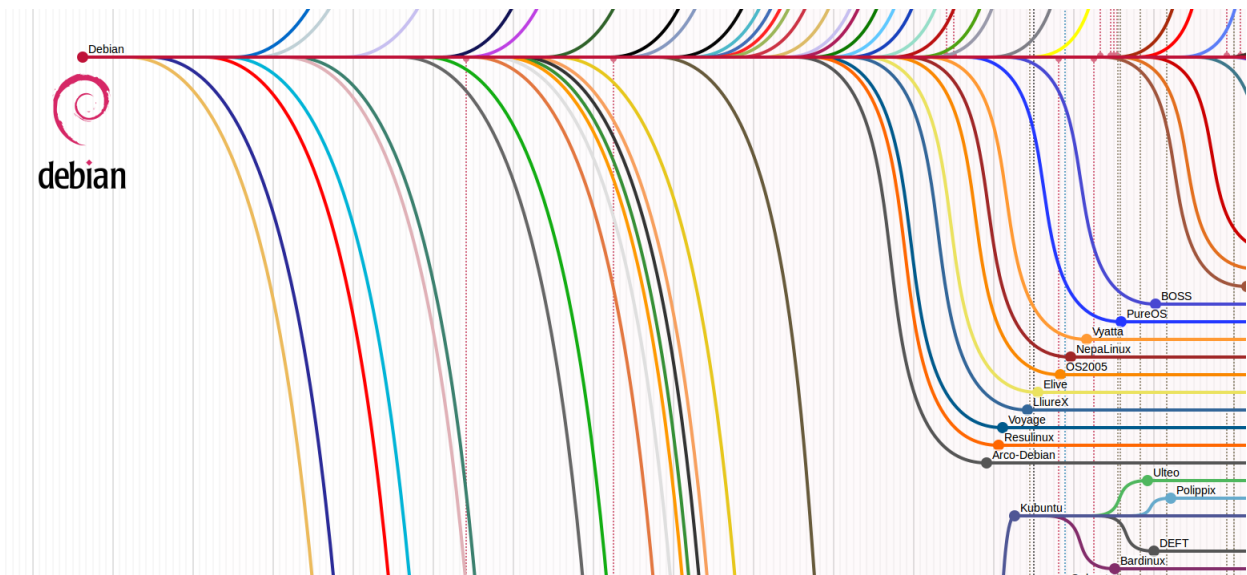


Figure 4: Timeline delle distribuzioni di Linux, da Wikipedia (particolare)

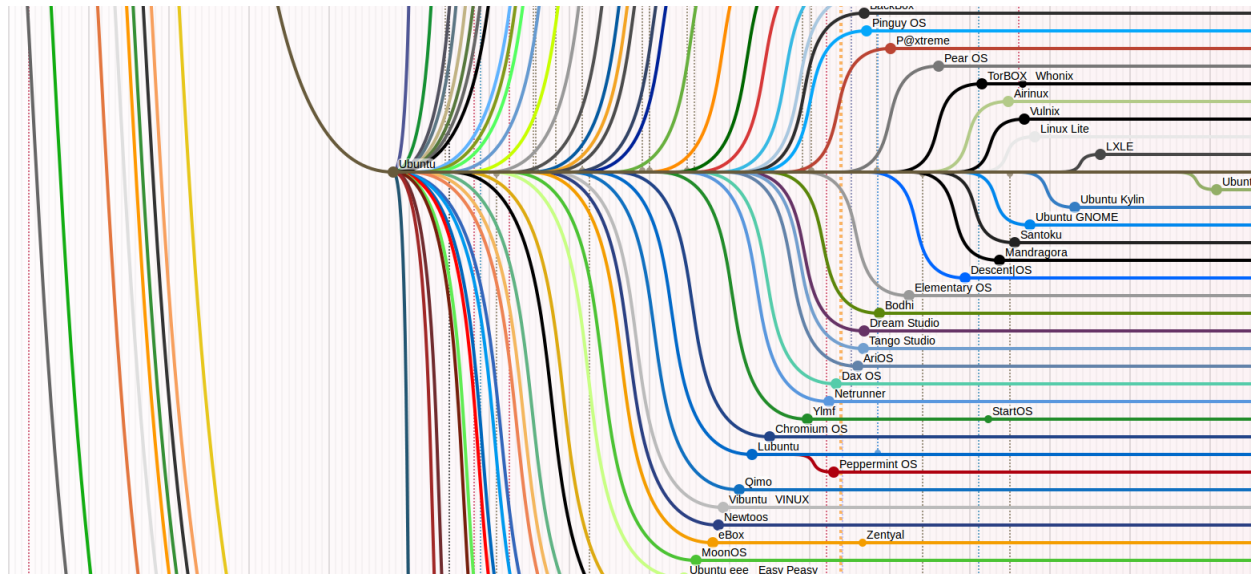


Figure 5: Timeline delle distribuzioni di Linux, da Wikipedia (particolare)

- si possono addirittura ricercare parti di comandi dati in precedenza con CTRL+r (e una volta trovato il comando, lo si può modificare)
- La bash scrive un **prompt** ed attende che l'utente scriva un *comando*
 - “prompt” sta per “pronto”, ed 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; per maggiori informazioni vedere più sotto
 - quindi, se si è semplicemente nella home, c'è solo la tilde ~
 - se la directory corrente non si trova nel sottoalbero radicato nella home, allora **cammino** è il path assoluto (e la tilde non c'è)
- Ogni comando verrà nel seguito 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`

- 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 ... pidn]`
 - le opzioni sono tipicamente composte da uno o due *dash* (ovvero, il carattere `-`) seguiti da alcuni caratteri (senza spazi)
 - * sempre solitamente, dopo un dash (versione “vecchia”) c’è un solo carattere, dopo 2 dash (versione “moderna”) c’è una parola
 - * spesso, ci sono 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`
 - * per completezza di trattazione: esistono anche le opzioni BSD-style, che sono senza dash; ad es.: `tar xfz nomefile.tgz`
 - gli argomenti sono solitamente (ma non necessariamente) nomi di file e/o directory
 - sono possibili anche altre modalità, che verranno man mano spiegate in queste note
- 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`
 - * considerando gli altri comandi visti sopra, si può anche eseguire: `man cp, man ps, man chmod`
 - come risultato, si apre una pagina che illustra tutte le possibili opzioni che sono accettate dal comando `man`
 - si vede subito dalla *synopsis* che l’esempio dato sopra è davvero molto semplificato, ma l’uso tipico è quello
 - **esercizio:** provare ad usare alcune delle opzioni di `man` riportate nella sinossi completa
 - la sezione sarà importante in seguito; per ora ci si limiti a 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`

- scrivere solo `man`, invece, non funziona
- 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 `[`
 - * il perchè lo capiremo quando studieremo le espressioni regolari
- per uscire da una pagina di manuale, premere il tasto `q`
 - * il perchè lo capiremo quando vedremo i comandi `less` e `more`

Gli utenti

- Durante l'installazione di un qualsiasi Linux, è necessario specificare (almeno) un *utente*
 - nel caso dell'immagine già fatta di Debian, è stato già creato un utente chiamato `studente` con password `informatica`
 - non tutti gli utenti possono fare login (per esempio, in molti sistemi Linux non può farlo l'utente `root`)
 - in effetti, l'installazione di Linux crea svariati utenti (almeno una decina), che servono solo per scopi interni del sistema operativo
- Ogni utente appartiene ad almeno 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, ed uno stesso utente può appartenere a più gruppi (per sapere quali, si può usare il comando `groups [nomeutente]`)
 - di nuovo, molti gruppi servono solo a scopi interni al sistema operativo
- Nel caso di Ubuntu e derivati, l'utente specificato a tempo di installazione sarà un *sudoer*, e apparterrà al gruppo predefinito `sudo`
 - questo vuol dire che quell'utente può eseguire comandi da superutente (per esempio, quelli per installare nuovi pacchetti) semplicemente preponendo `sudo` (Figura 6)
 - dove `sudo comando` è un comando particolare, che prende come argomento un altro comando (che potrebbe avere svariati argomenti)

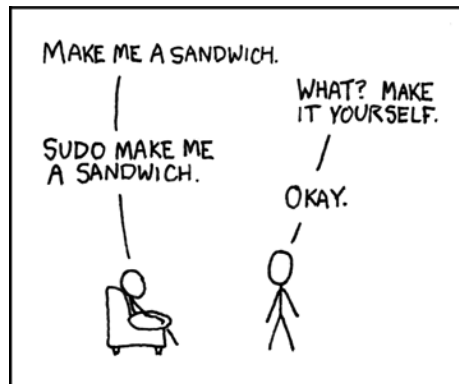


Figure 6: Il comando `sudo` secondo xkcd 149

- nel caso di Debian su disco già pronto, `studente` è sudoer (questo non è vero nel laboratorio!)
- È possibile creare un altro utente usando il comando `adduser nuovoutente`
 - di default, questo crea un utente *non* sudoer
 - ma il comando va dato da un utente sudoer (creare utenti rientra nelle prerogative di un amministratore di sistema), che deve anche aggiungere `sudo` all'inizio
 - **esercizio**: creare due nuovi utenti non-sudoer, con nome `utente1` ed `utente2`
- È possibile cambiare utente usando il comando su `-l nomeutente`
 - con l'utente sudoer, provare ad eseguire `apt-get update`
 - con l'utente sudoer, provare ad eseguire `sudo apt-get update`
 - con l'utente `utente1`, provare ad eseguire `apt-get update`
 - con l'utente `utente1`, provare ad eseguire `sudo apt-get update`
- 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
 - **esercizio**: rendere uno dei due nuovi utenti un sudoer, poi riprovare a fare gli esempi di sopra con `apt-get update`

Il filesystem

- 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* od 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
 - vedremo che esistono anche *file speciali* (non regolari)
- Linux ha un solo filesystem principale, che ha come radice la *directory / (root)*
 - tutti i *file* e le *directory* sono contenuti, direttamente od indirettamente, in tale *directory*
 - non esistono quindi i “volumi” di Windows
 - le foglie dell'albero possono essere o *directory* vuote o *file*
 - all'interno della stessa *directory* non ci possono essere due *file*, due *directory*, o un *file* e una *directory* con lo stesso nome
 - cambiare le maiuscole/minuscole è sufficiente a distinguere tra due *file* o *directory*: **nomeFile** è diverso da **nomefile**
- 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 `~`
 - infatti, come vedremo, la tilde è una scorciatoia per la *directory* home dell'utente corrente `x: /home/x`
- C'è inoltre il concetto di *current working directory* (*cwd*), che vale per ogni processo
 - vale 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)
 - * notare che `man cd` non funziona; vedremo in seguito il perché
- 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
- 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`
- A posteriori, 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
- **esercizio:** mostrare un esempio in cui un path relativo è valido anche cambiando la cwd
- Nel seguito, quando ad un comando dovrà essere dato come argomento un nome di un file o un nome di una directory, si intende che tale nome può essere un path relativo od assoluto