Introduzione al sistema Unix

Sistemi di Elaborazione – Programmazione di sistema in ambiente Unix

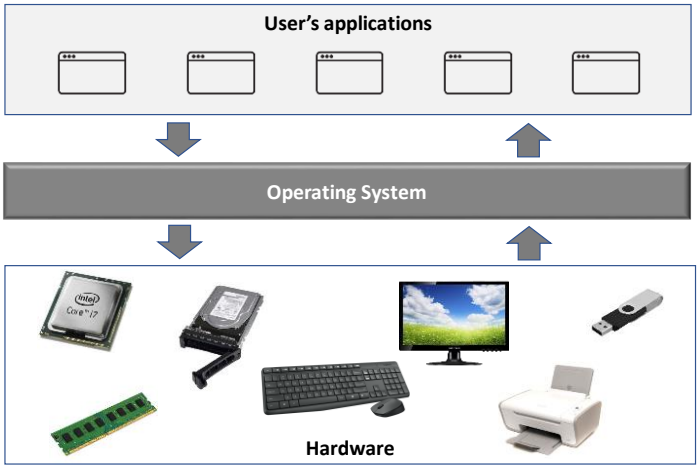
Anno Accademico 2023/2024

Ing. Giovanni Nardini

Università di Pisa

Cos’è Unix?

Cos'è un Sistema Operativo?



Ad alto livello, un calcolatore è formato da componenti hardware, usate da componenti software (le applicazioni) per svolgere il loro compito: CPU per eseguire le istruzioni del proprio codice, memoria per immagazzinare il codice e i dati, periferiche per fare input e output.

Il Sistema Operativo è l'intermediario tra i programmi dell'utente e le componenti hardware del sistema. Rende facile l'utilizzo delle risorse nascondendone i dettagli implementativi.

Quando programmiamo, abbiamo bisogno di accedere alle risorse hardware (es. per scrivere sul video). Per farlo, ci basta sfruttare le funzionalitá e delle astrazioni che il sistema operativo ci mette a disposizione.

Cos'è Unix?

Per come lo intendiamo oggi, **Unix non è un sistema operativo!**

- È una *famiglia* di sistemi operativi, che presentano una architettura e una filosofia comune
- Ci riferiamo a un sistema Unix quando il sistema operativo rispetta delle determinate caratteristiche

Alcune caratteristiche comuni ai sistemi Unix:
- sistemi multiutente e multiprogrammati
- insieme comune di comandi e interfacce per l'utente
- struttura del file system comune
- gestione dei dispositivi come se fossero dei file
- …

Un po' di storia di Unix

- **1969**
  - Sviluppato da Ken Thompson e Dennis Ritchie @ Bell Labs di AT&T
- **1973**
  - Riscritto usando linguaggio C
  - Prima: Assembler
- **1975**
  - Thompson, Joy, Haley e alcuni studenti di Berkeley (Univ. of California) sviluppano la '**B**erkeley **S**oftware **D**istribution' (BSD) di Unix
- **Poi**
  - Evolve in una **enorme** galassia di distribuzioni

K. Thompson (seduto) e D. Ritchie (in piedi) mentre lavorano sul sistema PDP-11

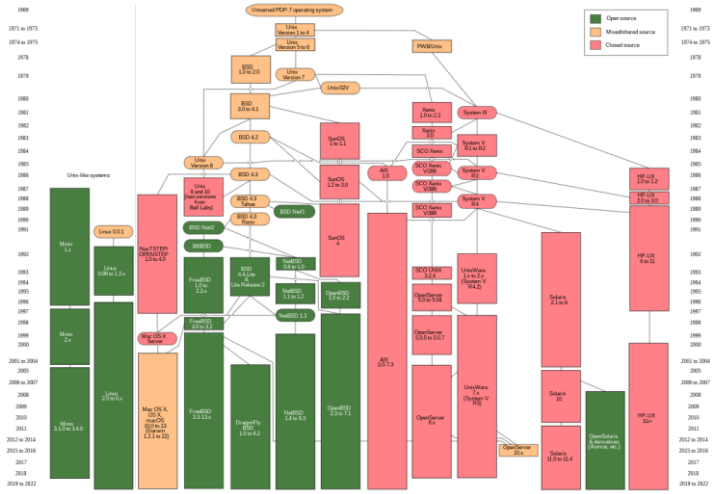
Sviluppato nel 1969 da Thompson e Ritchie, era un sistema multiutente e monoprogrammato, cioè eseguiva un programma alla volta.

Inizialmente, era scritto in linguaggio Assembler: il set delle istruzioni dipende dall'hardware che le esegue. Per cui, Unix non era portabile su macchine con hardware diverso.

Nel 1973, viene riscritto in linguaggio C, inventato appositamente da D. Ritchie.

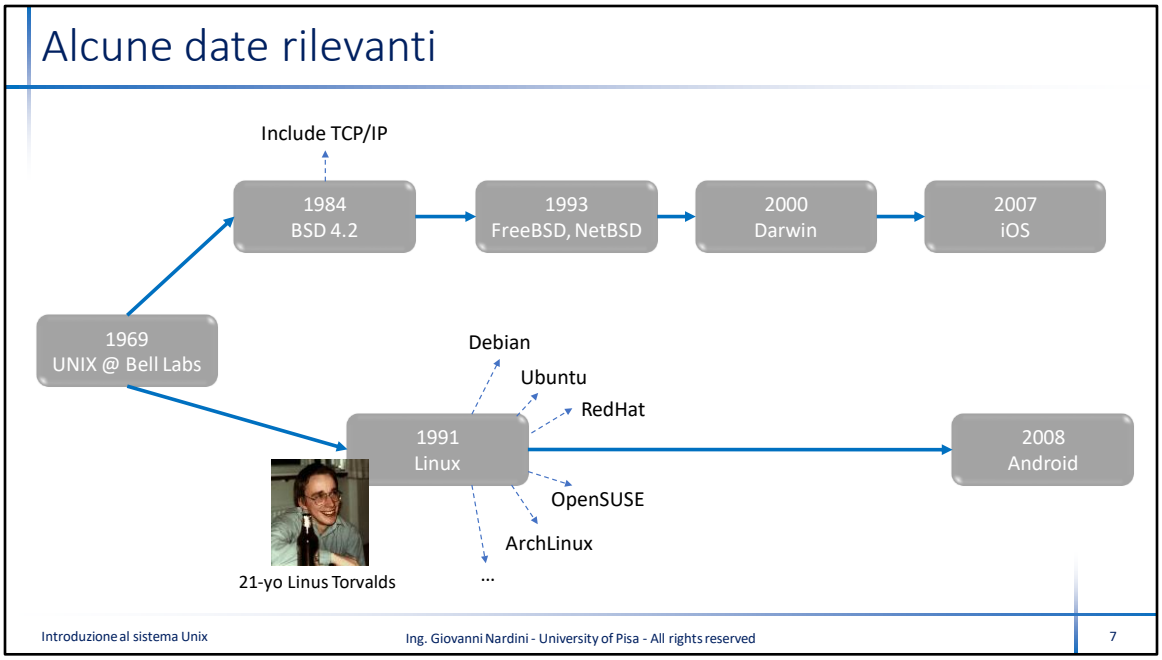
Dato che per motivi legali AT&T (anti-trust) non poteva commercializzare nuovi prodotti, Unix viene rilasciato con licenza open source (libero) alle università, e ciò ha favorito la sua diffusione e la nascita di molte distribuzioni derivate.

La galassia di Unix



fonte:
https://en.wikipedia.org/wiki/Operating_system

La frammentazione nell'evoluzione di Unix è anche (ma non solo) a battaglie legali. Per esempio, a un certo punto una spinoff di dell'Università di Berkeley chiamata BSDi cominciò a vendere una propria versione di Unix basata su BSD, e la commercializzò con il nome UNIX (tutto maiuscolo).
I Bell Labs la presero un po' male perché il codice sorgente conteneva delle parti originali della versione di AT&T, che però era stato licenziato solo alle università, e fece causa alla BSDi. Dopo un lungo tempo di cause legali tra i Bell Labs e Berkeley, il caso si chiuse quando Berkeley riscrisse le parti incriminate del codice del sistema operativo e distribuì una nuova versione (BSD4.4-lite).



BSD 4.2 è la prima distribuzione che includeva lo stack protocollare TCP/IP, che ancora oggi è il protocollo che rappresenta lo standard de-facto per la comunicazione in Internet. Da lì, vennero sviluppate due versioni molto popolari come FreeBSD e NetBSD.

Parallelamente, Linus Torvalds progettò **Linux**.
Linux è un sistema **Unix-like:** anche se si comporta allo stesso modo di Unix, un sistema Unix-like è stato riscritto da capo e non contiene alcun pezzo di codice della versione originale di Unix.
Lo stesso Linux vanta un numero piuttosto grande di distribuzioni: le varie distribuzioni condividono lo stesso nucleo e si differenziano per il software built-in con l'installazione del sistema operativo.

(Solo un sottoinsieme delle) Versioni di Unix

1BSD	2BSD	3BSD	4BSD	4.4BSD Lite 1
4.4BSD Lite 2	386 BSD	A/UX	Acorn RISC iX AIX/ESA	AIX
AIX PS/2	AIX/370	AIX/6000	ArchBSD	AIX/RT
AMiX	AOS Lite	AOS Reno	BSD Net/1	BSD Net/2
Atari Unix	BOS	BRL Unix	Chorus	Chorus/MiX
BSD/386	BSD/OS	Darwin	Debian GNU/Hurd	DEC OSF/1
Digital Unix	DragonFly BSD	Dynix	Dynix/ptx	HP-UX
FreeBSD	GNU	IBM IX/370	HPBSD	Mac OS X
HP-UX BLS	IBM AOS	Lites	MicroBSD	Mini
IRIX	Linux	MERT	MirBSD	Linux
Mac OS X	Mach	MIPS OS	Open UNIX	NetBSD
Minix	OPENSTEP	mt Xinu	OS/390 Unix	OpenBSD
NeXTSTEP	Plan 9	Open Desktop	ReliantUnix	OSF/1
QNX RTOS	QNX/Neutrino	QUNIX	SCO UnixWare	QNX
RISC iX	Security-Enhanced Linux	SCO UNIX	Trusted IRIX/B	SCO Xenix
SCO Xenix System V/386 SPIX	SunOS	Sinix	UNIX System III	Solaris
Trusted Xenix	Ultrix-11	Tru64 Unix	UNIX System V Release 4	Ultrix
Ultrix 32M		UnixWare	UNSW	UNIX System IV
UNIX System V		Xenix OS	Xinu	xMach

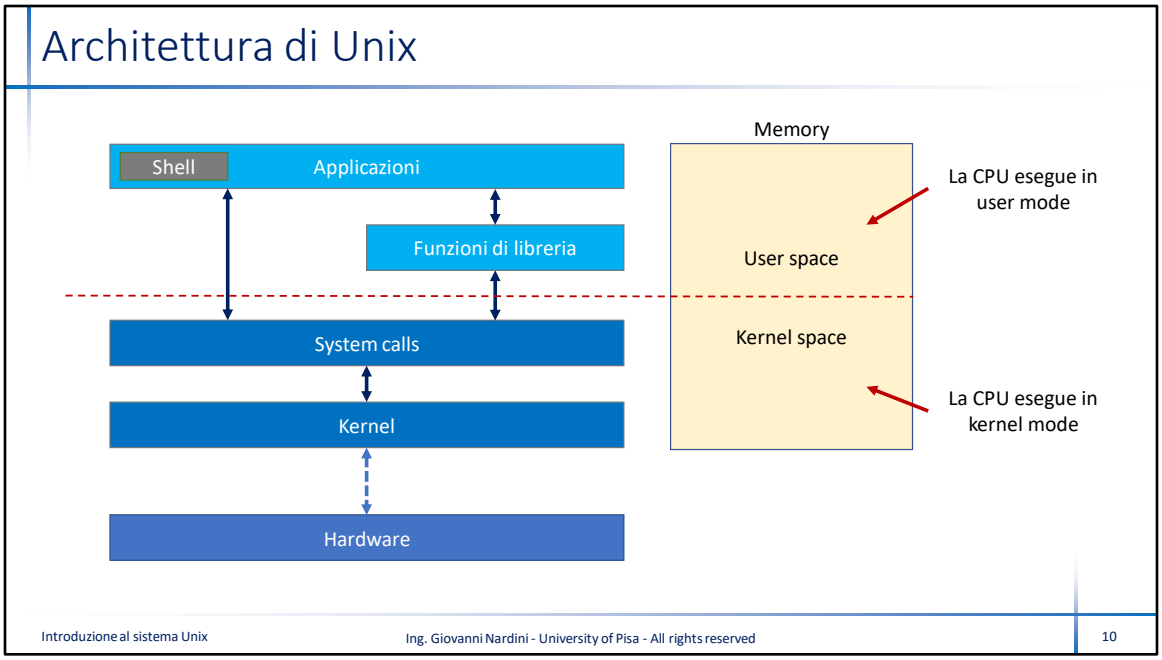
Alcune Unix, altre UNIX® e altre ancora "solo" Unix-Like

Unix (minuscolo) era il nome del SO originario di Thompson e Ritchie, ed è il termine generico con cui si identificano tutti i sistemi che derivano da esso.
UNIX (maiuscolo) è un marchio registrato di proprietà del consorzio «The Open Group», e solo i sistemi Unix che rispettano la Single UNIX Specification (SUS) possono dichiararsi sistemi UNIX (maiuscolo).
Unix-like sono i sistemi (come Linux) che non condividono codice con Unix, ma si comportano allo stedsso modo

La macchina virtuale che useremo durante il corso contiene una versione di Debian, che è una delle distribuzioni di Linux e perciò si tratta di un sistema Unix-like.

Perché Debian:
- Open source, e free to use (nota: open-source ≠ free)
- Semplice da usare
- Aggiornamenti rilasciati quando il sistema è stabile
- Ubuntu (distribuzione più popolare) si basa su Debian – modifica e aggiunge alcuni pacchetti software e fornisce aggiornamenti periodici e regolari

I concetti fondamentali



Il **kernel** è l'insieme dei software che gestiscono le risorse hardware del sistema di elaborazione ed espone le **system call**, ovvero delle funzioni (scritte in linguaggio C) che rappresentano l'interfaccia tra le applicazioni e le funzionalità del kernel stesso.
Le applicazioni utilizzano direttamente le system call per usare le risorse hardware del sistema, oppure possono utilizzare le **funzioni di libreria,** ovvero funzioni scritte e messe a disposizione da qualcun altro per svolgere un compito ben preciso (il programmatore non deve ogni volta «reinventare la ruota»). Le funzioni di libreria possono utilizzare - a loro volta - le system call per sfruttare le risorse hardware del sistema.

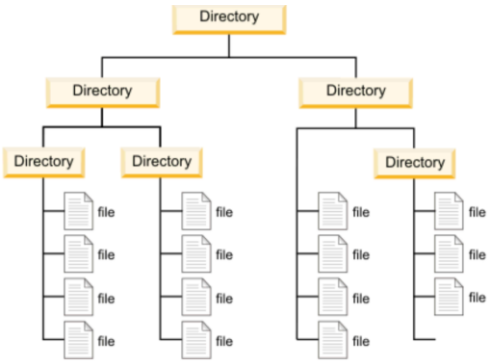
Quando si esegue un programma, il suo codice e i suoi dati vengono caricati nella memoria principale del calcolatore, in uno spazio chiamato **user space**. La CPU esegue le istruzioni del programma prelevandole dalla memoria e le esegue in **user mode (**CPU usa un insieme ristretto di istruzioni e può accedere solo alla porzione di memoria assegnata al programma che sta eseguendo in quel momento).
Quando nel programma viene invocata una system call (direttamente o attraverso una funzione di libreria) la CPU passa preleva le istruzioni del kernel, che risiedono in un'area della memoria principale chiamata **kernel space**. Tali istruzioni vengono eseguite dalla CPU in **kernel mode** (la CPU non ha limitazioni sul set di istruzioni che può eseguire, e ha accesso privilegiato e illimitato a tutta la memoria del sistema)

Nel caso di Linux, sarebbe più corretto riferirsi a un sistema operativo GNU/Linux, dove GNU (acronimo di GNU's Not Unix) è un sistema operativo Unix-like inventato nel 1984.
GNU/Linux è quindi il sistema operativo nato dall'unione del kernel Linux e di pacchetti software di GNU

- Linux = kernel + interfaccia system calls
- GNU: fornisce un set di applicazioni (as esempio: shell, interfaccia grafica, compilatore gcc)

File system

- Un **file system** definisce come sono organizzati i dati nella memoria secondaria e come possono essere recuperati all'interno del sistema
- Lo **U**nix **F**ile **S**ystem (**UFS**) ha un struttura gerarchica ad albero

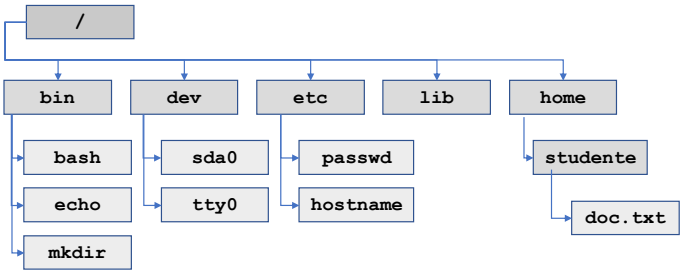


Senza un File System, i dati sarebbero soltanto un enorme blocco di bit senza modo di sapere dove inizia un pezzo di informazione e dove finisce. Lo scopo del File System, dunque, è di suddividere i dati in diversi «blocchi» e di associargli un nome: i **file**. Diversi tipi di File System si differenziano per struttura, logica e prestazioni (e.g., velocità di lettura/scrittura, dimensione massima dei file, politiche di sicurezza ecc.)

UFS usa una gerarchia ad albero: file raggruppari in **directory**, ciascuna directory può contenere file e altre directory (sotto-directory).

Unix File System

- **«Tutto è un file»**
  - Documenti, directory, dispositivi di I/O, interfacce di rete, ecc. sono file e possono essere acceduti usando le stesse system call
- In Unix, la directory **/** è la radice (root) del file system



Con file si intende un oggetto con cui si interagisce usando le stesse system call che si usano per scrivere/leggere dati su/da un file.

- quello che comunemente identifichiamo come un file viene detto **file regolare**
- una directory è un file «speciale» che contiene la lista dei file contenuti in quella directory
- dispositivi di I/O e le interfacce di rete sono rappresentate come file speciali. Una chiamata alla system call che scrive su questo oggetto, provoca l'esecuzione del driver del dispositivo.

Il vantaggio è che posso accedere a un file regolare/directory/dispositivo usando lo stesso insieme, limitato, di system calls.

La radice (**root**) è la directory base del file system che contiene tutte gli altri file e le altre directory, ed è indicata con "/". Sotto la root, ogni sistema Unix ha un insieme di directory «predefinite» che hanno un contennuto specifico:

- bin: file eseguibili del sistema
- dev: file che identificano i dispositivi
- etc: file di configurazione del sistema
- lib: file con librerie di sistema, moduli del kernel e driver dei dispositivi
- home: contiene le directory degli utenti del sistema, che a loro volta includono i file degli utenti
- …

Per raggiungere un determinato file, si percorre tutto l'albero partendo dalla root e attraversando le sottodirectory del **percorso.** In generale, qualunque file viene identificato tramite il suo percorso.

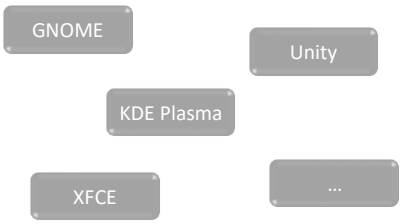
Definizione dei percorsi nel file system

- **Percorso assoluto**
  - Il nome del file viene specificato a partire dalla root `/`
  - `/home/studente/Documents/doc.txt`
  - I nomi dei file non possono essere `/` oppure `NULL`
- **Percorso relativo**
  - Il nome del file viene specificato a partire dalla directory in cui ci si trova attualmente
  - `Documents/doc.txt`
- Nomi speciali
  - `.` Directory corrente
  - `..` Directory parent
  - `~` Home dell'utente

Un percorso è una stringa (sequenza di caratteri) composta dai nomi delle directory e dei file – separati dal carattere / - attraversati per raggiungere il file.

Percorso assoluto → specifica tutta la sequenza di directory, a partire dalla root.
Percorso relativo → specifica tutta la sequenza di directory, a partire dalla directory in cui ci si trova attualmente (directory corrente)

Interfaccia grafica



- L'utente del sistema può richiedere servizi al sistema operativo utilizzando un'interfaccia grafica
- **GUI**: **G**raphical **U**ser **I**nterface
  - Chiamato anche Desktop Environment
- Intuitiva, facile da usare
- Alcune funzionalità avanzate non possono essere utilizzate

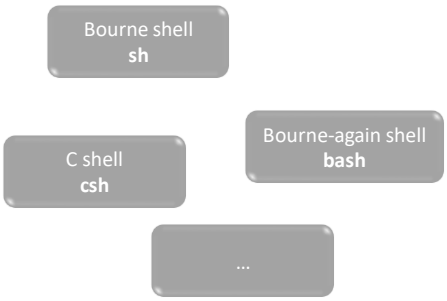
Basta navigare sul File Manager per ritrovare tutti i file e le applicazioni che ci servono.
Esistono molte GUI diverse, ciascuna con le sue caratteristiche: in base all'utenza di riferimento alcune GUI presentano funzionalità più semplici o la possibilità di compiere operazioni più avanzate.

L'interfaccia grafica è un'applicazione del SO, non fa parte del kernel.
La versione di Debian installata nella macchina virtuale usa GNOME, che è l'acronimo di **G**NU **N**etwork **O**bject **M**odel **E**nvironment (infatti GNU non fornisce il kernel del sistema GNU/Linux, ma il software soprastante)

Nel nostro caso, la GUI non ci dà l'accesso a tutte le funzionalità che possono interessarci in veste di utenti avanzati del sistema.
Inoltre:

- non tutti i SO hanno una GUI
- se volete accedere in remoto a un server, molto probabilmente non potrete farlo tramite un'interfaccia grafica user-friendly.

Shell



Una **shell** è un interprete dei comandi (*command-line interpreter*) che legge l'input dell'utente e esegue i comandi associati, eventualmente usando le system calls del sistema operativo

Anche la shell è un'applicazione del sistema operativo che non fa parte del kernel

Interprete dei comandi: programma composto da un ciclo infinito che legge input, esegue i comandi associati e produce un output

Diverse shell disponibili: ciascuna di esse specifica una sua sintassi e un suo linguaggio di programmazione (è infatti possibile scrivere degli script, come veri e propri programmi, che possono essere eseguiti per automatizzare dei comandi). Nel caso di Debian, la shell di default è **bash**, che è anche la più diffusa