

Sistemi operativi: struttura e servizi

Pietro Braione

Reti e Sistemi Operativi – Anno accademico 2025-2026

Argomenti

- A che servono i sistemi operativi?
- Requisiti per i sistemi operativi
- Struttura e servizi dei sistemi operativi
- Chiamate di sistema ed API
- I programmi di sistema

A che servono i sistemi operativi?

Cosa sappiamo sui sistemi operativi...

I più importanti per computer desktop/laptop sono Windows, macOS e Linux , mentre per i sistemi mobile sono Android e iOS

Di solito fornisce un ambiente a finestre per interagire con il computer

Ci permette di eseguire più applicazioni contemporaneamente, anche più dei core che abbiamo sul computer

Il sistema operativo è il primo programma che viene eseguito quando viene acceso il computer

Ci permette di installare le applicazioni che ci interessano

Mantiene ed organizza i nostri dati sotto forma di file e cartelle

Un sistema operativo...

- **Cos'è:**

- È un insieme di programmi (software)
- Che gestiscono gli elementi fisici di un computer (hardware)

- **A cosa serve:**

- Fornisce una piattaforma di sviluppo per le applicazioni, che permette loro di **condividere** ed **astrarre** le risorse hardware
- Agisce da intermediario tra utenti e computer, permettendo agli utenti di **controllare** l'esecuzione dei programmi applicativi e l'assegnazione delle risorse hardware ad essi
- **Protegge** le risorse degli utenti (e dei loro programmi) dagli altri utenti (e dai loro programmi) e da eventuali attori esterni

Spieghiamo un po' meglio...

- Dal punto di vista delle applicazioni, un sistema operativo è una **piattaforma di sviluppo**, ossia un insieme di funzionalità software che i programmi applicativi possono usare
- Tali funzionalità permettono ai programmi applicativi di poter usare in maniera conveniente le risorse hardware e di dividerle:
 - Da un lato, il sistema operativo **astrae** le risorse hardware, presentando agli sviluppatori dei programmi applicativi una versione delle risorse hardware più facile da usare e più potente rispetto alle risorse hardware «native»
 - Dall'altro, il sistema operativo **condivide** le risorse hardware tra molti programmi, permettendone l'esecuzione contemporanea

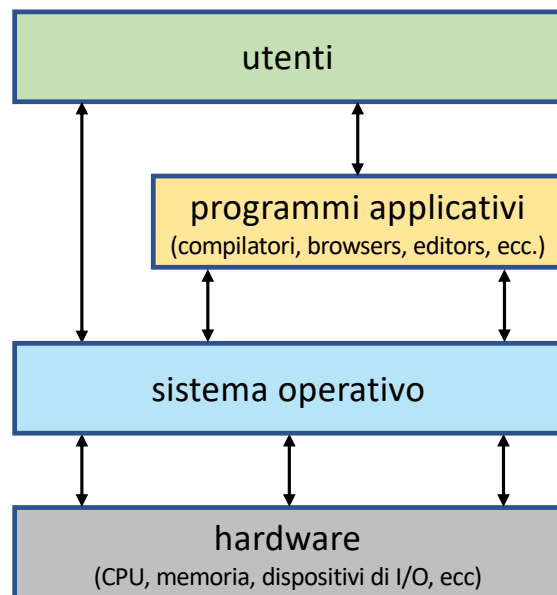
Spieghiamo un po' meglio...

- Dal punto di vista degli utenti un sistema operativo è un **sistema di gestione** e un'**interfaccia** verso le risorse e le applicazioni del computer
- Permette agli utenti di interagire con il computer e di fare diverse operazioni:
 - Installare / disinstallare applicazioni
 - Eseguire / terminare applicazioni
 - Assegnare risorse hardware (ad esempio, più o meno memoria) alle applicazioni
 - Conservare / organizzare dati
 - ...

Spieghiamo un po' meglio...

- Dal punto di vista del computer, un sistema operativo è un **assegnatore di risorse**
- Assegna le risorse hardware agli utenti e ai loro programmi suddividendole in maniera equa ed efficiente e controllando che questi le usino correttamente

Componenti di un sistema di elaborazione



- Utenti: persone (ma anche macchine, altri computer...)
- Programmi applicativi: risolvono i problemi di calcolo degli utenti
- Sistema operativo: coordina e controlla l'uso delle risorse hardware
- Hardware: risorse di calcolo (CPU, periferiche, memorie di massa...)

Requisiti per i sistemi operativi

Cosa si richiede ad un sistema operativo? (1)

- Oggigiorno i computer sono ovunque: vi sono molteplici tipologie di computer utilizzati in scenari applicativi diversi
- In quasi tutti i tipi di computer si tende ad installare un sistema operativo allo scopo di gestire l'hardware e semplificare la programmazione
- Ma ogni scenario applicativo in cui viene usato un computer richiede che il sistema operativo che vi viene installato abbia caratteristiche ben determinate

Cosa si richiede ad un sistema operativo? (2)

- **Server, mainframe:** massimizzare la performance, rendere equa la condivisione delle risorse tra molti utenti
- **Laptop, desktop, tablet:** massimizzare la facilità d'uso e la produttività della singola persona che lo usa
- **Dispositivi mobili:** ottimizzare i consumi energetici e la connettività
- **Sistemi embedded:** funzionare senza, o con minimo, intervento umano e reagire in tempo reale agli stimoli esterni (interrupt)

La maledizione della generalità

- Nella storia (ed anche oggi) alcuni sistemi operativi sono stati utilizzati per scenari applicativi diversi
- Ad esempio, Linux è usato nei server, nei computer desktop e nei dispositivi mobili (come parte di Android)
- La **maledizione della generalità** afferma che, se un sistema operativo deve supportare un insieme di scenari applicativi troppo ampio, non sarà in grado di supportare nessuno di tali scenari particolarmente bene
- Questo si è visto con OS/360, il primo sistema operativo che doveva supportare una famiglia di computer diversi (la linea 360 dell'IBM)

Politiche e meccanismi

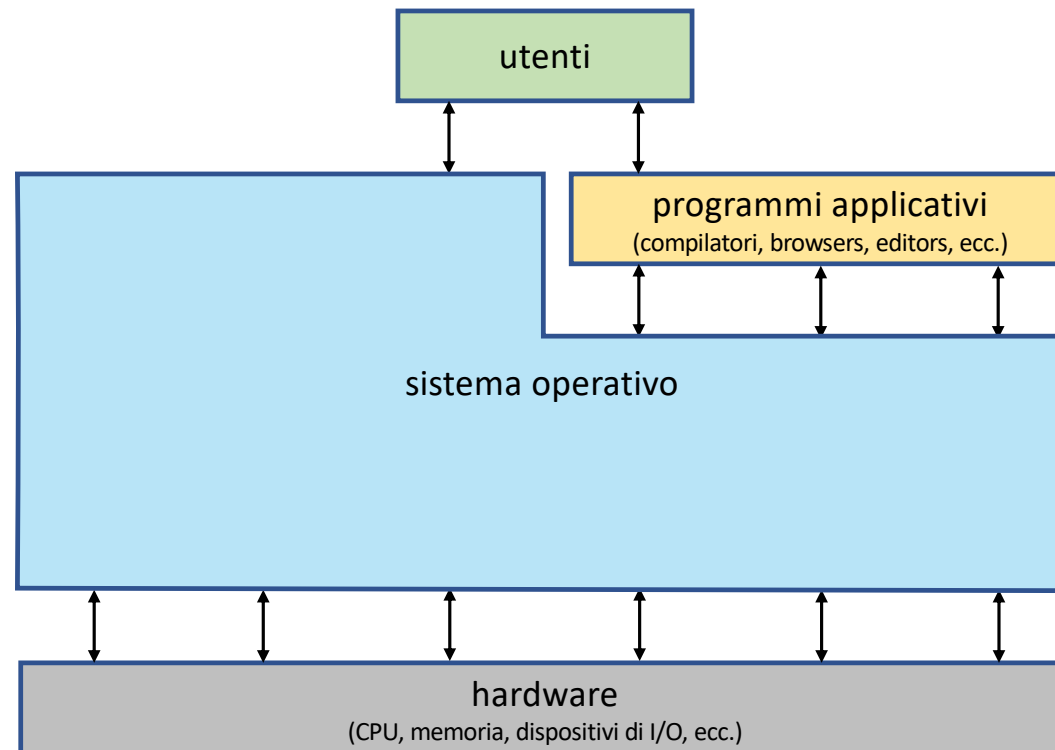
- Un principio importante è distinguere tra **meccanismi** e **politiche**
- Un meccanismo spiega *come* una certa operazione è effettuata; ad esempio: come fa il sistema operativo a sospendere, e poi riprendere, l'esecuzione di un programma?
- Una politica dice *che cosa* deve essere fatto in certe situazioni; ad esempio: quando è il momento di sospendere l'esecuzione di un programma e di mandare in esecuzione un altro?
- Le politiche impattano profondamente sulle caratteristiche percepite del sistema di elaborazione, i meccanismi no
- I meccanismi sono più stabili delle politiche, che possono cambiare a seconda delle caratteristiche percepite che vogliamo che il sistema di elaborazione abbia
- Avere politiche configurabili è utile per combattere la maledizione della generalità

Struttura e servizi dei sistemi
operativi

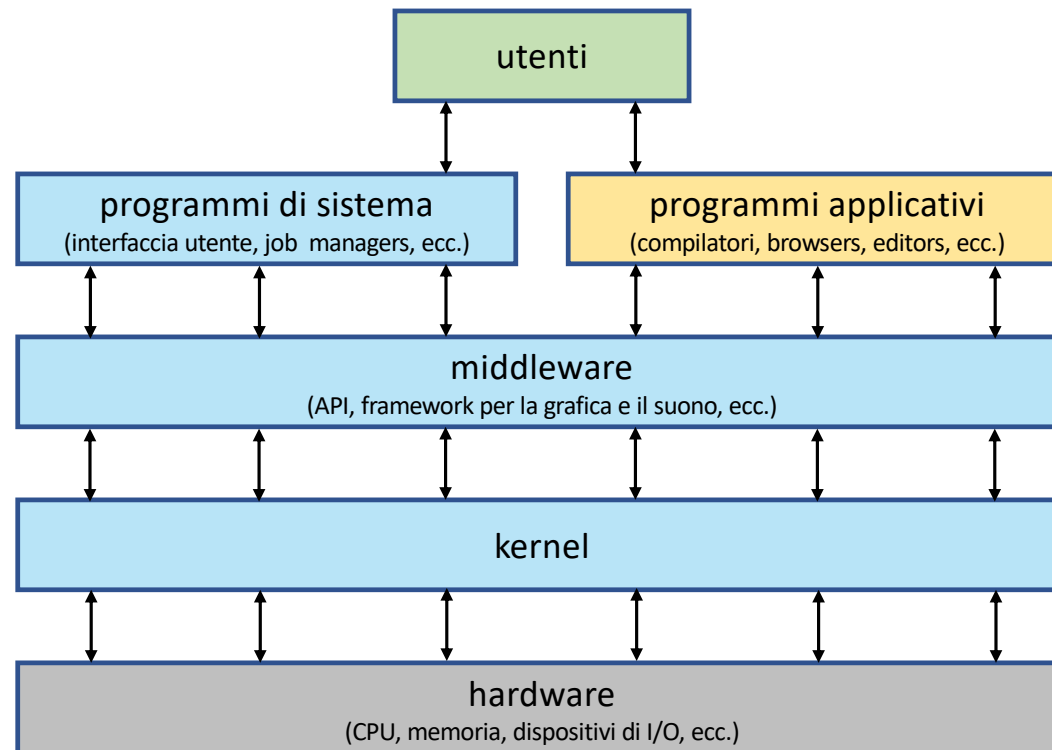
Struttura dei sistemi operativi

- Non c'è una definizione universalmente accettata di quali programmi fanno parte di un sistema operativo
- In generale però un sistema operativo comprende almeno:
 - **Kernel:** il «programma sempre presente», che «si impadronisce» dell'hardware, lo gestisce, ed offre ai programmi i servizi per poterlo usare in maniera condivisa ed astratta
 - **Middleware:** servizi di alto livello che astraggono ulteriormente i servizi del kernel e semplificano la programmazione di applicazioni (API, framework per grafica e per suono...)
 - **Programmi di sistema:** non sempre in esecuzione, offrono ulteriori funzionalità di supporto e di interazione utente con il sistema (gestione di jobs e processi, interfaccia utente...)
- Alcuni sistemi operativi forniscono anche dei programmi applicativi (editor, word processor, fogli di calcolo...), ma non li considereremo parti del sistema operativo stesso

Componenti di un sistema di elaborazione, rivisitati



Componenti di un sistema di elaborazione, rivisitati



Servizi offerti da un sistema operativo

- Un sistema operativo offre un certo numero di **servizi**:
 - Per i **programmi applicativi**: perché possano eseguire sul sistema di elaborazione usando le risorse astratte esposte dal sistema operativo
 - Per gli **utenti**: per gestire l'esecuzione dei programmi e stabilire a quali risorse hardware i programmi (e gli altri utenti) hanno diritto
 - Per garantire che il sistema di elaborazione funzioni in maniera efficiente
- Gli utenti però interagiscono con il sistema operativo attraverso i programmi di sistema...
- ...i quali utilizzano gli stessi servizi dei programmi applicativi...
- Quindi, in definitiva, il sistema operativo ha bisogno di esporre i suoi servizi esclusivamente ai programmi (applicativi o di sistema)

Principali servizi (1)

- **Controllo processi:** questi servizi permettono di caricare in memoria un programma, eseguirlo, identificare la sua terminazione e registrarne la condizione di terminazione (normale o errorea)
- **Gestione file:** questi servizi permettono di leggere, scrivere, e manipolare files e directory
- **Gestione dispositivi:** questi servizi permettono ai programmi di effettuare operazioni di input/output, ad esempio leggere da/scrivere su un terminale
- **Comunicazione tra processi:** i programmi in esecuzione possono collaborare tra di loro scambiandosi informazioni: questi servizi permettono ai programmi in esecuzione di comunicare

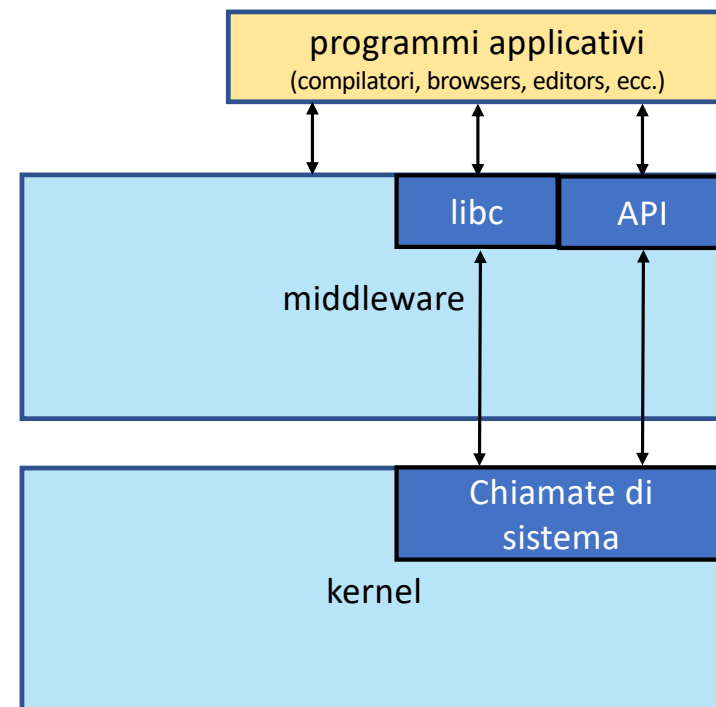
Principali servizi (2)

- **Protezione e sicurezza:** permette ai proprietari delle informazioni in un sistema multiutente o in rete di controllarne l'uso da parte di altri utenti e di difendere il sistema dagli accessi illegali
- **Allocazione delle risorse:** alloca le risorse hardware (CPU, memoria, dispositivi di I/O) ai programmi in esecuzione in maniera equa ed efficiente
- **Rilevamento errori:** gli errori possono avvenire nell'hardware o nel software (es. divisione per zero); quando avvengono il sistema operativo deve intraprendere opportune azioni (recupero, terminazione del programma o segnalazione della condizione di errore al programma)
- **Logging:** mantiene traccia di quali programmi usano quali risorse, allo scopo di contabilizzarle

Chiamate di sistema ed API

Chiamate di sistema ed API

- Il kernel offre i propri servizi ai programmi come **chiamate di sistema**, ossia di procedure invocabili in un determinato linguaggio di programmazione (C, C++...)
- I programmi però non utilizzano direttamente le chiamate di sistema, ma delle librerie di middleware dette **Application Program Interface (API)**
- Spesso le API sono fortemente legate con le librerie standard del linguaggio di implementazione (es. libc se le API sono implementate in C), al punto che anche queste diventano una parte implicita dell'API



Differenza tra chiamate di sistema ed API

- Le API sono esposte dal middleware, le chiamate di sistema dal kernel
- Le API sono implementate in linguaggi ad alto livello (di solito C o C++), le chiamate di sistema sono implementate in linguaggi ad alto livello e in assembler
- Le API invocano le chiamate di sistema nella loro implementazione
- Le API sono standardizzate (esempio: standard POSIX e Win32), le chiamate di sistema no (ogni kernel ha le sue)
- Le API sono stabili, le chiamate di sistema possono variare al variare della versione del sistema operativo
- Le API offrono funzionalità più ad alto livello e più semplici da usare, le chiamate di sistema offrono funzionalità più elementari e più complesse da usare
- Un sistema operativo può offrire più API di standard diversi!

Astrazioni

- Le risorse esposte attraverso le chiamate di sistema e le API sono delle versioni astratte (**astrazioni**) delle risorse hardware corrispondenti
- Le astrazioni sono più semplici da usare delle risorse hardware perchè nascondono (astraggono) dei dettagli che rendono problematico l'uso diretto delle risorse hardware, ed aggiungono caratteristiche desiderabili:
 - Processi al posto di processori: astraggono dal numero
 - Files al posto di blocchi su memorie secondarie: astraggono da dimensione / posizione, aggiungono metadati (nome, data creazione...)

I programmi di sistema

Programmi di sistema

- La maggior parte degli utenti utilizza i servizi del sistema operativo attraverso i programmi di sistema
- Questi permettono agli utenti di avere un ambiente più conveniente per l'esecuzione dei programmi, il loro sviluppo, e la gestione delle risorse del sistema

Tipologie di programmi di sistema (1)

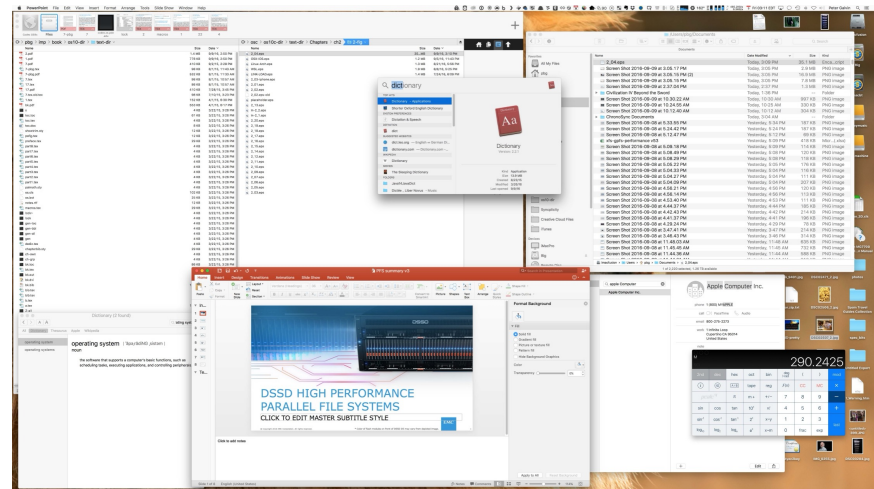
- **Interfaccia utente (UI):** permette agli utenti di interagire con il sistema stesso; può essere grafica (GUI) o a riga di comando (CLI); i sistemi mobili hanno un'interfaccia touch
- **Gestione file:** creazione, modifica, e cancellazione file e directory
- **Modifica dei file:** editor di testo, programmi per la manipolazione del contenuto dei file
- **Visualizzazione e modifica informazioni di stato:** data, ora, memoria disponibile, processi, utenti... fino informazioni complesse su prestazioni, accessi al sistema e debug. Alcuni sistemi implementano un **registry**, ossia un database delle informazioni di configurazione

Tipologie di programmi di sistema (2)

- **Caricamento ed esecuzione dei programmi:** loader assoluti e rilocabili, linker, debugger
- **Ambienti di supporto alla programmazione:** compilatori, assembleri, debugger, interpreti per diversi linguaggi di programmazione
- **Comunicazione:** forniscono i meccanismi per creare connessioni tra utenti, programmi e sistemi; permettono di inviare messaggi agli schermi di un altro utente, di navigare il web, di inviare messaggi di posta elettronica, di accedere remotamente ad un altro computer, di trasferire file...
- **Servizi in background:** lanciati all'avvio, alcuni terminano, altri continuano l'esecuzione fino allo shutdown. Forniscono servizi quali verifica dello stato dei dischi, scheduling di jobs, logging...

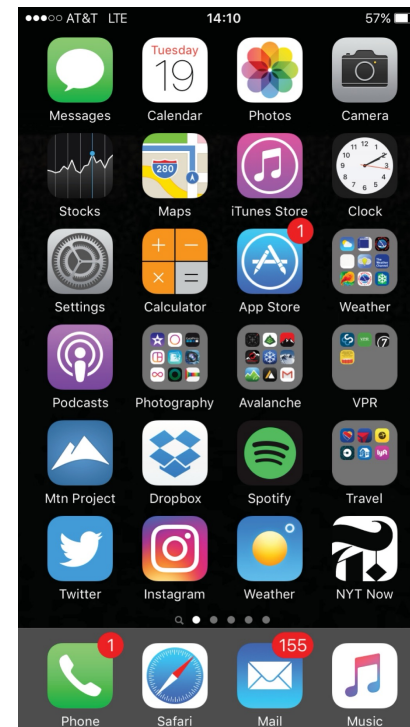
Interfaccia utente: le interfacce grafiche

- L'interfaccia grafica (GUI) è di solito basata sulla metafora della scrivania, delle icone e delle cartelle (corrispondenti alle directory)
- Nate dalla ricerca presso lo Xerox PARC lab negli anni 70, popolarizzate dal computer Apple Macintosh negli anni 80
- Su Linux le più popolari sono Gnome e KDE



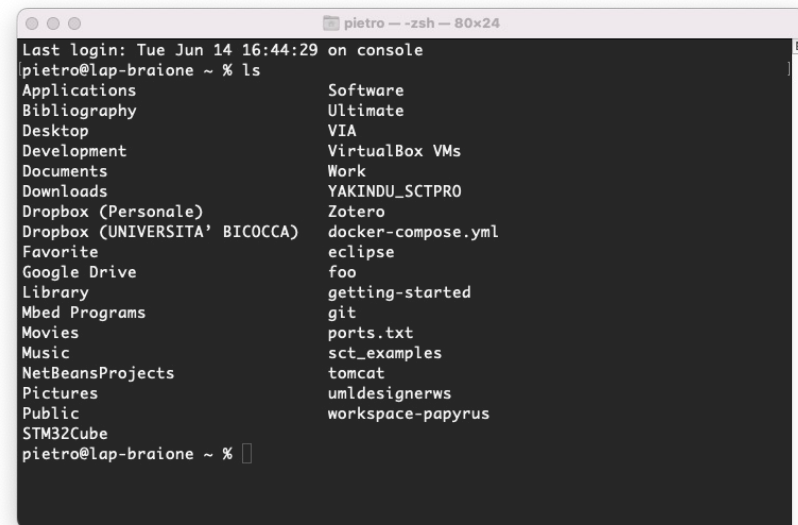
Interfaccia utente: Le interfacce touch-screen

- I dispositivi mobili richiedono interfacce di nuovo tipo
- Nessun dispositivo di puntamento (mouse)
- Uso dei gesti (gestures)
- Tastiere virtuali
- Comandi vocali



Interfaccia utente: l'interprete dei comandi

- L'interprete dei comandi permette agli utenti di impartire in maniera testuale delle istruzioni al sistema operativo
- In molti sistemi operativi è possibile configurare quale interprete dei comandi usare, nel qual caso è detto **shell**
- Due modi per implementare un comando:
 - Built-in: l'interprete esegue direttamente il comando (tipico nell'interprete di comandi di Windows)
 - Come programma di sistema: l'interprete manda in esecuzione il programma (tipico delle shell Unix e Unix-like)
- Spesso riconosce un vero e proprio linguaggio di programmazione con variabili, condizionali, cicli...



```
pietro@lap-braione ~ % ls
Applications      Software
Bibliography      Ultimate
Desktop           VIA
Development       VirtualBox VMs
Documents         Work
Downloads         YAKINDU_SCTPRO
Dropbox (Personale) Zotero
Dropbox (UNIVERSITA' BICOCCA) docker-compose.yml
Favorite          eclipse
Google Drive      foo
Library           getting-started
Mbed Programs     git
Movies            ports.txt
Music             sct_examples
NetBeansProjects  tomcat
Pictures          umldesignerws
Public           workspace-papyrus
STM32Cube
```


L'implementazione dei programmi di sistema

- I programmi di sistema sono implementati utilizzando le API, esattamente come i programmi applicativi
- Consideriamo ad esempio il comando `cp` delle shell dei sistemi operativi Unix-like:
 - `cp in.txt out.txt`
 - Copia il contenuto del file `in.txt` in un file `out.txt`
 - Se il file `out.txt` esiste, il contenuto precedente viene cancellato, altrimenti `out.txt` viene creato
- È implementato come programma di sistema
- Una possibile struttura del codice è riportata sulla destra (le invocazione delle API sono riportate in grassetto)

- **Apri** `in.txt` in lettura
- Se non esiste
 - **Scrivi** un messaggio di errore su terminale
 - **Termina** il programma con codice errore
- **Apri** `out.txt` in scrittura
- Se non esiste, **crea** `out.txt`
- Loop
 - **Leggi** da `in.txt`
 - **Scrivi** su `out.txt`
- End loop
- **Chiudi** `in.txt`
- **Chiudi** `out.txt`
- **Termina** normalmente