

# TPS

*(Tecnologie e Progettazione di Sistemi  
Informatici e di Telecomunicazioni)*

## Modulo 4 Il Sistema Operativo

Istituto Tecnico - Indirizzo Informatica & Telecomunicazioni  
Articolazione Informatica - Classi Terze

## Panoramica

- [Generalità sui sistemi operativi](#)
- [Evoluzione dei sistemi operativi](#)
- [La gestione del processore](#)
- [La gestione della memoria](#)
- [Il file system](#)
- [Struttura e realizzazione del file system](#)
- [La sicurezza del file system](#)
- [La gestione dell'I/O](#)
- [Video](#)

## Generalità sui sistemi operativi

All'accensione, il primo codice ad essere eseguito (BIOS) è contenuto nella ROM ed effettua:

1. POST (Power On Self Test) dei vari moduli componenti il computer
  - scheda madre
  - dimensione memoria RAM a disposizione
  - tastiera, mouse, ecc
  - hard disk, eventuali stampante, modem, ecc
2. Caricamento del kernel (nucleo del sistema operativo) in RAM

## Generalità sui sistemi operativi

A questo punto, il sistema è pronto ad eseguire i programmi utente.

*Definizione: Il sistema operativo è un insieme di programmi che gestiscono il funzionamento del computer agendo come intermediario fra l'utente ed il calcolatore.*

Su una data macchina si possono anche installare più sistemi operativi partizionando opportunamente il disco rigido.

## Funzioni del sistema operativo

Il S.O. svolge fondamentalmente due funzioni:

1. Gestisce le risorse HW (CPU, memoria, periferiche)
2. Fornisce il supporto per l'esecuzione dei comandi utente

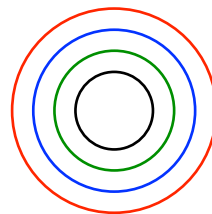
Il S.O. è memorizzato sull'hard disk e solo la sua parte principale (kernel o nucleo) risiede in memoria; gli altri moduli vengono caricati solo quando necessario.

## Struttura del sistema operativo

Il S.O. si può immaginare come successivi strati (onion skins) avvolti concentricamente intorno all'HW.

Dall'esterno verso l'interno abbiamo quindi:

- Shell
- Funzioni di base
- Nucleo
- Hardware



L'utente quindi interagisce solo con la shell, che nasconde i dettagli degli strati sottostanti.

## Il kernel

Interagisce con i programmi applicativi che quindi non possono accedere direttamente all'HW.

Offre un insieme di funzionalità dette *primitive di sistema*.

Questa organizzazione permette all'utente di non doversi preoccupare dei dettagli dell'HW, che sono mascherati dal S.O.

## La shell

È un programma che permette all'utente di interagire in modo (più o meno) naturale ed intuitivo con la macchina.

Viene anche detta interfaccia utente.

Vi sono vari stili di interfacce utente:

- CLI (Command Line Interface)  
(es.: le varie shell di UNIX)
- GUI (Graphical User Interface)  
(es.: Windows, OS X, Linux)
- CUI (Conversational User Interface)  
(il futuro ...)

## I S.O. più diffusi

Per gestire reti (server di rete):

- Linux
- Windows Server

Per PC:

- Windows
- OS X
- Linux

Per mini e mainframe:

- UNIX
- Linux
- Altri sistemi operativi (proprietary)

## I S.O. più diffusi

In realtà, i più diffusi S.O. sono quelli per dispositivi mobili (smartphone/tablet)

Solo nel 3Q2013 ne sono state vendute 261M unità così ripartite:

- |                 |       |
|-----------------|-------|
| – Android       | 81%   |
| – iOS           | 12.9% |
| – Windows Phone | 3.6%  |
| – Blackberry    | 1.6%  |

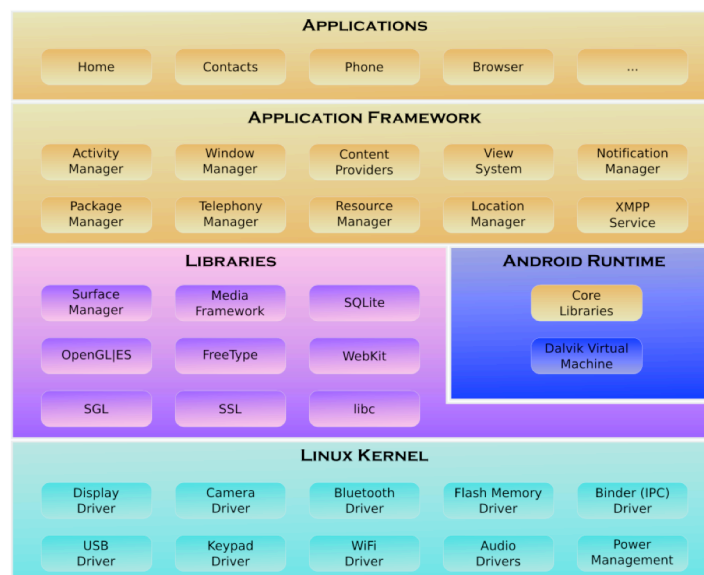
In particolare, Android viene utilizzato anche su:

PC portatili, netbook, lettori mp3/mp4, lettori di ebook, fotocamere, smart TVs, smart watches, smart glasses, navigatori, console di gioco, telecamere, frigoriferi (!), specchi (!!!)

## Android - architettura

- HW: basato su architettura ARM (x86 su alcuni apparecchi)
- Kernel: derivato da Linux con pesanti modifiche per adattarlo ad un ambiente operativo molto diverso da quello di un normale computer
- Applicazioni: principalmente scritte in Java, eseguite all'interno di una Virtual Machine (Dalvik, non Java, per motivi di compattezza ed efficienza)

## Android - architettura



## Android - ciclo di vita

- Ogni versione, oltre al numero di release, ha un codename corrispondente ad un dolcetto, in ordine alfabetico (cupcake, donut, eclair, froyo, gingerbread, ice cream sandwich, honeycomb, jelly bean, kit kat, ecc)
- Google rilascia i sorgenti per i suoi apparecchi (es.: Google Nexus)
- Gli altri produttori, se vogliono allinearsi, devono portare la nuova release sui loro apparecchi (ormai venduti e vecchi ...)

## Evoluzione dei sistemi operativi

Lettura autonoma pag. 179-190 del testo

## La gestione del processore

- **Programma**  
Entità statica, è il codice memorizzato su memoria di massa
- **Processo**  
Entità dinamica, istanza in continua evoluzione di un programma, residente in RAM.  
È costituito da
  - Codice (non modificabile => può essere condiviso fra più processi che eseguono lo stesso programma)
  - Dati (un set di dati per ogni istanza di uno stesso programma)

## Multitasking e Multiprocessing

- **Multitasking**  
Esecuzione di più processi da parte di un unico processore. Il parallelismo che si ottiene è solo apparente.
- **Multiprocessing**  
Multitasking distribuito su più processori. Il parallelismo è reale.



## Gestione dei processi

- **Process Descriptor (PD) o Process Control Block (PCB)**  
È la struttura dati che contiene tutte le informazioni necessarie al SO per gestire l'esecuzione in parallelo dei vari processi
- **Stato dei processi**  
Una fra le cinque possibili situazioni in cui si può trovare un processo in esecuzione

## Stato dei processi

1. **New**  
Appena creato (1 volta sola nella sua vita)
2. **Running**  
La CPU sta eseguendo il codice del processo (1 per processore)
3. **Waiting**  
In attesa che si verifichi un evento (che si liberi una risorsa)
4. **Ready-to-Run**  
Ha tutte le risorse, tranne la CPU
5. **Terminated**  
Finito di girare, in attesa di liberare le risorse prima di farlo scomparire (1 volta sola nella sua vita)

## Process Control Block (PCB)

Il PCB contiene, fra l'altro, le seguenti informazioni fondamentali per gestire ciascun processo:

- PID - l'identificatore del processo
- Stato - lo stato attuale del processo
- PC - il Program Counter
- Registri - i registri della CPU
- Priorità - la priorità attuale del processo
- Puntatori alla memoria del processo
- Puntatori alle risorse allocate dal processo (p.es. i file aperti)

## Process Scheduling

Per scheduling si intende la politica di scelta dell'assegnazione della CPU ai processi in Ready List.

Al cambio di processo running il *dispatcher* effettua un *context switch*:

- salva il contenuto dei registri nel PCB del processo che lascia la CPU
- recupera il contenuto dei registri dal PCB del processo che deve impadronirsi della CPU

## User mode e Kernel mode

User mode e Kernel (o supervisor) mode sono due diversi livelli di privilegio del codice in esecuzione.

- User mode: quando si esegue il codice del programma utente; in generale, il processo è interrompibile
- Kernel mode: quando si esegue il codice del routines interne del S.O. oppure le chiamate al sistema; in generale, il processo non è interrompibile

## Politiche di scheduling

### **Windows XP:**

- CPU scheduling con prelazione
- 32 livelli di priorità in Round Robin
- Due classi di priorità:
  - Real time (16..32)
  - Variable (1..15)
    - La priorità si abbassa all'esaurirsi del quanto di tempo
    - in seguito ad uno sblocco, la priorità viene aumentata (boosted)
    - i processi in foreground sullo schermo hanno una priorità più alta

## Politiche di scheduling

### **Linux:**

- CPU scheduling con prelazione
- 140 livelli di priorità
- Tre classi di priorità:
  - Real time 0..99  
(FCFS non pre-empted oppure RR pre-empted)
  - System thread 0..99  
(FCFS non pre-empted)
  - User 100..139  
(RR Pre-empted) priorità calcolata dinamicamente in base al grado di interattività

## Comunicazione fra processi

I metodi più diffusi di comunicazione fra processi sono basati su:

- Memoria condivisa (shared memory) ossia variabili o strutture dati condivise da più processi  
metodo usato tipicamente fra processi residenti su di una stessa macchina
- Scambio di messaggi (message passing) ossia invio di messaggi su canali di comunicazione  
Può essere usato anche fra processi residenti su macchine diverse

## La gestione della memoria

Classificazione delle memorie per tempo tipico di accesso e capacità:

<b>Tempo</b>		<b>Capacità</b>
1 ns	Registri	< 1 kB
qualche ns	Cache	1-8 MB
10 ns	RAM	2-8 GB
10 ms	Dischi magnetici	1-10 TB
100 s	Nastri magnetici	1-100 TB

## Il memory manager

Il memory manager si occupa principalmente di gestire l'utilizzo della RAM da parte dei processi.

- Sapere quali parti sono in uso e quali libere
- Allocare e deallocare le porzioni di memoria necessarie ai processi
- Gestire lo *swapping* tra RAM e disco quando la RAM non è sufficiente

## Indirizzi logico e fisico

Gli indirizzi di memoria generati da compilatore e linker devono essere *rilocabili* (indirizzi logici).

Al momento dell'attivazione di un nuovo processo, il *loader* trasforma tali indirizzi negli indirizzi *fisici* effettivi.

$$\mathbf{ind. \ fisico = ind. \ logico + offset}$$

Tale operazione prende il nome di *rilocazione* e può essere di due tipi:

- Rilocazione *statica*: l'offset è determinato una volta per tutte alla partenza del processo
- Rilocazione *dinamica*: l'offset è fornito dal contenuto di un registro (registro di rilocazione) e può quindi variare durante la vita del processo

## Indirizzi logico e fisico

Ciascun processo deve quindi "vedere" uno spazio di indirizzamento contiguo.

Alcune osservazioni:

- I programmi attuali hanno dimensioni spesso notevoli
- Il continuo caricamento e scaricamento di programmi porta ad una frammentazione della memoria
- Spesso solo una parte delle istruzioni (e dello spazio dati) di un programma vengono realmente utilizzate

Per ovviare a questi inconvenienti si possono adoperare diverse tecniche.

## Swapping, DLL, Overlay

### **Swapping**

Per liberare dello spazio di memoria RAM si possono scaricare su disco dei processi temporaneamente inattivi.

Si tratta però di un'operazione molto onerosa e lenta, da attuarsi solo in casi estremi.

### **DLL** (Dynamic Link Library)

Un programma molto ricco di funzionalità può essere suddiviso in un blocco principale di codice sempre presente in RAM ed in alcune DLL che restano su disco e vengono caricate in RAM solo se necessario.

## Swapping, DLL, Overlay

### **Overlay**

Se un programma contiene zone di codice che si escludono a vicenda, nel senso che non servono contemporaneamente, il programmatore può destinare loro la stessa zona di memoria, in cui quindi tali zone di codice sono sovrapposte (overlay); a run-time il loader caricherà solo la parte necessaria.

## Allocazione della memoria

### **Partizionamento**

La memoria viene suddivisa in *partizioni* che possono essere di dimensione fissa o variabile.

Si tratta di tecniche di gestione dell'allocazione della memoria (descritte sul libro a pagg. 213-216) piuttosto complesse e con limitazioni tali (limitazione del numero dei processi, frammentazione e deframmentazione della memoria, ricerca del miglior segmento cui associare un nuovo processo, lento e progressivo declino delle prestazioni, ecc).

Per questi motivi sono state superate dai più moderni sistemi a *Memoria Virtuale*.

## Allocazione della memoria

### **Memoria Virtuale**

Il SO mantiene in memoria solo le parti di codice e dati in uso e lascia su disco tutto il resto.

Questa strategia risulta vincente grazie al *principio di località*.

Lo spazio di indirizzamento di un programma e la dimensione del suo codice non sono quindi più vincolati dalla dimensione fisica della memoria.

Il programmatore può quindi scrivere il proprio codice disponendo di uno spazio di memoria (*virtuale*, non reale) limitato solo dall'architettura della CPU (p.es. 4GB per CPU a 32 bit).



## Memoria Virtuale - Paginazione

Sia i programmi sia la memoria centrale vengono divisi in *pagine* di dimensione fissa.

La memoria risulta quindi suddivisa in un numero fisso di *pagine fisiche*.

Ciascuna *pagina fisica* può essere libera oppure occupata da una pagina di codice o dati di un determinato processo.

Poiché una stessa pagina di codice può essere caricata in indirizzi diversi in tempi diversi, il codice deve essere *rilocabile dinamicamente*.

## Memoria Virtuale - Paginazione

La mappatura degli indirizzi da virtuale a fisico viene eseguita da un dispositivo HW apposito integrato nella CPU: la MMU (Memory Management Unit).

La struttura dati che contiene le informazioni necessarie per la mappatura si chiama *Page Table*.

La mappatura è totalmente a carico del SO, perciò risulta invisibile al SW applicativo.

Particolarmente importante il caso del *Page Fault* ed il suo trattamento.

La paginazione elimina il problema della frammentazione della memoria ed è usata nella maggior parte dei SO moderni.

## Memoria Virtuale - Paginazione

A ciascuna pagina fisica sono associati opportuni flag, fra cui:

- *locked* (aka *pinned*, *wired*, *fixed*): la pagina deve rimanere sempre in memoria (p.es. il codice che gestisce i page fault!)
- *modified*: la pagina è stata modificata (solo pagine dati)
- *referenced*: la pagina è stata recentemente usata (resettato periodicamente)

Tali flag vengono usati dagli algoritmi di sostituzione delle pagine (NRU, LRU) quando è necessario eliminare una pagina per fare spazio ad un'altra

## Il File System

Il File System (FS) è la parte del SO che si occupa della gestione dei file.

In particolare il FS gestisce:

- la memorizzazione di file e cartelle sulle memorie di massa
- l'accesso a file e cartelle da parte dei programmi applicativi
- gestione di accessi concorrenti da parte di più processi
- la protezione da accessi non desiderati
- l'utilizzo di supporti diversi (magnetici, ottici, elettronici) astruendo dalle loro caratteristiche fisiche

## Il File System

L'elemento fondamentale è il *file*: un insieme di dati identificato da un nome e memorizzato su un supporto di memoria di massa.

Il contenuto di un file è costituito da una *sequenza di byte*, eventualmente aggregati a formare strutture dati più complesse.

L'interpretazione delle informazioni che un file contiene ed il modo in cui sono organizzate dipende dal tipo del file.

Un file infatti può contenere informazioni molto diverse: codice sorgente, codice eseguibile, testo, immagini, suoni, video, ecc

## Il File Descriptor

Per ogni file, il SO mantiene un insieme di informazioni, raccolte all'interno di una struttura dati detta File Descriptor (FD):

- nome
- identificatore
- tipo
- locazione
- dimensione
- data e ora di:
  - creazione
  - ultima modifica
  - ultimo accesso
- proprietario
- permessi

I FD a loro volta sono memorizzati sul supporto di memoria di massa in aree ad essi dedicate (directory)

## Operazioni sui file

Ogni FS mette a disposizione determinate operazioni, tipicamente:

- creazione
- scrittura
- lettura
- posizionamento (seek)
- troncamento
- accodamento
- cancellazione
- rinomina
- spostamento
- copia
- lock (parziale o totale, v. riquadro a pag. 228-229)

## Metodi di accesso

Per accedere ai dati memorizzati in un file vi sono essenzialmente due modalità di accesso:

- sequenziale, attraverso le operazioni:
  - reset
  - read next (record)
  - write next (record)
  - skip +/-n (record)
- diretto, attraverso le operazioni:
  - read n
  - write n
  - seek n
  - read next
  - write next

## Struttura della directory

Partizionamento hard disk

- suddivisione in unità distinte (volumi) ciascuna con il suo file system
- il primo settore contiene l'MBR (Master Boot Record) che a sua volta contiene la *Partition Table*
- ogni volume è poi strutturato ad albero
- concetti fondamentali:
  - dir corrente
  - filename, basename, extension
  - pathname relativo
  - pathname assoluto

## Sistemi multiutente

Vengono introdotti alcuni attributi aggiuntivi:

- proprietario
- gruppo (un utente può far parte di più gruppi)
- diritti d'accesso (*chi* può fare *cosa* su quel file o dir)
  - lettura
  - scrittura
  - esecuzione
  - accodamento
  - cancellazione
- realizzati tramite ACL (Access Control List)
- UNIX utilizza un sistema semplificato (user, group, world, read, write, execute)

## Struttura e realizzazione del File System

Organizzazione disco fisso: unità di memorizzazione è il *blocco* (tipicamente di 512 o 1024 byte) indirizzato come segue:

- storicamente C/H/S
  - tracce (cylinders)
  - facce (heads)
  - settori (sectors)
- attualmente LBA (Logical Block Addressing)
  - N blocchi numerati da 0 a N-1

*N.B.: I blocchi sono creati tramite la formattazione di basso livello*

## Struttura e realizzazione del File System

Su disco si trovano:

- *Boot control block*: info necessarie al boot
- *Partition control block* (aka: superblock): info sulle varie partizioni in cui è suddiviso il disco
- *Directory structures*: elenco dei file contenuti nella cartella
- *File descriptors*: informazioni collegati ad un file
- *Data blocks*: i dati contenuti nei file

In memoria si trovano:

- *Directory structures (cache)*
- *Open files table*
- *Open files table for each process*
- *Data blocks buffers (cache)*

## Allocazione di un File

- **Allocazione contigua**
  - Il file è una sequenza di blocchi contigui
  - ✦ accesso semplice e veloce
  - ✦ poche informazioni di supporto
  - trovare spazio contiguo su disco
  - frammentazione del disco
  - periodica ricompattazione

## Allocazione di un File

- **Allocazione concatenata**
    - Il file è una lista di blocchi, in fondo ad ogni blocco c'è un puntatore al blocco successivo.
    - ✦ no frammentazione
    - ✦ no blocchi contigui
    - accesso diretto lento (attraversare tutta la lista)
    - spazio aggiuntivo (molto poco)
    - se perdo 1 blocco => perdo tutto il resto del file [lista bi-direzionale]
- eventualmente raggruppati in cluster di 2 o 4 blocchi per aumentare l'efficienza

## Allocazione di un File

- Allocazione concatenata - FAT

È la variante introdotta dall'MS-DOS, OS/2 e Windows:

All'inizio di ogni partizione c'è una tabella detta *File Allocation Table (FAT)* contenente 1 entry per ogni blocco; la FAT contiene la struttura della lista, evitando di dover accedere ai blocchi dati per muoversi nel file.

- ✦ no frammentazione
- ✦ no blocchi contigui
- ✦ FAT sempre in memoria => tempi di accesso ridotti
- ✦ accesso diretto più veloce
- ✦ se perdo 1 blocco => perdo solo quel blocco
- dimensione della FAT non è costante (dipende dalla dimensione del disco)
- necessità di sincronizzare FAT in memoria con FAT su disco

## Allocazione di un File

- Allocazione indicizzata (blocco indice o i-node)

- l' i-node contiene l'elenco dei blocchi dati del file
- per accedere all'i-mo blocco basta accedere all'i-node e cercare l'i-mo elemento della lista dei blocchi
- il nome del file non fa parte dell'i-node, ma è contenuto nella directory insieme ad un puntatore all'i-node
- l'i-node può contenere solo un numero prefissato di riferimenti ai blocchi, quindi fino ad una data lunghezza di file
- se il file è più grande, l'i-node contiene dei puntatori ad altri i-node che a loro volta puntano ai blocchi dati



## La sicurezza del File System

Principali rischi riguardanti il File System

- cancellazioni accidentali
- guasti e malfunzionamenti HW
- guasti e malfunzionamenti SW
- eventi catastrofici (incendi, allagamenti)
- sottrazione di dati fisica oppure tramite intrusioni informatiche
- alterazione di dati tramite intrusioni informatiche

## La sicurezza del File System

È necessario attuare una strategia di sicurezza che permetta di:

- salvaguardare l'operatività in tempi brevi
- salvaguardare l'integrità dei dati

Tipicamente si agisce su due fronti:

- esecuzione di backup periodici
- ridondanza dei supporti magnetici (RAID)

## I backup

Si distinguono due tipi di backup:

- *Completo*: si salva l'intero contenuto del sistema
- *Incrementale*: si salva solo quello che è stato aggiunto/modificato dopo l'ultimo backup effettuato

La periodicità dei backup dipende dalle necessità dell'applicazione, tipicamente si procede ad un *backup completo settimanale* e ad un *backup incrementale giornaliero*.

## I livelli di RAID

- RAID 0: striping per aumentare la velocità, ma nessuna ridondanza
- RAID 1: mirroring (no parità né striping) aumento di velocità in lettura, rallentamento in scrittura
- RAID 2: striping a livello di bit con aggiunta di unità dedicate ad ECC (valore storico, mai usato)
- RAID 3: striping a livello di byte con parità dedicata (poco usato in pratica)
- RAID 4: striping a livello di blocco con parità dedicata

## I livelli di RAID

- RAID 5: striping a livello di blocco con parità *distribuita* - minimo 3 dischi - recupera il guasto di un singolo drive ricostruendone i dati sulla base degli altri drive - molto lento
- RAID 6: striping a livello di blocco con *doppia* parità *distribuita* - recupera il guasto di *due* drive - adatto per sistemi ad alta disponibilità

## La gestione dell'I/O

# Video

1. [Computer Operating Systems: Managing Hardware and Software Resources](#) (basic)
2. [OS Functions: Security, System Management, Communication and Hardware & Software Services](#) (basic)
3. [Enterprise, Workgroup & Personal Operating Systems](#)
4. [Files Systems: FAT, NTFS, and HFS+](#)