



Prof. Mario F. Pavone

The diagram illustrates the hierarchy of computer memory, structured as a pyramid with eight layers. The layers are labeled from top to bottom: CPU, CPU CACHE, PHYSICAL MEMORY, SOLID STATE MEMORY, and VIRTUAL MEMORY. To the right of the pyramid, there are three groups of labels: 'SUPER EXPENSIVE VERY CAPAC', 'FAST EXPENSIVE SMALL CAPAC', and 'FA PRICED REASONABLE MODERATE CAPAC'. To the left of the pyramid, there are three groups of labels: '0, 50, 100 MB, 1 GB, 1 TB, 1 PB, 1 EB, 1 ZB, 1 YB', '0, 1 MB, 1 GB, 1 TB, 1 PB, 1 EB, 1 ZB, 1 YB', and '0, 1 MB, 1 GB, 1 TB, 1 PB, 1 EB, 1 ZB, 1 YB'.

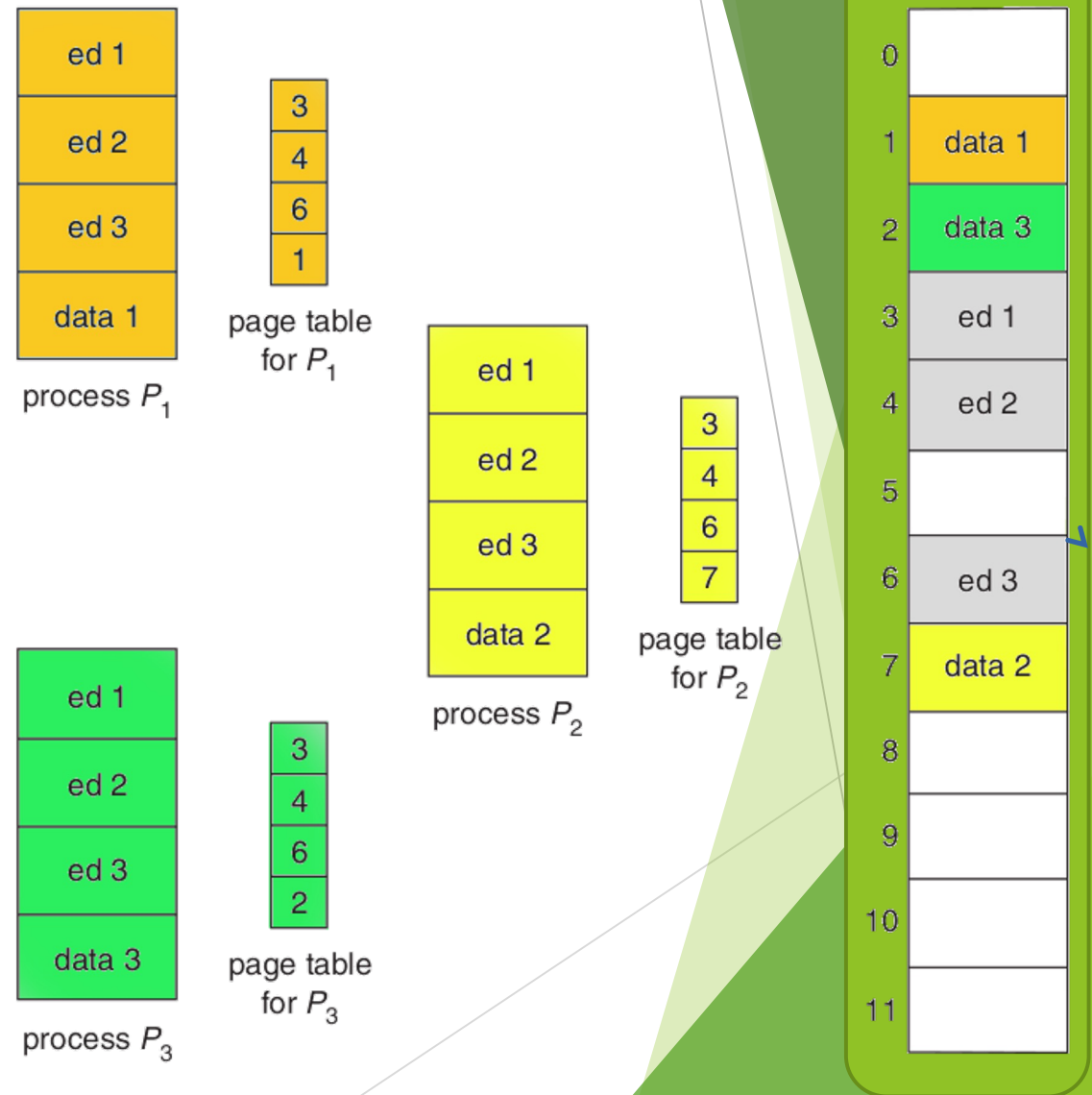
Memory Type	Speed	Cost	Capacity
CPU	Very Fast	Very Expensive	Very Small
CPU CACHE	Fast	Expensive	Small
PHYSICAL MEMORY	Medium	Reasonable	Moderate
SOLID STATE MEMORY	Fast	Expensive	Small
VIRTUAL MEMORY	Slow	Cheap	Large

Dimensione della Pagina

- ▶ La scelta della dimensione della pagina di base è importante:
 - ▶ vantaggi di una **pagina grande**:
 - ▶ tabella della pagine più piccola;
 - ▶ migliore efficienza nel trasferimento I/O;
 - ▶ tende a minimizzare il numero di page fault (minore overhead);
 - ▶ vantaggi di una **pagina piccola**;
 - ▶ minore frammentazione interna;
 - ▶ migliore risoluzione nel definire il working set in memoria (meno memoria sprecata);
- ▶ relazione con la **dimensione del blocco su disco**.

Pagine Condivise

- ▶ I processi posso anche condividere in vari modi alcune pagine:
 - ▶ solo lettura:
 - ▶ codice eseguibile condiviso (**codice rientrante**);
 - ▶ lettura/scrittura:
 - ▶ IPC tramite memoria condivisa;
- ▶ implementazione su **tabella delle pagine ordinaria o multilivello**:
 - ▶ semplice ed efficiente.



Pagine Condivise

► Gestione della cache:

- problemi di sincronizzazione con cache basate su indirizzi virtuali (anche se usano gli ASID);
- soluzioni:
 - disabilitare la cache sulle pagine condivise;
 - usare cache con ricerca basata su **indirizzi virtuali e tag fisici (VIPT)**:
 - la cache ricerca in parallelo con la TLB sulla base dell'indirizzo virtuale;
 - per capire se si tratta di un duplicato dobbiamo aspettare che la TLB dia in output l'indirizzo fisico;

Pagine Condivise

- ▶ *tabella delle pagine invertita:*
 - ▶ *singolo core*
 - ▶ alterazione tabella su context switch o su page fault;
 - ▶ *multi core*
 - ▶ difficilmente gestibile se non con teoriche tabelle delle pagine invertite con corrispondenze multi-a-uno.

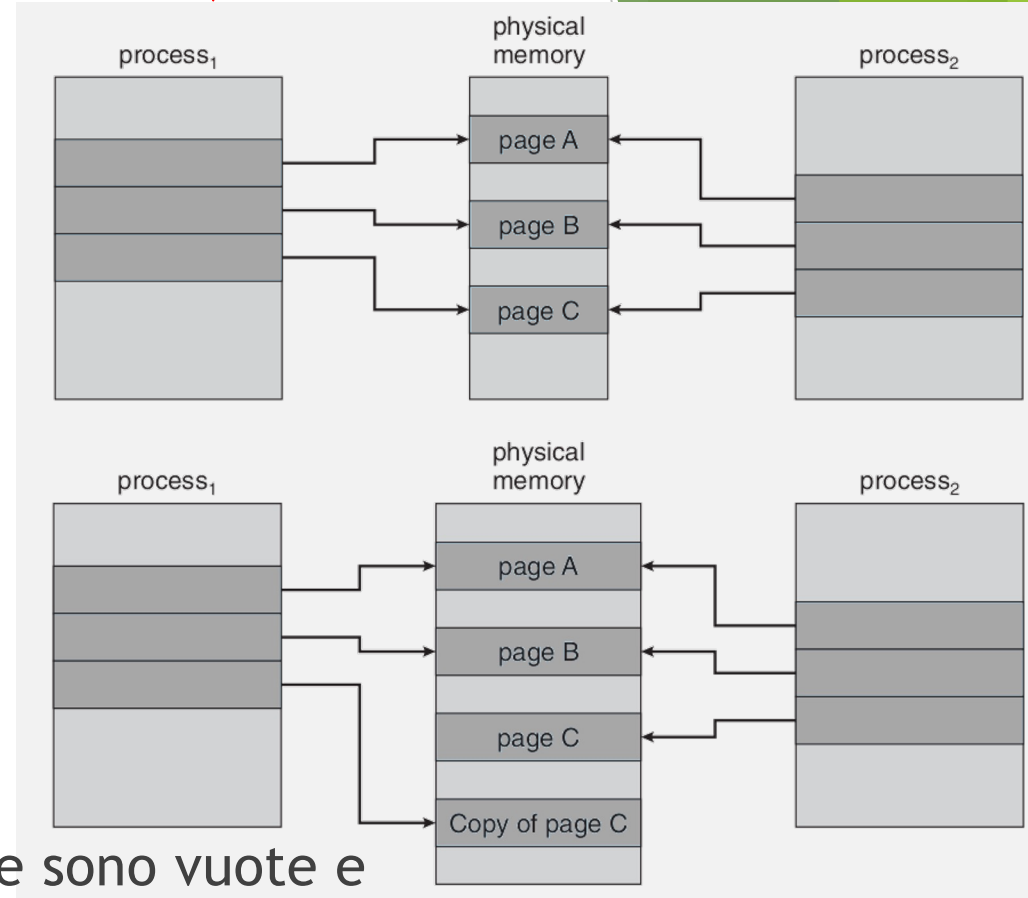
Copy-on-write & Zero-fill-on-demand

► copy-on-write:

- condivide finché possibile tutti i tipi di pagine (codice e dati);

► zero-fill-on-demand:

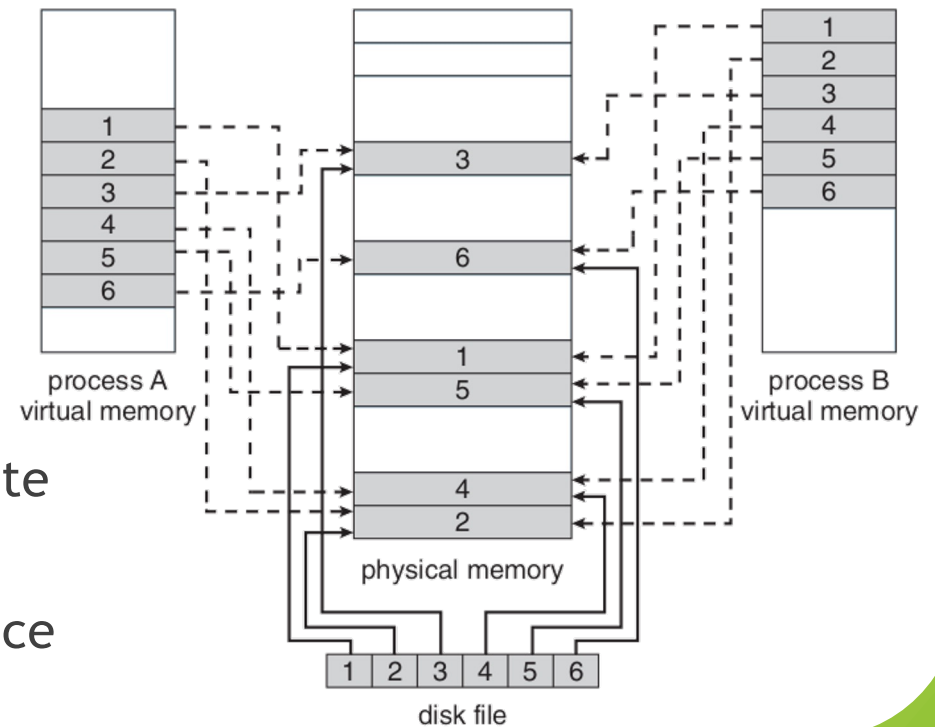
- principio di base: le nuove pagine sono vuote e allocate su richiesta;
 - azzeramento efficiente gestito dal kernel;
 - sicurezza;
 - prestazioni.
- copy-on-write su una *read-only static zero page*.



NOTA: riempire a zero le pagine può essere fatto quando il sistema è inattivo!!

Librerie Condivise

- ▶ *Grandi librerie* condivise sono comunemente usate;
 - ▶ **linking statico**: inclusione del codice in fase di linking;
 - ▶ **linking dinamico**: collegamento e caricamento a run-time di librerie condivise:
 - risparmio di spazio su disco e in RAM;
 - sviluppo indipendente e facilità di aggiornamento;



File Mappati

► File mappati in memoria:

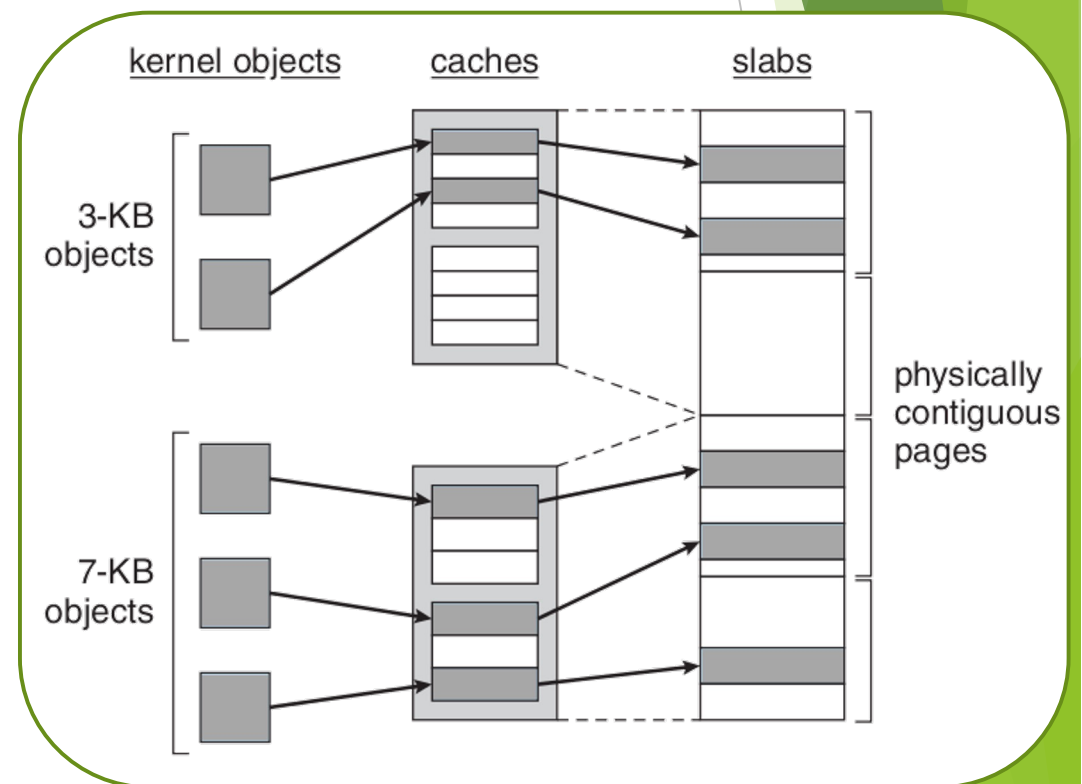
- modello alternativo di I/O su file;
- possibilità di condivisione;
- accesso al file più veloce;
- gestiscono automaticamente:
 - librerie condivise;
 - caricamento codice eseguibile;
 - caricamento dei dati statici.
- consentire a più processi di mappare lo stesso file contemporaneamente

Allocazione della Memoria per il Kernel

- ▶ **Memoria dei processi utente:** paginata ma con frammentazione interna;
- ▶ **Memoria interna al kernel:**
 - ▶ miriamo a **frammentazione interna** minima o nulla;
 - ▶ **impossibilità di paginare** l'allocazione in alcuni casi.

Allocazione della Memoria per il Kernel

- ▶ Soluzione: **slab allocation**;
 - ▶ **slab**: sequenza di pagine contigue;
 - ▶ **cache**: uno o più slab;
 - ▶ una cache per tipo di **struttura dati interna omogenea**;
- ▶ gestione:
 - ▶ stato di uno slab:
 - ▶ *pieno, vuoto, parziale*;
 - ▶ dinamica;
- ▶ **vantaggi**:
 - ▶ **niente spreco**;
 - ▶ **efficienza**.



SISTEMI OPERATIVI (M-Z)

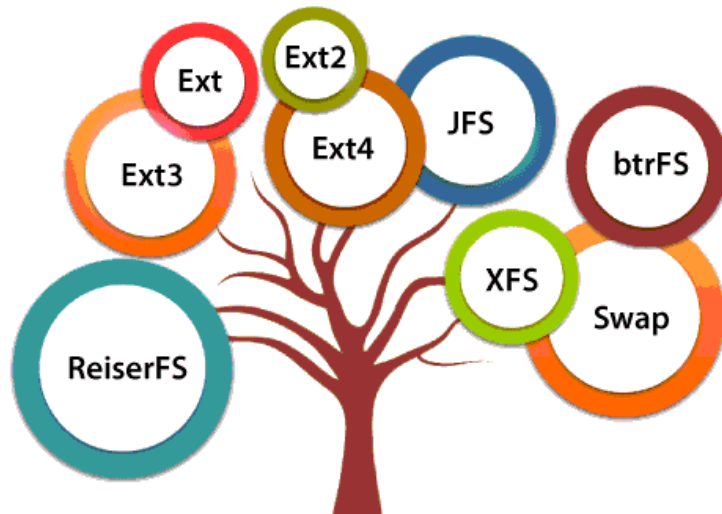
C.d.L. in Informatica
(laurea triennale)

Dip. di Matematica e Informatica
Università degli Studi di Catania

Anno Accademico
2022-2023

Prof. Mario F. Pavone
mpavone@dmi.unict.it

Types of Linux File System



File System e Dischi



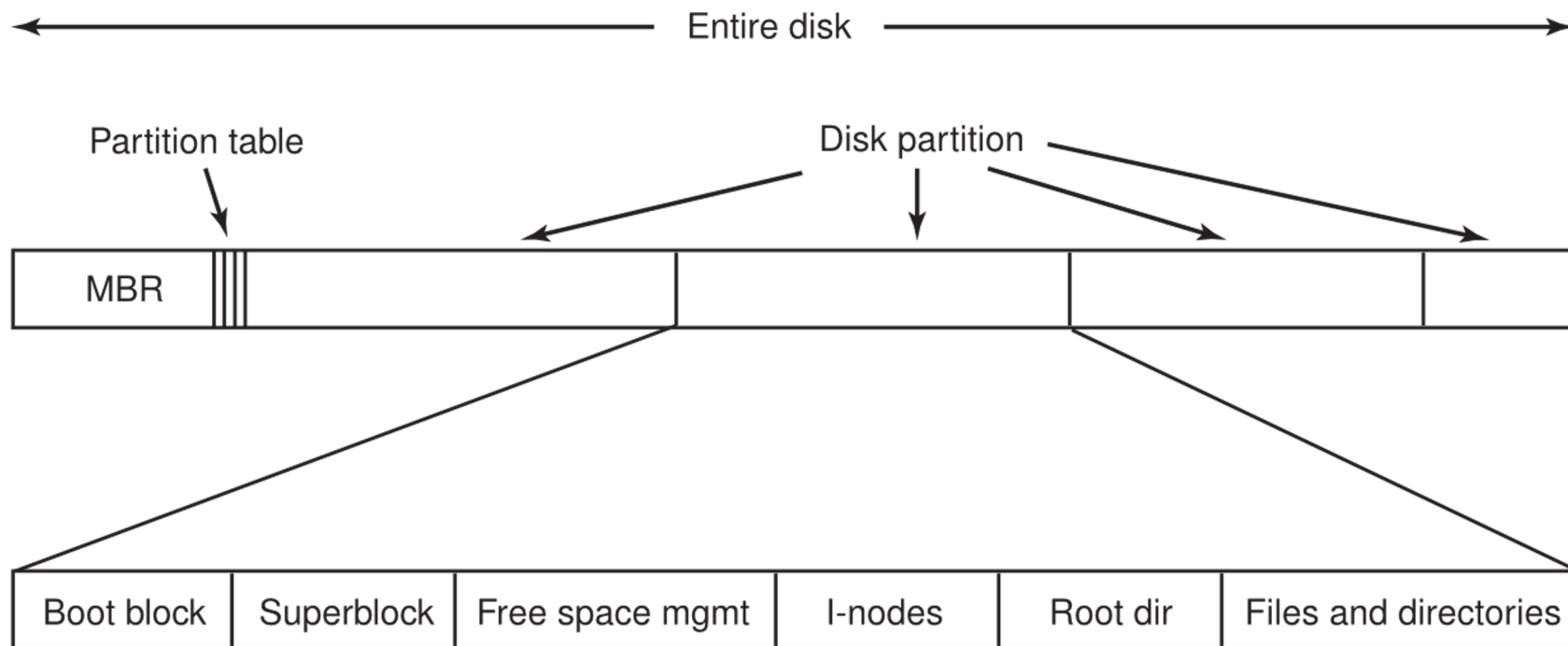
I FILE SYSTEM

- Problema di base: gestire **grandi quantità** di informazioni, in modo **persistente** e condiviso **tra più processi**;
- astrazione: **file** e **directory**;
- i dettagli di gestione ed implementazione costituiscono il **file system**;
- esempi di **dettagli**:
 - nomenclatura;
 - tipi di file;
 - tipi di accesso;
 - metadati (o attributi);
 - operazioni supportate sui file;
 - accesso condiviso ai file: i **lock**:
 - **shared vs. exclusive**;
 - **mandatory vs. advisory (obbligatori vs. consultivi)**;
- **strutture dati** per la gestione dei file: globale e per processo.



STRUTTURA DI UN FILE SYSTEM

- **Master Boot Record (MBR)** => settore 0 del disco;
- partizioni e **boot record** (o boot block);
- **superblocco**;



- layout moderno alternativo: **GPT** (GUID Partition Table) definito dallo standard **EFI**.

