Sistemi Operativi e Reti di Calcolatori (SOReCa)

Corso di Laurea in *Ingegneria Informatica e Automatica (BIAR)*

Terzo Anno | Primo Semestre

A.A. 2024/2025

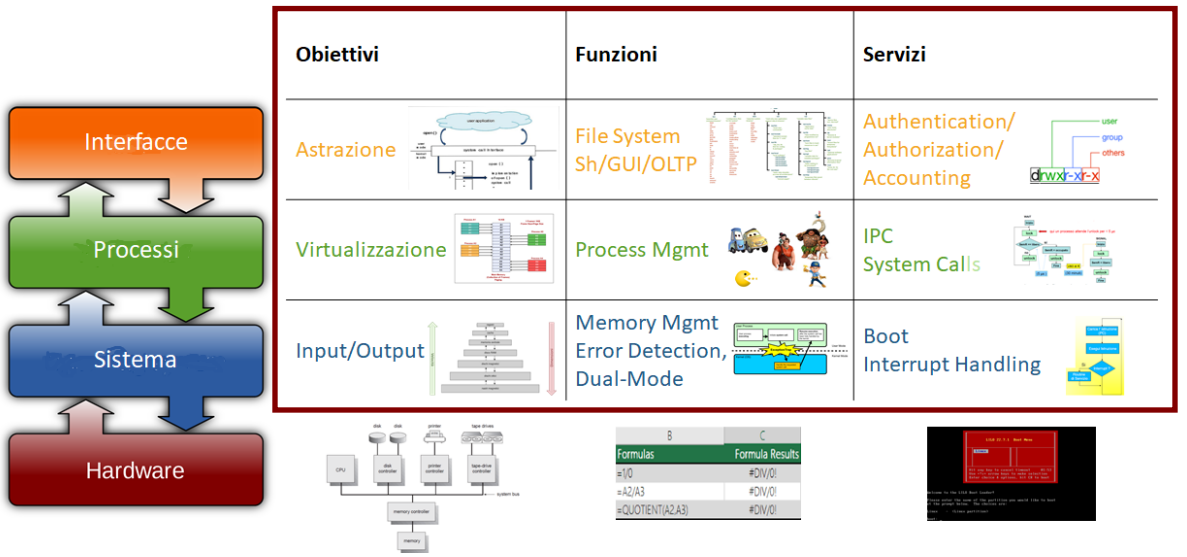
Il Sistema Operativo

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Sistemi operativi (3 CFU)

- Il sistema operativo
- Concorrenza e sincronizzazione
- Deadlock
- Inter-process communication (IPC)
- Scheduling
- Memoria centrale e virtuale
- Memoria di massa e File system
- Sicurezza informatica



Lezioni: Settembre - Ottobre

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Introduzione ai Sistemi Operativi

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Sistemi Operativi

1. Imparare:
   - Principi
   - Funzionamento
   - Struttura

   Con sono stati ideati i Sistemi Operativi.

2. Costruire le fondamenta per l'utilizzo (IT, IoT, OT)

3. … ed anche Capire i S.O. al fine di
   - sceglierli, configurarli, gestirli e usarli
   - poter sviluppare applicazioni sfruttando i development kit
   - poterne scrivere porzioni

Non è un corso di progetto di sistemi operativi.

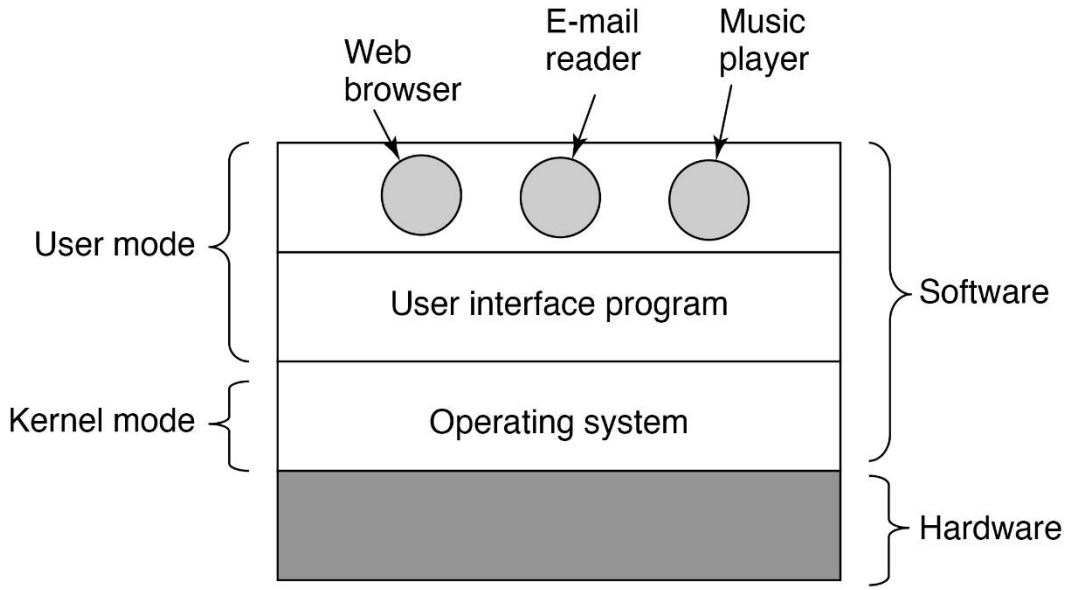
Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Sistemi Operativi: definizioni

- everything a vendor ships when you order “the operating system”

- the operating system is the one program running at all times on the computer—usually called the kernel



Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Hardware

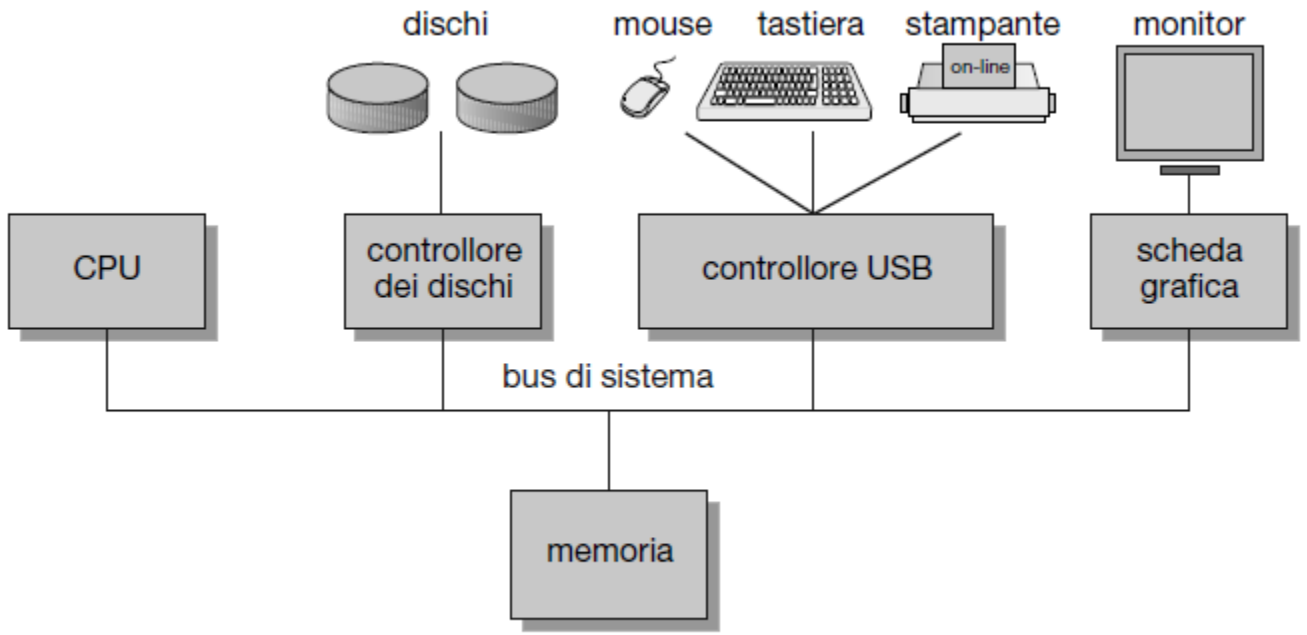
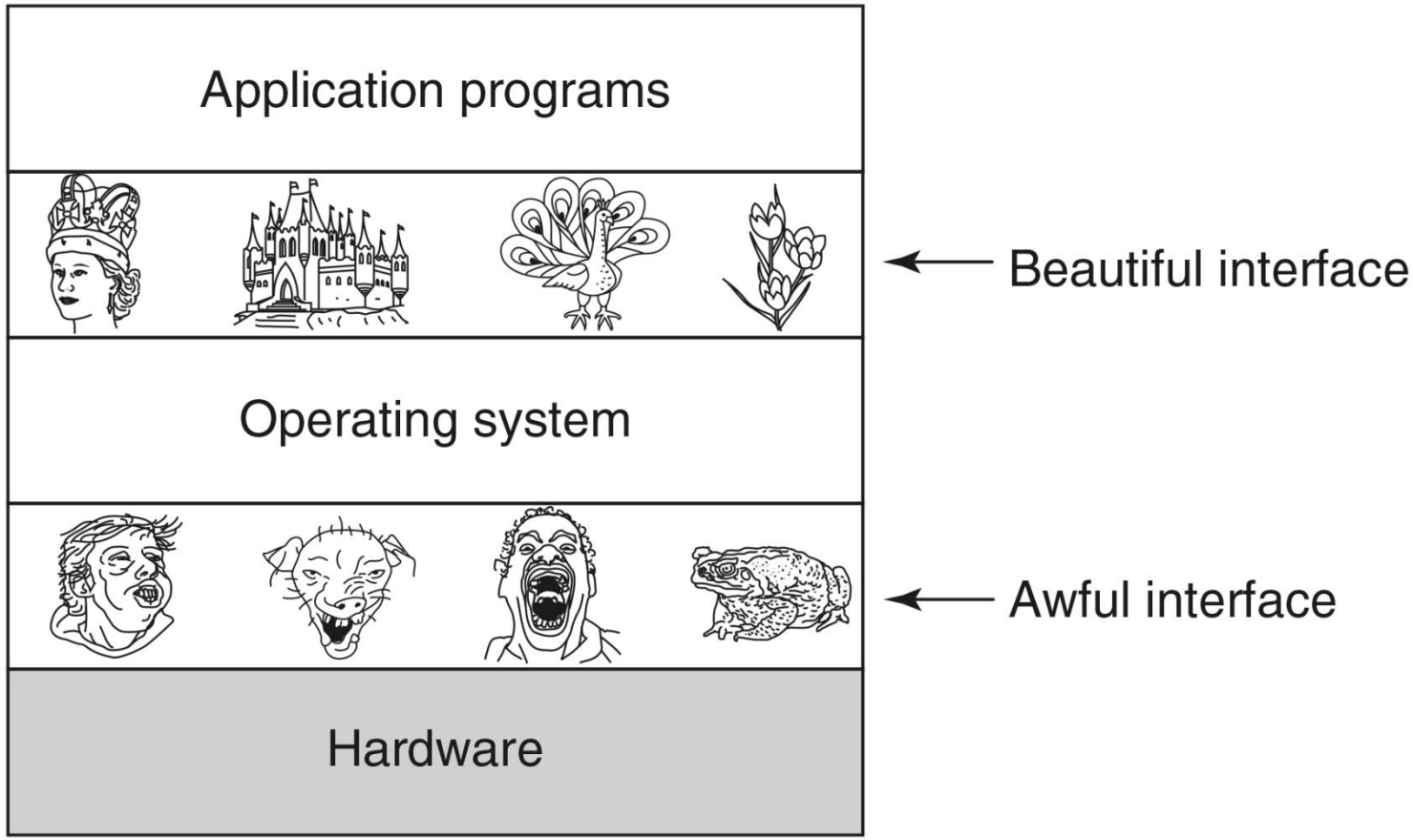


Figura 1.2 Un tipico sistema elaborativo.

Definizione di sistema operativo

1. I **sistemi operativi** esistono poiché rappresentano una soluzione ragionevole al problema di realizzare un sistema elaborativo che si possa impiegare facilmente, per eseguire i programmi e agevolare la soluzione dei problemi degli utenti.
2. Il sistema operativo è il solo programma che funziona sempre nel calcolatore, generalmente chiamato **kernel** (*nucleo*).
3. Oltre al kernel vi sono due tipi di programmi: i **programmi di sistema**, associati al sistema operativo, ma che non fanno necessariamente parte del kernel, e i **programmi applicativi**, che includono tutti i programmi non correlati al funzionamento del sistema.
4. I sistemi operativi mobili non sono costituiti esclusivamente da un kernel, ma anche da un **middleware**, ovvero da una collezione di ambienti software che fornisce servizi aggiuntivi per chi sviluppa applicazioni.

Operating system as resourse manager



Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

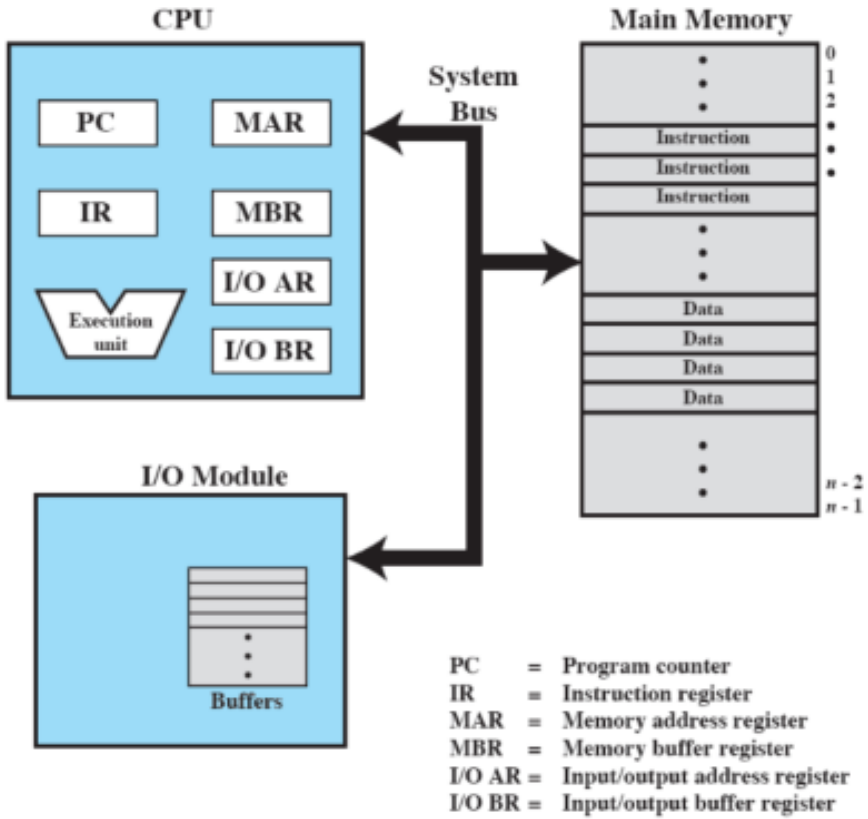
Sapienza
Università di Roma

Task del Sistema operativo

- Gestione dei processi
- Gestione della memoria
- Gestione dei file
- Gestione della memoria di massa
- Gestione della cache
- Gestione dell'I/O

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Funzioni Principali del S.O.



Le funzioni principali del S.O. sono:

- La gestione dell'unità centrale di elaborazione (CPU)
- La gestione della memoria volatile (RAM)
- La gestione della memoria di massa (Hard Disk)
- La gestione dell'I/O (input/output)
- La gestione dei processi
- La gestione dei file (file system)

Ma il S.O. fornisce anche:

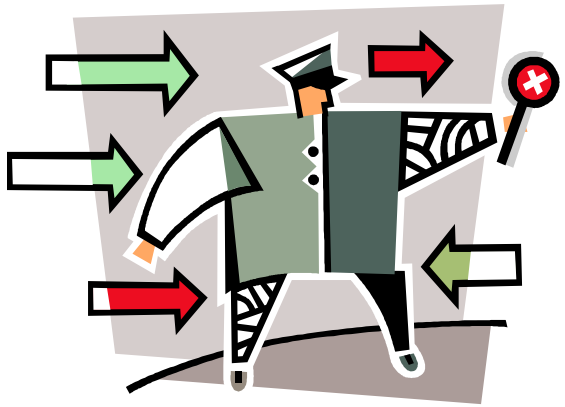
- interfaccia per l'utente (user, admin o developer)
- interprete dei comandi o shell
- gestione della sicurezza
- controllo accesso alle risorse

Ovvero la gestione efficace ed efficiente delle risorse.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating System



- User view: OS provides a set of services to system users

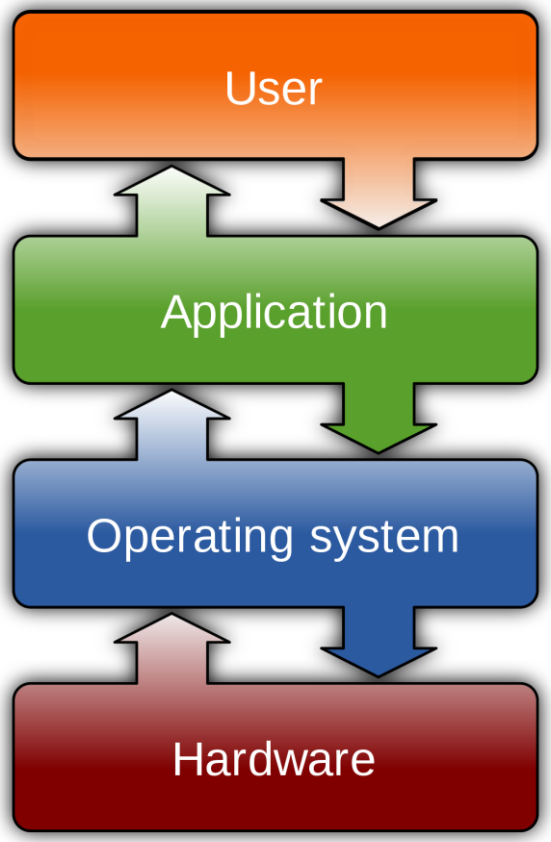
- System view: OS exploits the hardware resources:
  - processors
  - memory
  - I/O devices
  - …

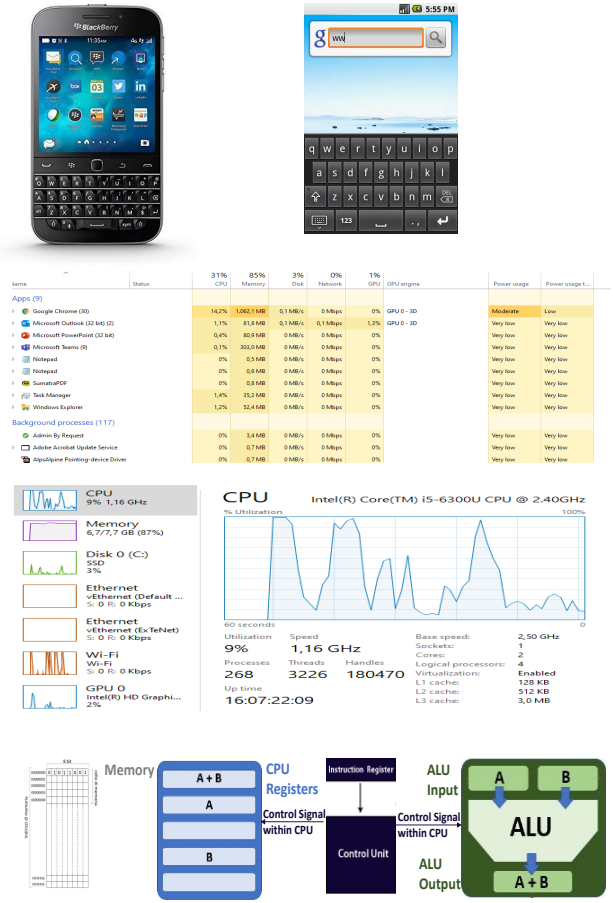
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Intro 5W-1H

1. **OS** (who): the only program (alone too) that is always working on a switched-on computer. But alone is unuseful!
2. **Main Goal** (why): virtualizzazione dell'elaborazione
3. **Main Task** (how): controllo delle applicazioni attive
4. **Operation** (what): computer resource management
5. **Generally Adopted** (where): computer «general purpose» (elaboratori)
6. **Official Birthday** (when): January 1°, 1970 (epoch time)

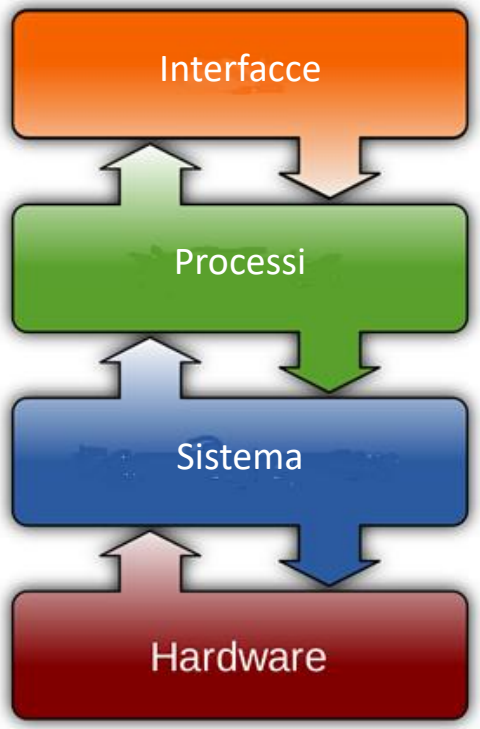


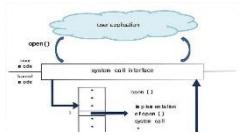
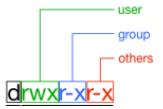
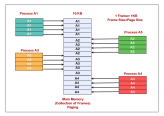
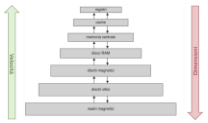
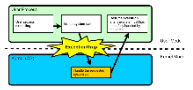


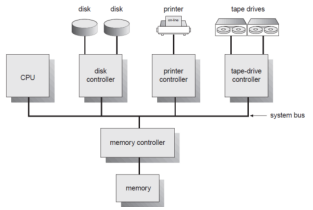
DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

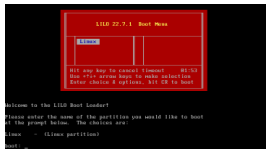
Operating Systems: 3x3 Matrix



Obiettivi	Funzioni	Servizi
Astrazione 	File System Sh/GUI/OLTP 	Authentication/ Authorization/ Accounting 
Virtualizzazione 	Process Mgmt 	IPC System Calls 
Input/Output 	Memory Mgmt Error Detection, Dual-Mode 	Boot Interrupt Handling 



B	C
Formulas	Formula Results
=1/0	#DIV/0!
=A2/A3	#DIV/0!
=QUOTIENT(A2,A3)	#DIV/0!

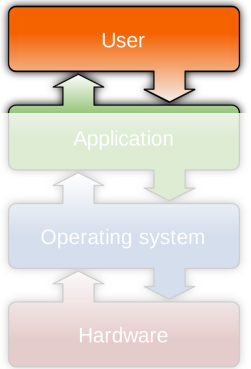


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Intro

Main Goal (why): virtualizzazione dell'elaborazione



Es. Virtualizzazione della Tastiera

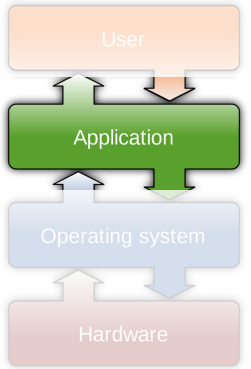
Rendere il sistema quanto più possibile «user-friendly» per gli utenti,.

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Intro

Main Task (how): controllo delle applicazioni attive

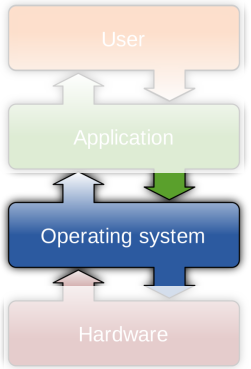


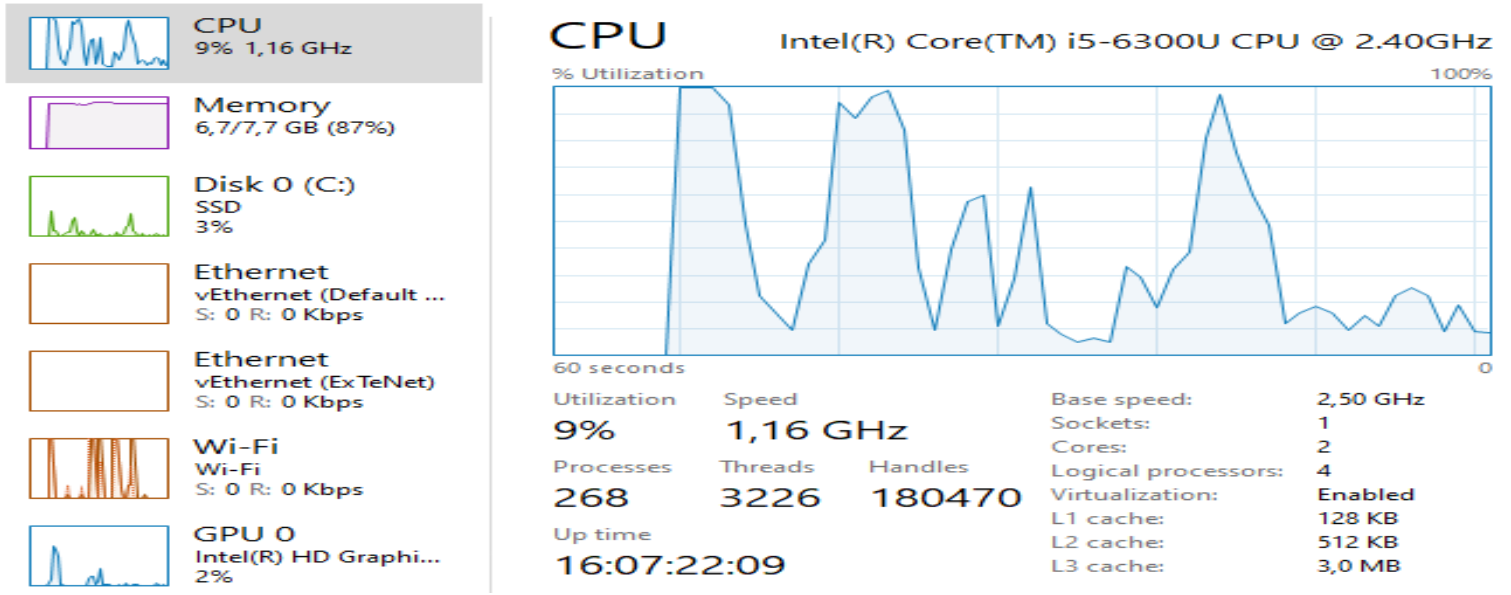
Name	Status	31% CPU	85% Memory	3% Disk	0% Network	1% GPU	GPU engine	Power usage	Power usage t...
Apps (9)									
Google Chrome (30)		14,2%	1.062,1 MB	0,1 MB/s	0 Mbps	0%	GPU 0 - 3D	Moderate	Low
Microsoft Outlook (32 bit) (2)		1,1%	81,8 MB	0,1 MB/s	0,1 Mbps	1,3%	GPU 0 - 3D	Very low	Very low
Microsoft PowerPoint (32 bit)		0,4%	80,9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Microsoft Teams (9)		0,1%	303,0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Notepad		0%	0,5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Notepad		0%	0,6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
SumatraPDF		0%	0,8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Task Manager		1,4%	35,2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Explorer		1,2%	52,4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Background processes (117)									
Admin By Request		0%	3,4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Adobe Acrobat Update Service		0%	0,7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AlpsAlpine Pointing-device Driver		0%	0,7 MB	0 MB/s	0 Mbps	0%		Very low	Very low

Il funzionamento del S.O. necessita della creazione di questa nuova fattispecie: l'applicazione.

Operating Systems: Intro

Operation (what): computer resource management





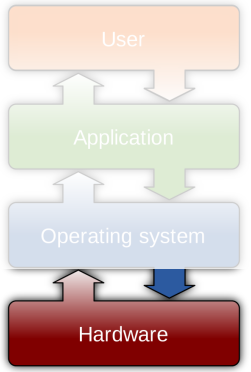
Le risorse del sistema creazione di questa nuova fattispecie: l'applicazione.

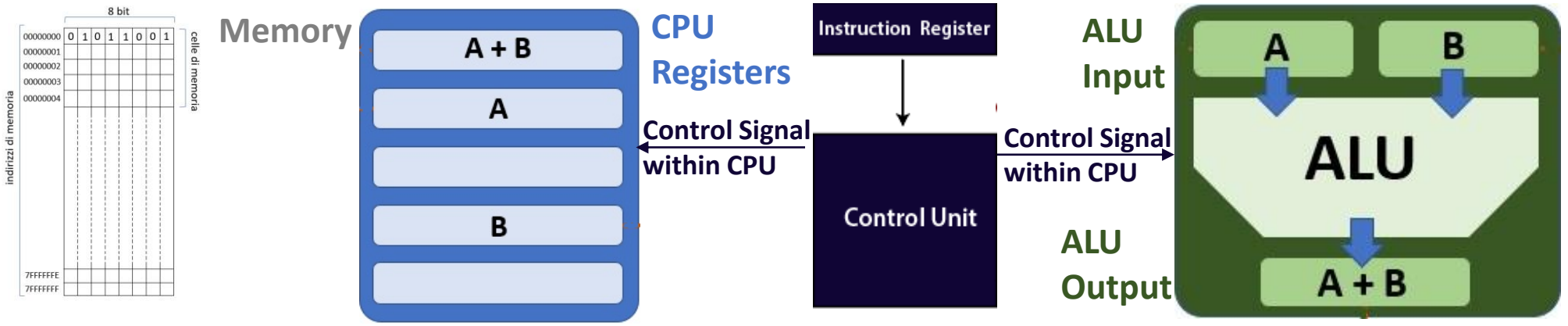
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Intro

Generally Adopted (where): computer «general purpose» (elaboratori)



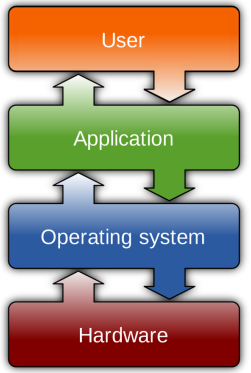


«Sistemi Operativi» è naturale prosecuzione dei «Impianti di Elaborazione».

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Intro

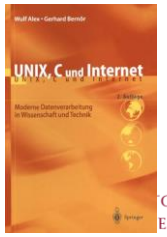


1. **Official Birthday** (when): January 1°, 1970 (epoch time)

L'**epochtime** (o Unix time o POSIX time o Unix timestamp) è il numero di secondi trascorsi dal 1 gennaio 1970 (mezzanotte UTC/GMT), senza contare i secondi bisestili (in ISO 8601: 1970-01-01T00:00:00Z).
Letteralmente parlando l'epoch è il tempo 0 di Unix (mezzanotte 1/1/1970), ma 'epoch' è spesso usato come sinonimo di tempo Unix. Alcuni sistemi memorizzano le date epocali come un intero firmato a 32 bit, il che potrebbe causare problemi il 19 gennaio 2038 (noto come il problema dell'anno 2038 o Y2038). Il convertitore in questa pagina converte i timestamp in secondi (10 cifre), millisecondi (13 cifre) e microsecondi (16 cifre) in date leggibili.

C, Internet e UNIX sono invenzioni intrinsecamente legate fra di loro.

(cfr. Wulf AlexGerhard Bernör, «UNIX, C und Internet, Moderne Datenverarbeitung in Wissenschaft und Technik“,



Storia dei Sistemi Operativi

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: D. Cenni Storici
Evoluzione dei Sistemi Operativi

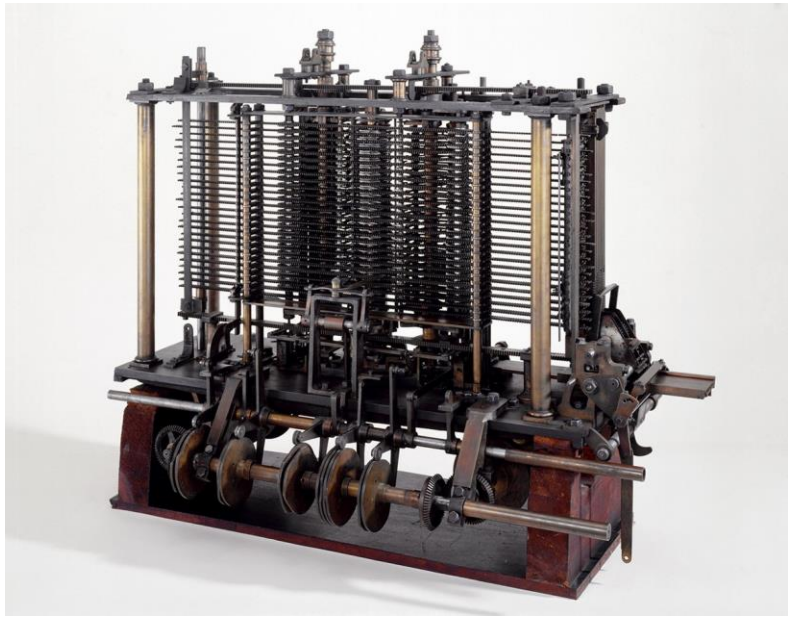
- Prima generazione 1945 – 1955 (computer a valvole)
  - assenza di SO o SO dedicato

- Seconda generazione 1955 – 1965 (transistor)
  - SO batch (a lotti) per sistemi mainframe
- Terza generazione 1965 – 1980 (circuiti integrati)

  - SO in multiprogrammazione
  - SO interattivi (Time-Sharing)
  - SO real time
- Quarta generazione 1980 – ad oggi (VLSI Very Large Scale Integration)
  - SO per personal computer, sistemi palmari, sistemi multi-processore, sistemi distribuiti, multimediali, ecc..

Generazione 0 (~1800)

- Charles Babbage (1792-1871), UK, analitic engine
- Purely mechanical system
- He understood the necessity of the programmer and hired Ada Lovelace (lord Biron’s son)

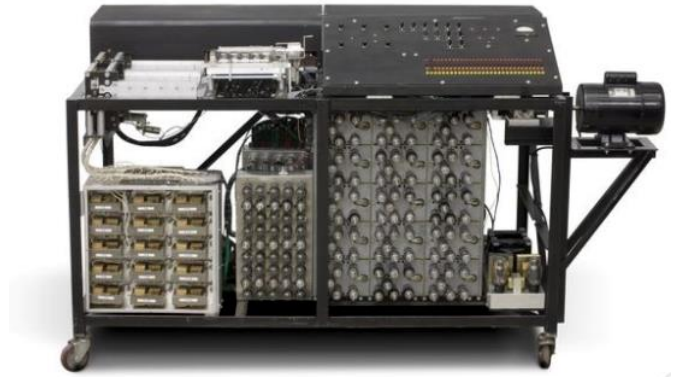


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Prima Generazione (1945-55): Vacuum Valves

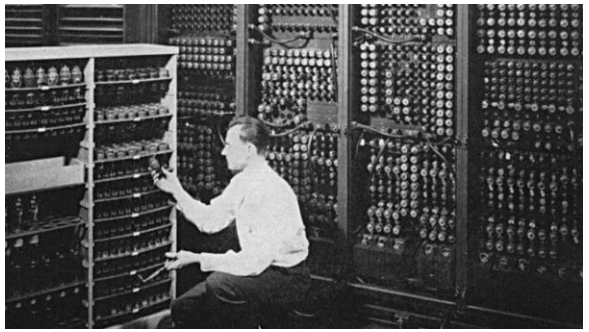
- Atanasoff and Berry, Iowa State University
  - 300 valves



- Conrad Zuse, Berlin:
  - Z3 pc
  - electromechanical relays

- Bletchley park, UK, 1944
  - Colossus

- Aiken, Harvard
  - Mark I

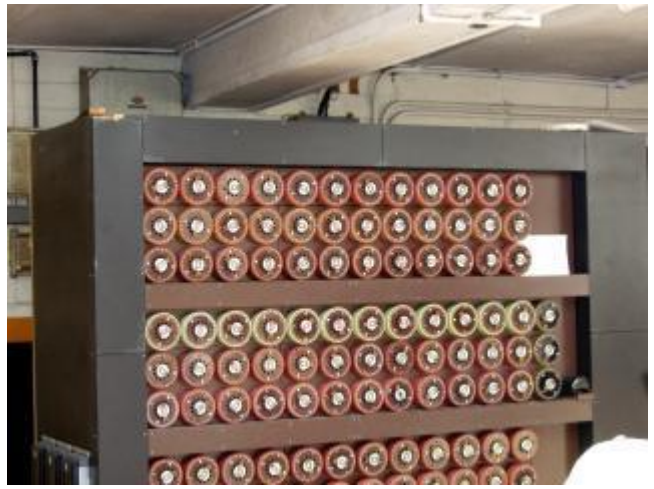


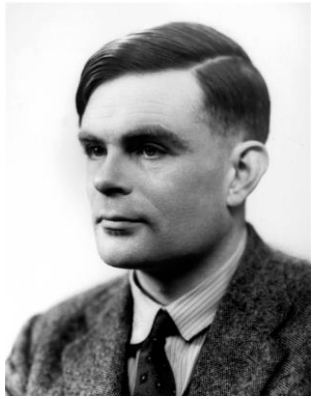
- Mauchley and Eckert, Pensylania
  - ENIAC

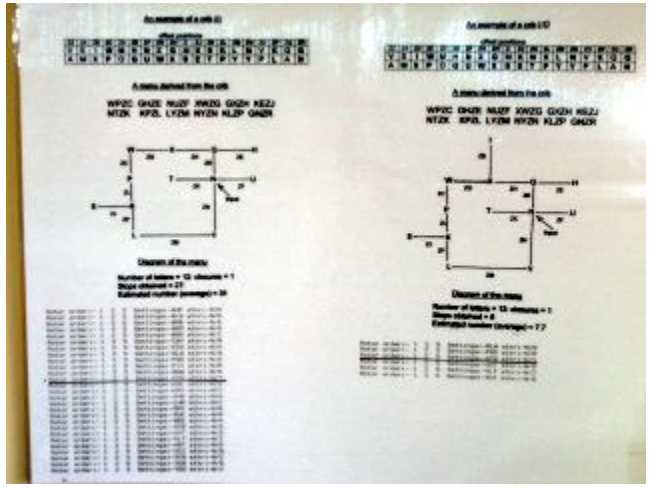
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Prima Generazione (1945-55): Bletchley Park (II Guerra Mondiale)



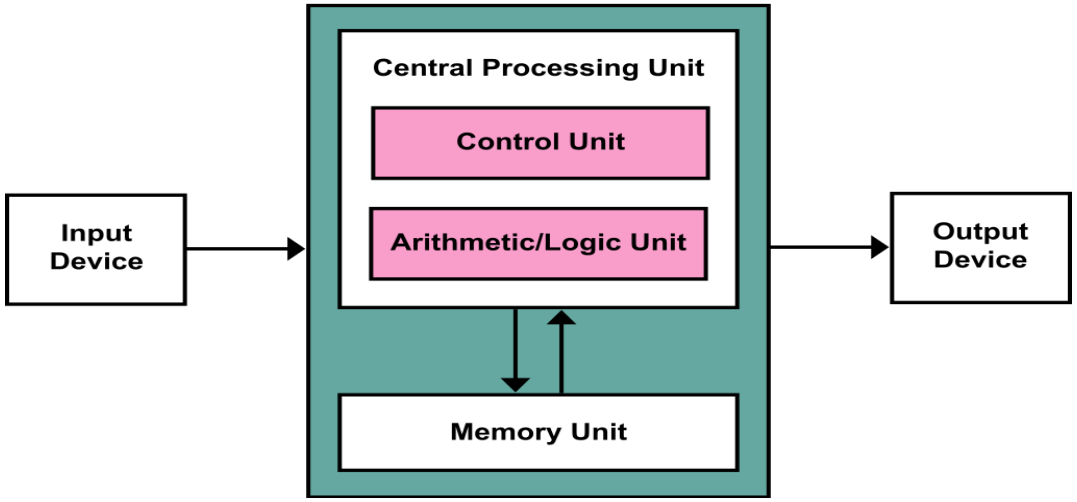


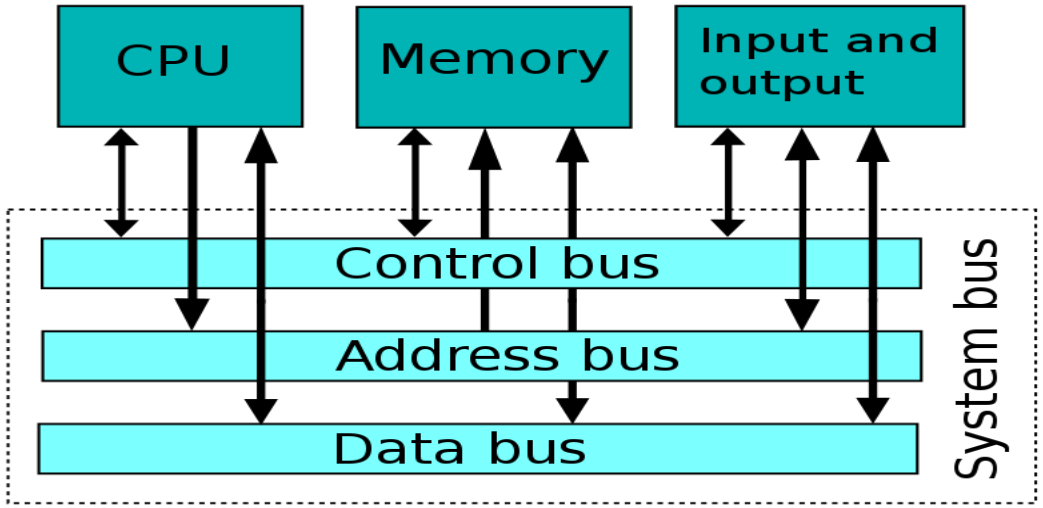


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Prima Generazione (1945-55): Macchina di von Neumann (~1944-1952)





Modello di **architettura** di un **computer**, sviluppato per il Sistema IAS machine (Institute for Advanced Study, Princetown, USA).

- **Stored-program computer**: **dati** ed **istruzioni** risiedono **assieme** in una **memoria comune** (istruzioni trattate alla stregua dei dati)
- **Sequential Instruction**: istruzioni eseguite in modo sequenziale
- **Bus**: memoria collegata all'unità di elaborazione centrale mediante bus

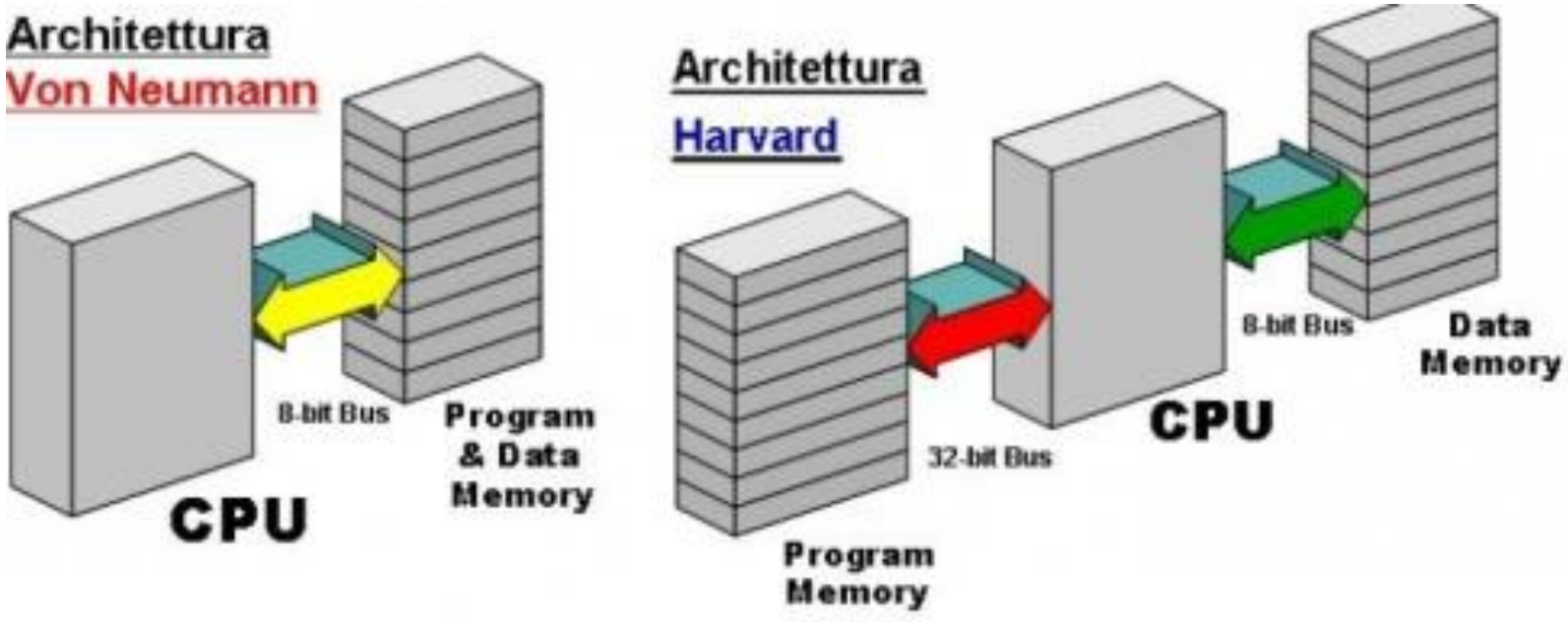
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Prima Generazione (1945-55): Macchina di von Neumann vs Harward

Stored-program computer: proprio la commistione di **dati** ed **istruzioni** ha come conseguenze dirette:

- **Metadata**: enormi benefici nella catalogazione dei dati digitali (rispetto a quelli analogici).
- **Injection**: necessità di operare una distinzione accurate dell'input, mediante validazione.



Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Prima Generazione (1945-55): Valvole

SO inesistente o troppo dedicato

- Le prime macchine da calcolo usavano relè meccanici, ma erano molto lente (tempi di ciclo misurabili in sec.); i relè furono poi sostituiti da valvole termoioniche
- Tutta la **programmazione** (semplici calcoli matematici) veniva effettuata **interamente in linguaggio macchina** (no assembler)
  - predisponendo una serie di cablaggi su schede particolari per controllare le funzioni più elementari della macchina
  - migliorata negli anni 50, con l'introduzione dell'I/O su nastro o schede perforate

- Grossi calcolatori a singolo utente (simultaneamente)
  - il programmatore era anche utente e operatore

Seconda Generazione (1955-65): Transistor 1/3

SO batch (a lotti) per sistemi mainframe

- Programmatore diverso dall'operatore: nascono i ruoli di progettista, costruttore, operatore e programmatore
- Inizialmente: codice FORTRAN su schede perforate, output stampato
- Successivamente:
  - registrazione di più schede su nastro (tramite calcolatore)
  - Sequenzializzazione automatica dei job: il controllo passa automaticamente da un job al successivo
  - Primo rudimentale SO che leggeva da nastro i job, li eseguiva e salvava su nastro gli output
- Riduzione del tempo di setup raggruppando **job simili** (batch) + **operazioni *offline***

Calcolatore **IBM 1401** adatto a leggere/scrivere su schede e nastri

Per eseguire i calcoli, calcolatori più costosi come **IBM 7094**

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

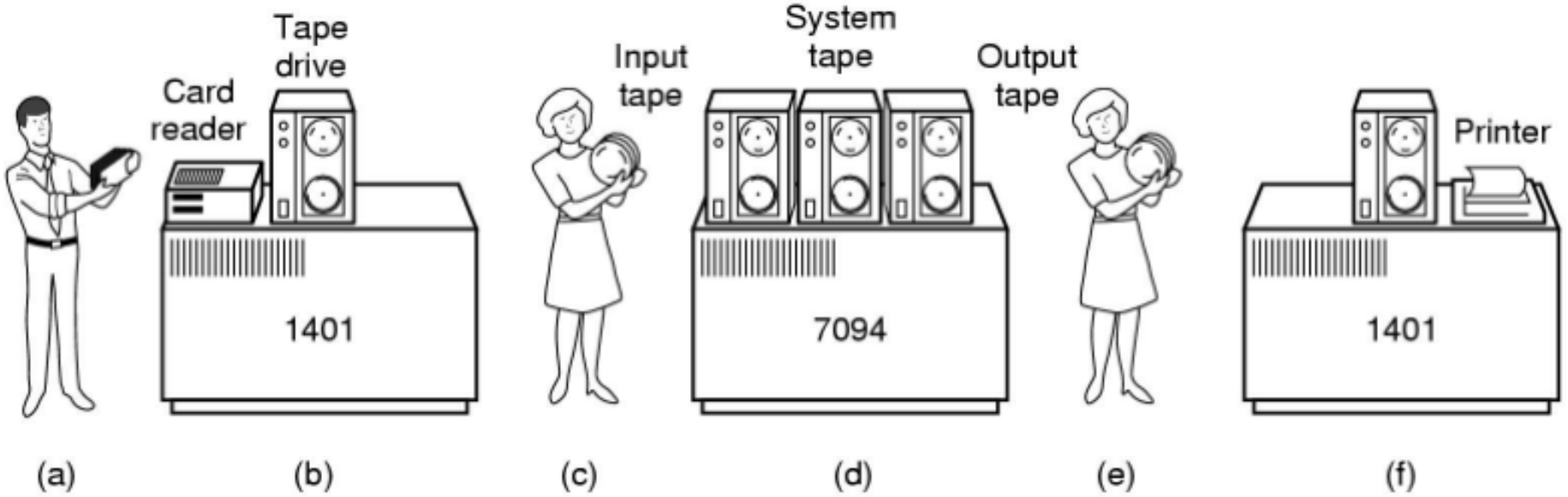
Seconda Generazione (1955-65): Transistor 2/3

- **Un esempio di Sistema Operativo *Batch (a lotti)***

(a,b) le schede relative a un gruppo di programmi vengono lette da un computer specializzato (1401) e trasferite su nastro (*tape*)

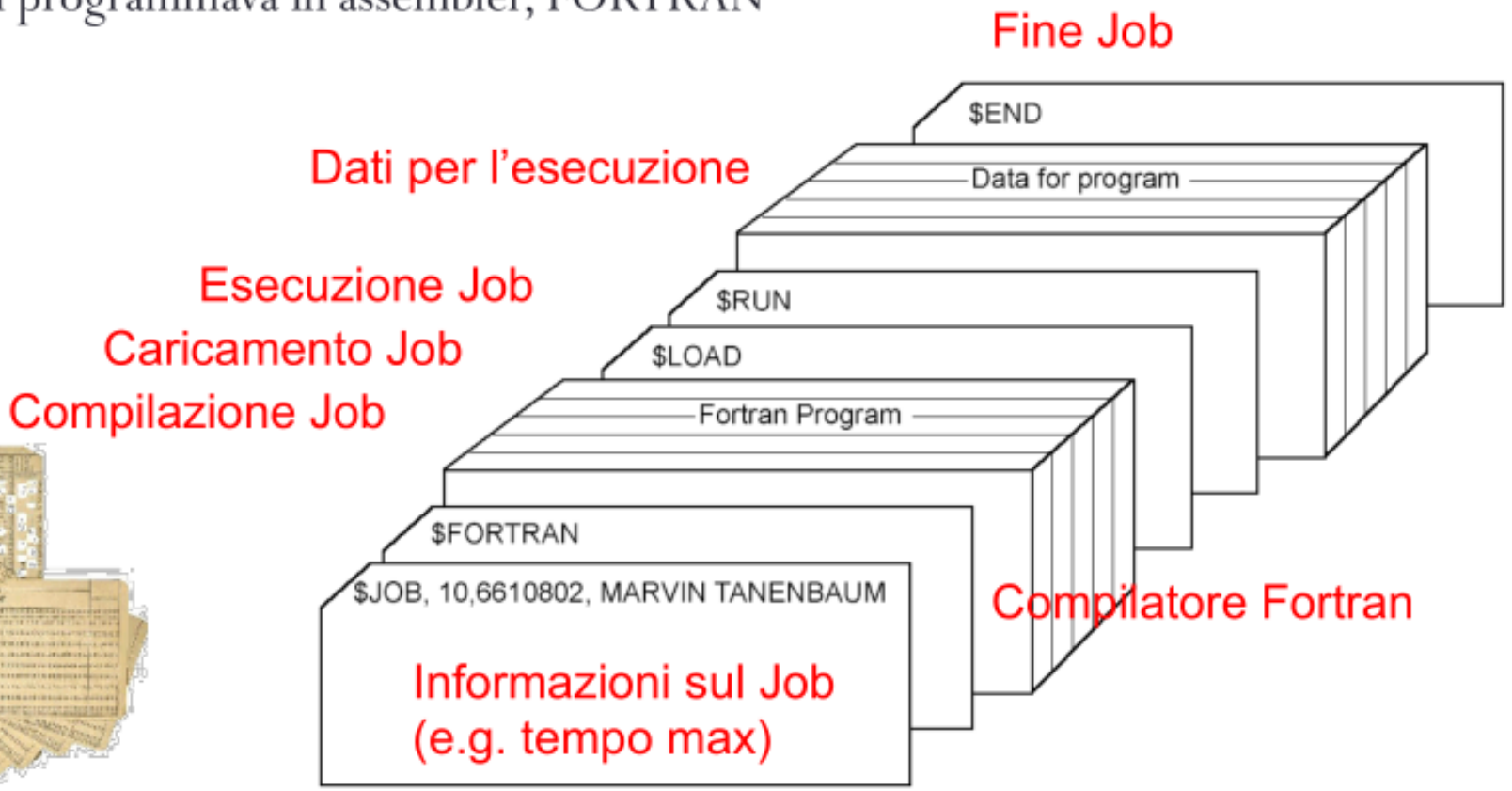
(c,d) il nastro di input viene trasportato su un 7094, che effettua il calcolo e produce un nastro di risultati

(e,f) il nastro dei risultati complessivi viene stampato da un 1401



Seconda Generazione (1955-65): Transistor 3/3

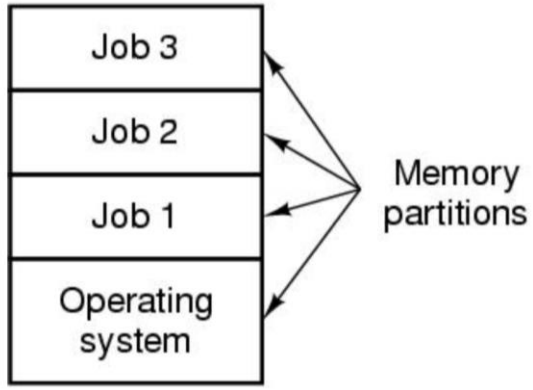
- **Struttura di un tipico *job*** in un sistema operativo batch (FMS – Fortran Monitor System)
  - Si programmava in assembler, FORTRAN



Terza Generazione (1965-80): Circuiti Integrati

Anni 60: Sistemi batch multiprogrammati

- Avvento dei circuiti integrati (migliore rapporto prezzo/prestazioni)
- **Più job sono tenuti in memoria** nello stesso momento
- L'esecuzione dei job deve poter essere interrotta e ripresa in un secondo momento
- **Miglior sfruttamento della CPU** (ad es. nei tempi di attesa di I/O si può allocare la CPU ad un altro job)
- **Maggiori complicazioni nel design del SO**
  - Gestione della Memoria: il sistema deve allocare memoria per più job
  - Scheduling della CPU: il sistema deve scegliere tra più job pronti
  - Allocazione dei dispositivi e routine di I/O fornite dal sistema
    - ad es. **gestione degli interrupt**
- **IBM OS/360** con **spooling** e **multiprogrammazione**



Terza Generazione (1965-80): IBM OS/360

- Goal: generate different topologies of pcs for different users
- Same family
- IBM 360 (still used to manage large database or www)
- Cons: complex OS
  - Thousands of programmers
  - Millions of assembly instructions
  - Thousands of errors

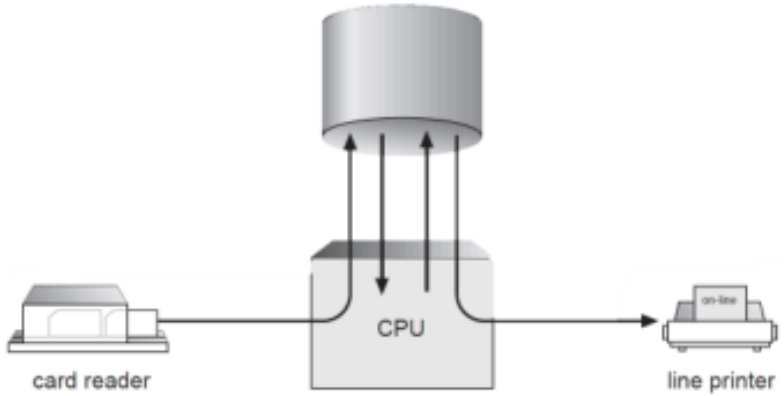
    ➔Silbershatz: SO ~ dinosaurs

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Terza Generazione (1965-80): Spooling

- **Spooling (Simultaneous Peripheral Operation On Line):**
  - simultaneità di I/O e attività di CPU come ulteriore miglioramento dell'efficienza
- Il **disco** viene impiegato come buffer molto ampio, dove:
  - leggere in anticipo i dati (1401 non più necessario)
  - memorizzare temporaneamente i risultati (in attesa che il dispositivo di output sia pronto)
  - caricare codice e dati del job successivo
    - possibilità di sovrapporre I/O di un job con elaborazione di un altro job



Terza Generazione (1965-80): Multitasking, Timesharing, Swapping

Anni 70: Sistemi Time-Sharing – Computazione Interattiva

- I sistemi batch avevano il difetto di allungare i tempi di risposta (tempo tra inserimento lista di job e output ultimo job)

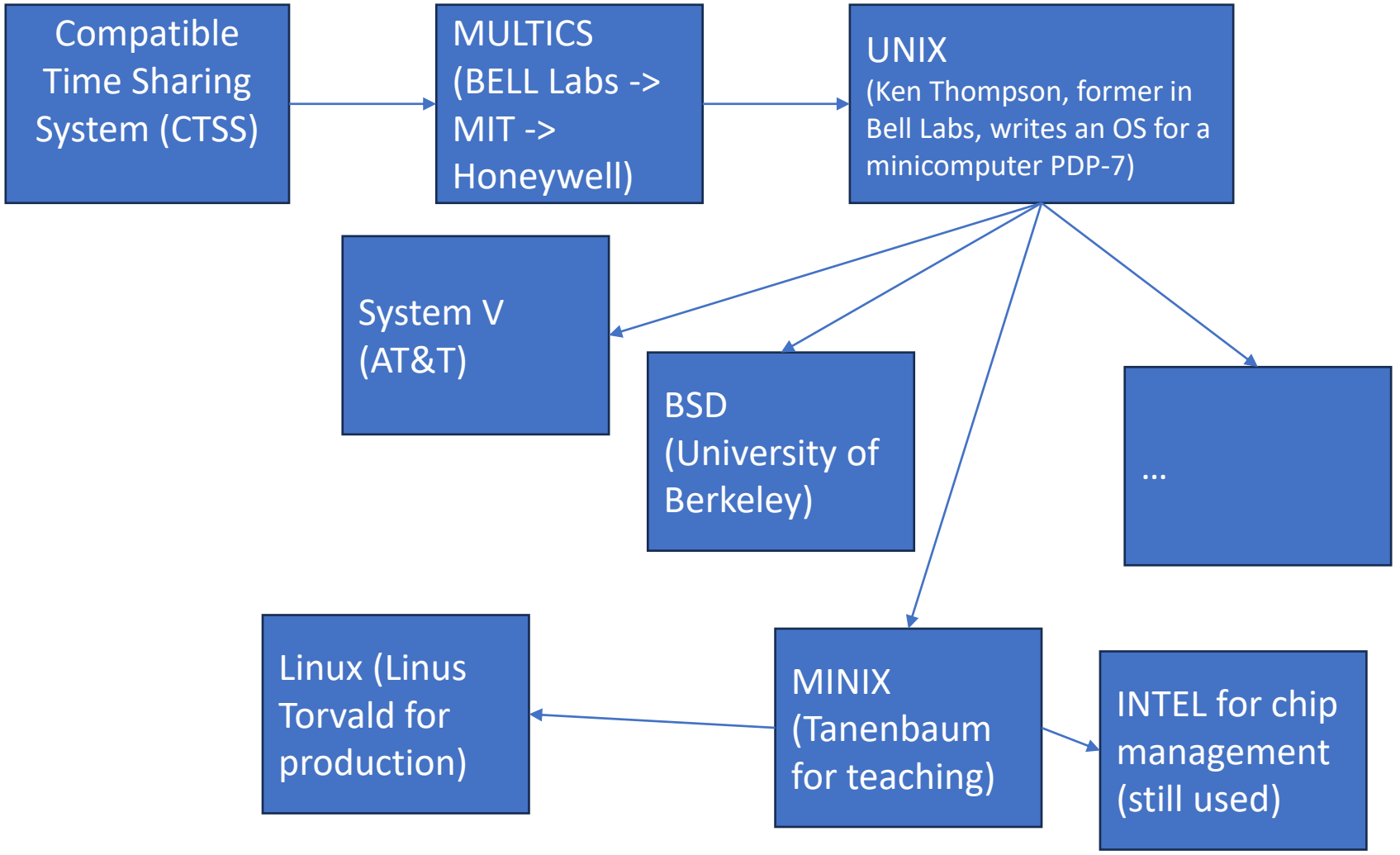
- La CPU è condivisa tra **più job tenuti in memoria e su disco**
- Un job viene caricato dal disco alla memoria, e viceversa (swapping)
- Timesharing: multiprogrammazione + comunicazione on-line tra utente e sistema
  - quando il SO termina l'esecuzione di un comando, attende il prossimo "statement di controllo" non dal lettore di schede bensì dalla tastiera dell'utente (è possibile valutare se continuare o fermare la schedulazione di jobs)

Terza Generazione (1965-80): Esempi

- **MULTICS** (MULTIplexed Information and Computing Service) by MIT, Bell Labs e General Electric
  - Idea iniziale: una macchina molto grande con capacità di calcolo per tutti gli abitanti di Boston, basato sull'idea del sistema elettrico (plug-in)
  - Poco successo commerciale, grande influenza sui sistemi successivi
  - Implementa servizio **centralizzato** e **time-sharing**

- **UNIX:** Versione singolo utente di MULTICS per PDP-7
  - PDP-1 . . . -11: minicalcolatori a 18bit
  - Codice open: molte aziende lo personalizzarono sviluppando sistemi Unix-like
    - Due versioni principali: **SystemV** by AT&T Inc., e **BSD** (Berkeley Software Distribution)
  - **MINIX**: clone UNIX per scopi didattici by A. S. Tanenbaum

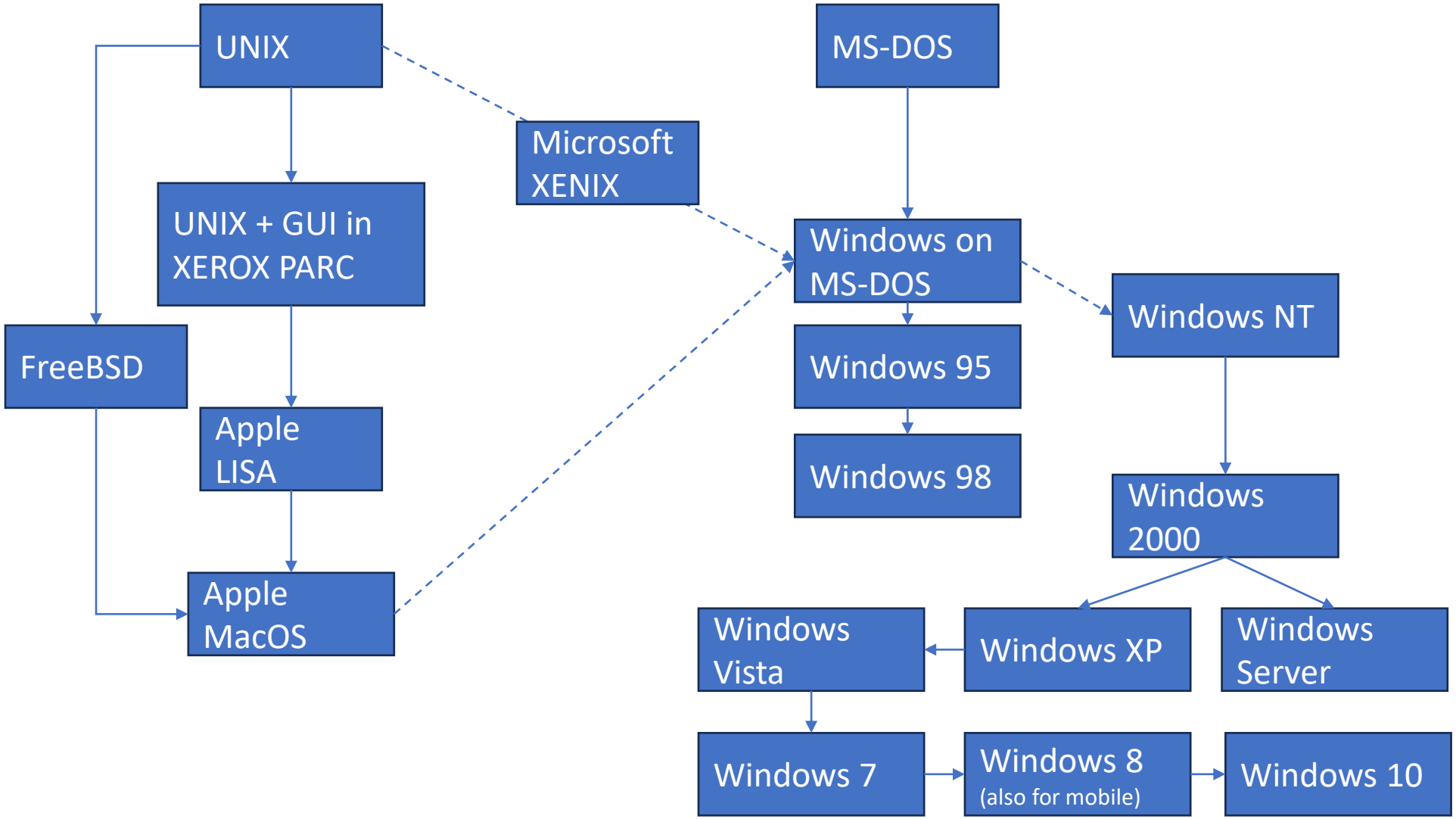
Terza Generazione (1965-80): Genesi di UniX



Quarta Generazione (1980 - 1995): Personal Computer

- Large scale Integration chipset (Intel)
- Floppy disk (Intel hired Kildall to develop Control Program for Microcomputers)
- Kildall creates Digital Research based on CP/M, dominating the OS market
- IBM creates a PC and contacted Bill Gates for his BASIC interpreter
  - Then asked him which OS to use
    1. Digital Research
  - Then asked him an OS
    - BG bought DOS (Disk Operating System) from Seattle Computer Products (75K$)
    - BG hired DOS creator Tim Paterson
    - MS-DOS was a success (marketing strategy: sell MS-DOS with the PC)
    - Kildall died (July 1994)

Quarta Generazione (1980 - 1995): Personal OS



Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Quarta Generazione (1980 - 1995): Network OS

- **Anni 90: SO di rete**
  - distribuzione della computazione tra più processori in rete
  - ma l'utente **ha** coscienza della differenza tra i singoli nodi
  - modello *client/server*

- **Il presente/futuro:**
  - **Sistemi distribuiti** (l'utente ha una visione unitaria del sistema di calcolo)
    - Condivisione delle risorse, tolleranza ai guasti, aumento delle prestazioni
    - Esempi di servizi di rete/protocolli: NFS, *reti P2P* e loro applicazioni (es. per il file sharing come Emule, BitTorrent, ecc..), *Cloud computing* (infrastruttura di calcolo e risorse distribuite e virtualizzate)
  - **Sistemi embedded**

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

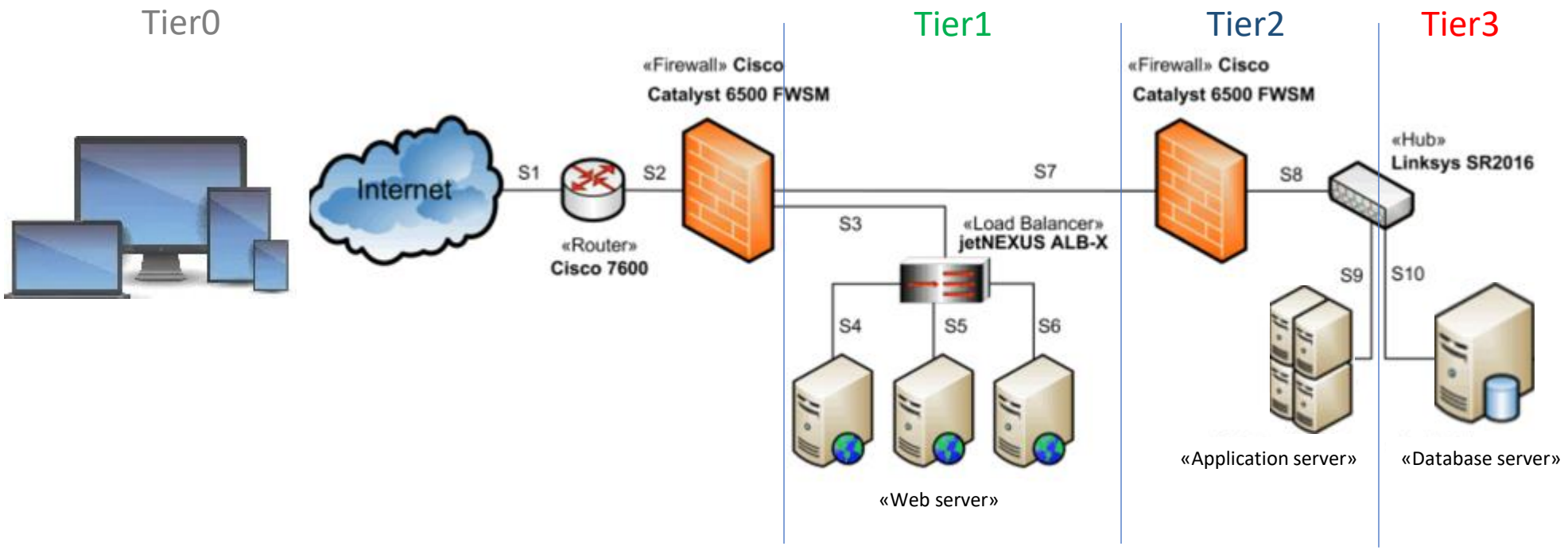
Quarta Generazione (1980-): esempi di “personal” OS

- **CP/M** (Control Program for Microcomputer) basato su disco della Digital Research fondata da Kildall
  - Su PC-IBM con Zilog Z80, o Intel 8080/85 e 80286
- **MicroSoft**:
  - MS-DOS (Disk Operating System) e poi Windows 3.1 (microprocessore a 16 bit)
  - Windows 95 e Windows 98 (ancora con codice assembly a 16bit ma per microprocessorei a 32 bit (Intel 80386, 80486, ecc..)
  - NT e Windows 2000 (a 32bit)
  - Me (update di Windows 98)
  - XP , Vista, Win7, Win8
- **IBM OS/2** (per microprocessori a 32 bit, richiedeva parecchia RAM/risorse)
- **Linux**: versione professionale di MINIX by Linus Torvalds
  - Disribuzioni Linux: Debian, Fedora, Gentoo, Ubuntu, ecc..
  - Open-source (ma lo era anche Unix)
- **Mac OS** di Apple con GUI (Graphical User Interface) ad icone e mouse
- Svariate versioni di sistemi Unix-like come **Sun Solaris**

Fifth generation (1995 -): mobile computing

- Idea from ‘70
- Nokia N9000 first smartphone (term forged by Ericsson in 1997)
- Symbian OS (Samsung, Sony, Ericsson, Motorola, Nokia)
- Blackberry OS (2002) from RIM
- Apple iOS for iPhone (2007)
- In 2011 Nokia moved from SymbianOS to Windows Phone
- Android (first version 2008) from Google quickly removed any competitor except for iOS

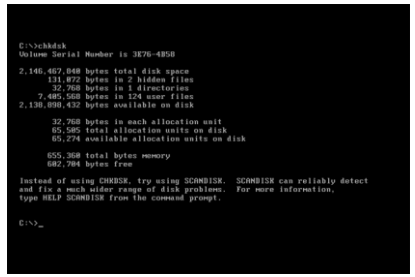
Fifth generation (1995 -): Network computing

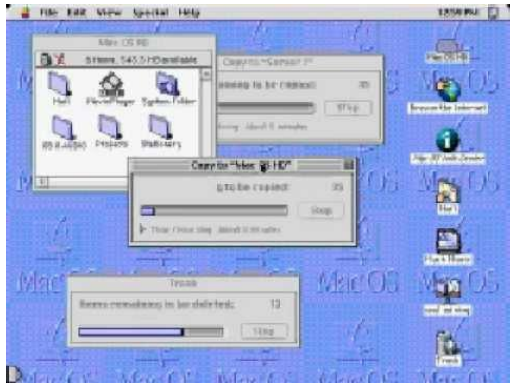


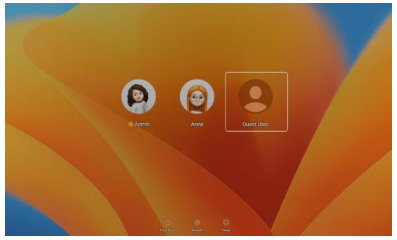
- Gli elementi visualizzati come **«Server»** nel Tier0, Tier1, Tier2 e Tier3 sono **endpoint** ➔ computer come **punto finale** (estremo) di **comunicazioni**, realizzando una specifica funzione applicativa, possibile, grazie al Sistema Operativo sottostante.
- Gli elementi identificati come **«Router»** o **«Firewall» consentono** l'instaurarsi delle **comunicazioni**, eseguendo funzioni di instradamento e **filtro**, al fine di **separare** un **tier** dall'altro, sfruttando le primitive del Sistema Operativo sottostante.
- I «**Server**» sono **connessi** fra loro, in modo:
  - Parallelo (**cluster**) all'interno dello stesso Tier (es. Tier1: «Web Server», Tier2 «App Server»), al fine di garantire l'erogazione del servizio anche se un server dovesse rompersi..
  - Seriale (**gerarchico**) fra i diversi Tier (es. «Web server» verso «App server»), al fine di **difendere** i **dati** (**Tier3**) e l'accesso alle **funzioni** applicative (**Tier2**)

Operating Systems: Prima Classificazione 1/2

- Una prima classificazione basata sui criteri:
  - **Interfaccia testuale**
    - *Interprete di comandi o shell*
  - a **interfaccia grafica** (GUI Graphical User Interface)
    - *Metafora del desktop*
  - **Multitasking**
    - gestire più attività contemporaneamente
  - **Multiutente**
    - far lavorare più utenti contemporaneamente







Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Prima Classificazione 2/2

DOS	Interfaccia testuale	Monotasking monoutente	Microsoft
Windows	Interfaccia grafica	Multitasking monoutente o multiutente a seconda delle versioni	Microsoft
Unix	Interfaccia testuale	Multitasking multiutente	Bell Laboratories
Linux	Interfaccia testuale e/o grafica	Multitasking multiutente	Derivato da Unix **Open source**
OS 2	Interfaccia grafica	Multitasking multiutente	IBM
Mac OS	Interfaccia grafica	Multitasking Monoutente o multiutente	Apple

Tipologie di Sistemi di Elaborazione

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Tipologie di Sistemi di Elaborazione

- Sistemi monoprocessore
- Mainframe
- Personal computer
- Sistema multiprocessore
- Sistema distribuito
- Cluster
- Cloud Computing
- Mainframe
- Computer palmare/Smartphone
- Sistema multimediale
- Sistema di elaborazione in tempo reale
- Sistema dedicato (embedded system, IoT, OT)

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Traditional View

- Traditionally, the computer has been viewed as a sequential machine
  - A processor executes instructions one at a time in sequence
  - Each instruction is a sequence of operations
- Some popular approaches to parallelism
  - Symmetric MultiProcessors (SMPs)
  - Clusters

Operating Systems: E. Tipologie di Sistemi di Elaborazione

Sistemi Monoprocessore

Dispongono di un'unica CPU centrale

- Esegue istruzioni di natura generale

È affiancata da una serie di processor secondari

- Svolgono compiti particolari
- Eseguono un insieme ristretto di istruzioni
- Non eseguono processi utente
- E.g. controllore disco, tastiera, BIOS, etc.

Sistemi Desktop Odierni

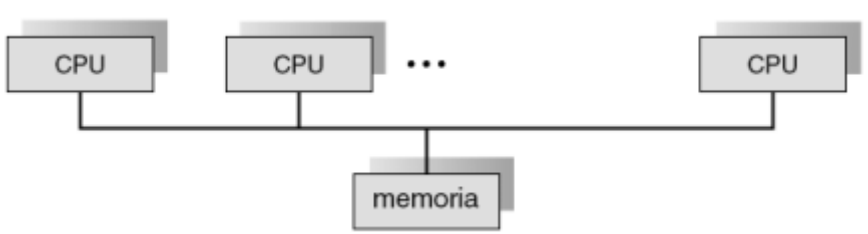
- interazione evoluta con sistemi centrali
- piccole attività di elaborazione locale
- Sistemi desktop con grafica e dispositivi per interazione avanzata
- Sistemi interattivi multiprocessor
- ripartizione memoria tra processi (multiprogrammazione)
- condivisione CPU (multitasking)
- gestione CPU in condivisione di tempo (time sharing)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Sistemi Multiprocessore 1/2

- Sistemi con **più processori** in stretta comunicazione tra loro
  - Conosciuti anche come ***sistemi multiprocessore***
- *Sistemi con processori strettamente connessi* – i processori condividono la memoria, i bus e l'orologio; la comunicazione di solito passa attravers la memoria condivisa
- Vantaggi dei sistemi paralleli:
  - Maggiore quantità di elaborazione effettuata (n unità != velocità * n)
  - Economia di scala sulle periferiche
  - Aumento di affidabilità
    - Graceful degradation: degradazione progressiva (proporzionale al numero di guasti), oppure
    - Fault tolerant: sistemi tolleranti ai guasti (necessitano riconoscimento, diagnosi e eventuale riparazione)



Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Sistemi Multiprocessore 2/2

- **Sistema multiprocessore asimettrico**
  - Ogni processore è assegnato ad uno specifico lavoro; il processore principale (master) organizza e gestisce il lavoro per i processori slave
  - Organizzazione gerarchica dei processori
  - Più comune nei sistemi molto grandi
- **Sistema multiprocessore simmetrico (SMP)**
  - Ogni processore può eseguire tutte le operazioni
  - Organizzazione non gerarchica dei processori
  - Possono essere eseguiti contemporaneamente molti processi senza che si produca un deterioramento delle prestazioni
  - Necessario il bilanciamento
  - Gran parte dei moderni sistemi forniscono supporto SMP

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Typical SMP organization

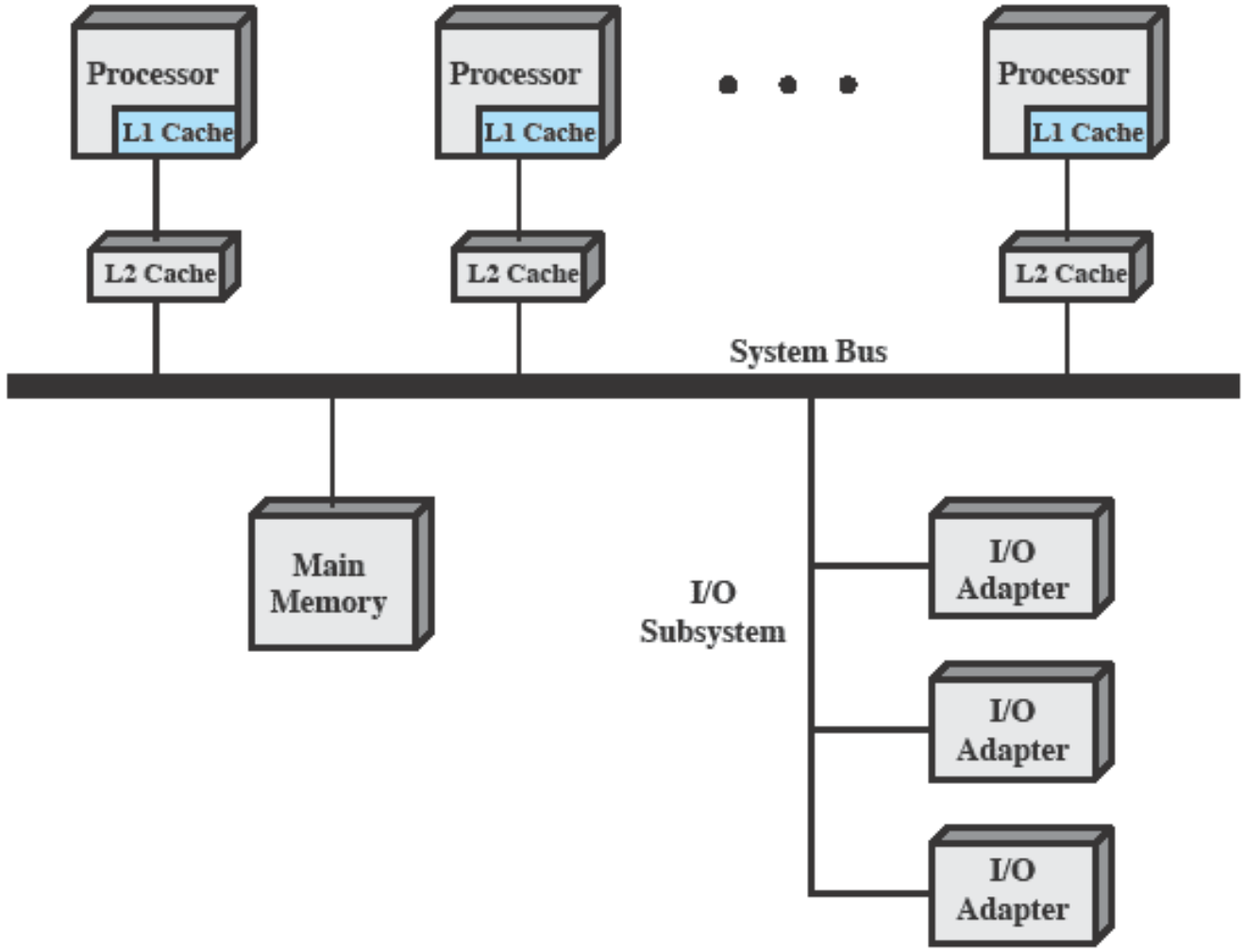


Figure 4.9 Symmetric Multiprocessor Organization

Multiprocessor OS Design Considerations

- The key design issues include
  - Simultaneous concurrent processes or threads
  - Scheduling
  - Synchronization
  - Memory Management
  - Reliability and Fault Tolerance

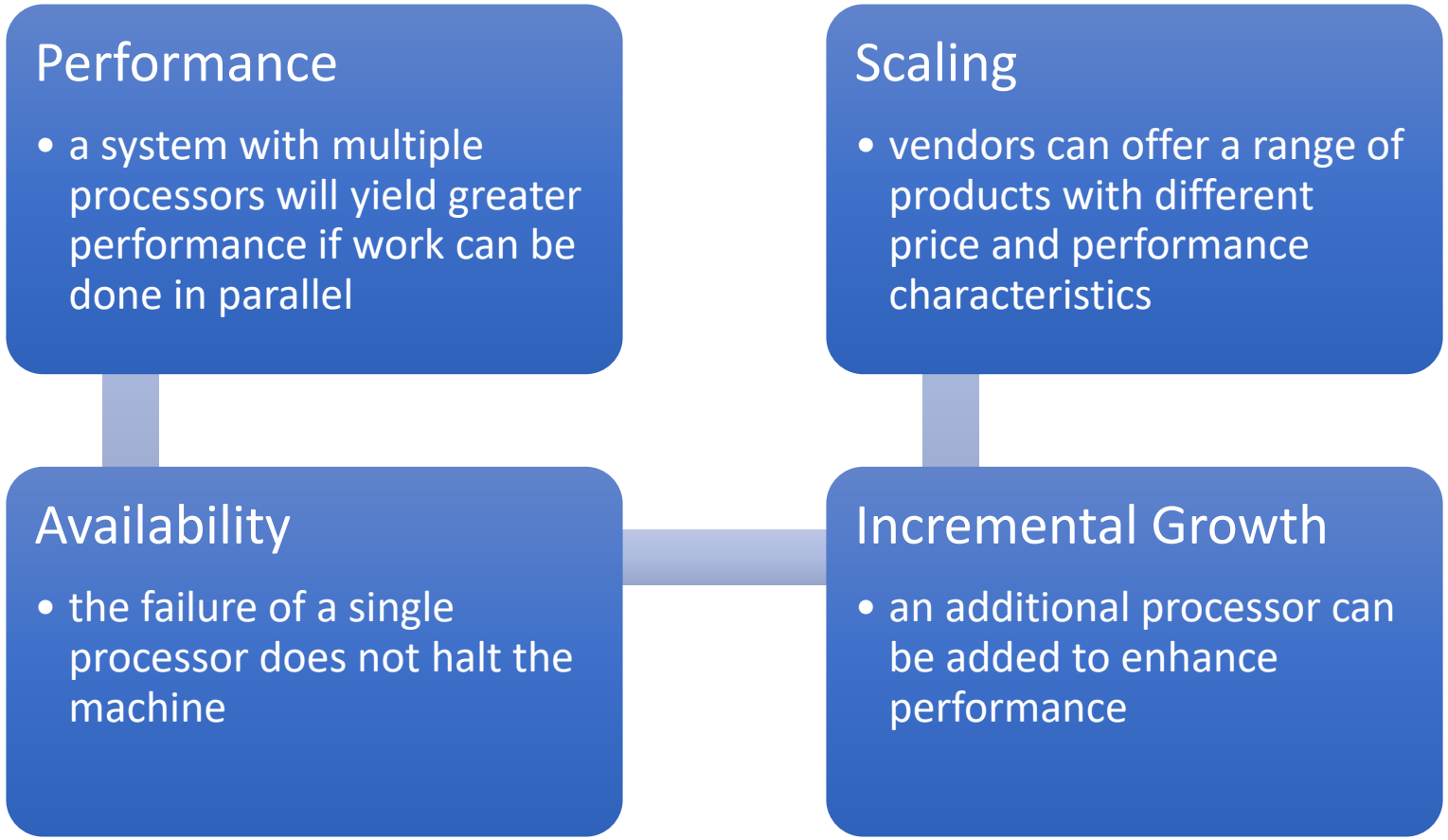
Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Symmetric Multiprocessors (SMP)

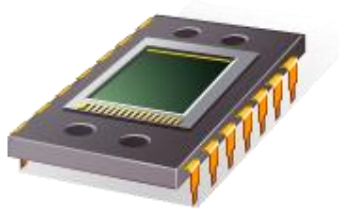
- A stand-alone computer system with the following characteristics:
  - two or more similar processors of comparable capability
  - processors share the same main memory and are interconnected by a bus or other internal connection scheme
  - processors share access to I/O devices
  - all processors can perform the same functions
  - the system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels

SMP Advantages

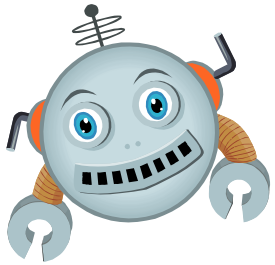


Multicore Computer

- Also known as a chip multiprocessor
- Combines two or more processors (cores) on a single piece of silicon (die)
  - each core consists of all of the components of an independent processor
- In addition, multicore chips also include L2 cache and in some cases L3 cache



Intel Core i7



Supports two forms of external communications to other chips:

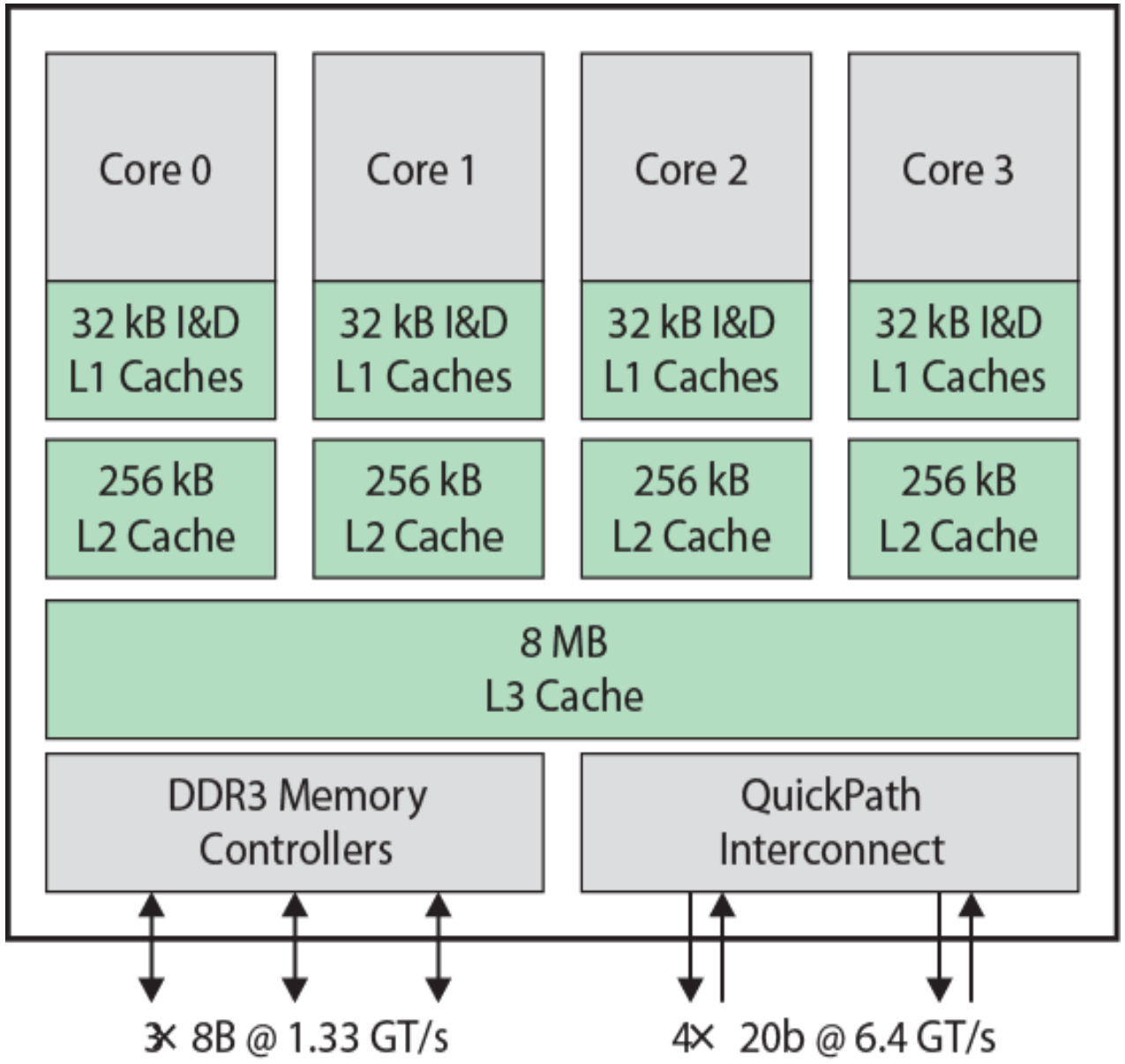
DDR3 Memory Controller

- brings the memory controller for the DDR (double data rate) main memory onto the chip
- with the memory controller on the chip the Front Side Bus is eliminated

QuickPath Interconnect (QPI)

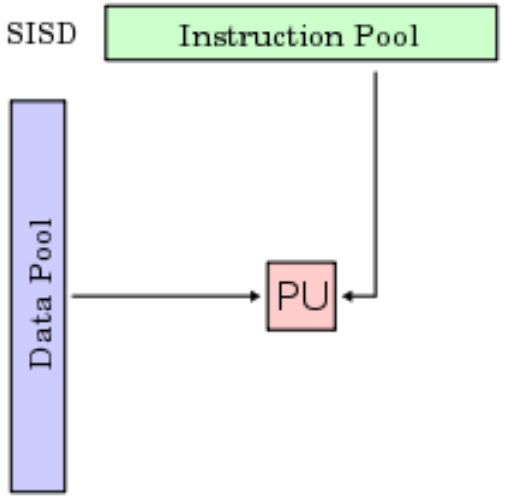
- enables high-speed communications among connected processor chips

Intel Core i7

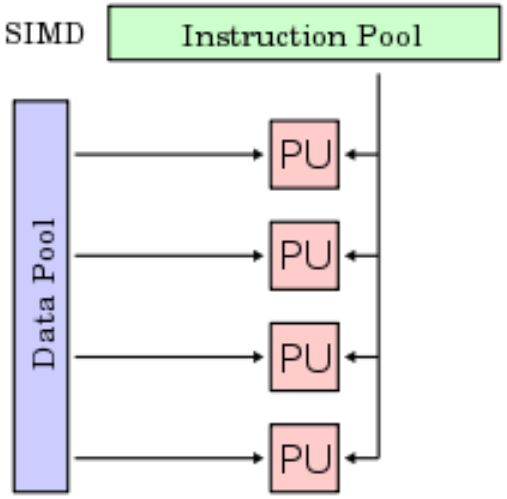


Intel Corei7 Block Diagram

Categories of Computer Systems (Flynn's Taxonomy)



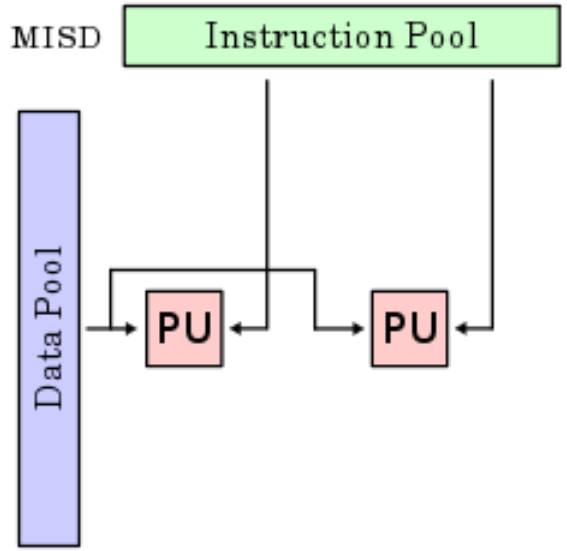
- Single Instruction Single Data (SISD)
  - Single processor executes a single instruction stream to operate on data stored in a single memory

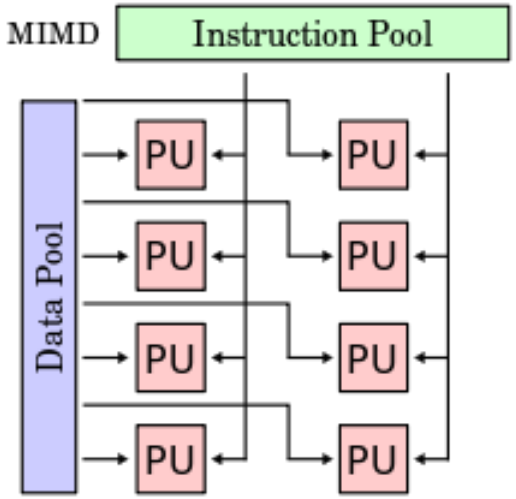


- Single Instruction Multiple Data (SIMD)
  - Each instruction is executed on a different set of data by the different processors

Categories of Computer Systems

- Multiple Instruction Single Data (MISD) stream
  - A sequence of data is transmitted to a set of processors, each executing a different instruction sequence





- Multiple Instruction Multiple Data (MIMD)
  - A set of processors simultaneously execute different instruction sequences on different data sets

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Sistemi Distribuiti 1/2

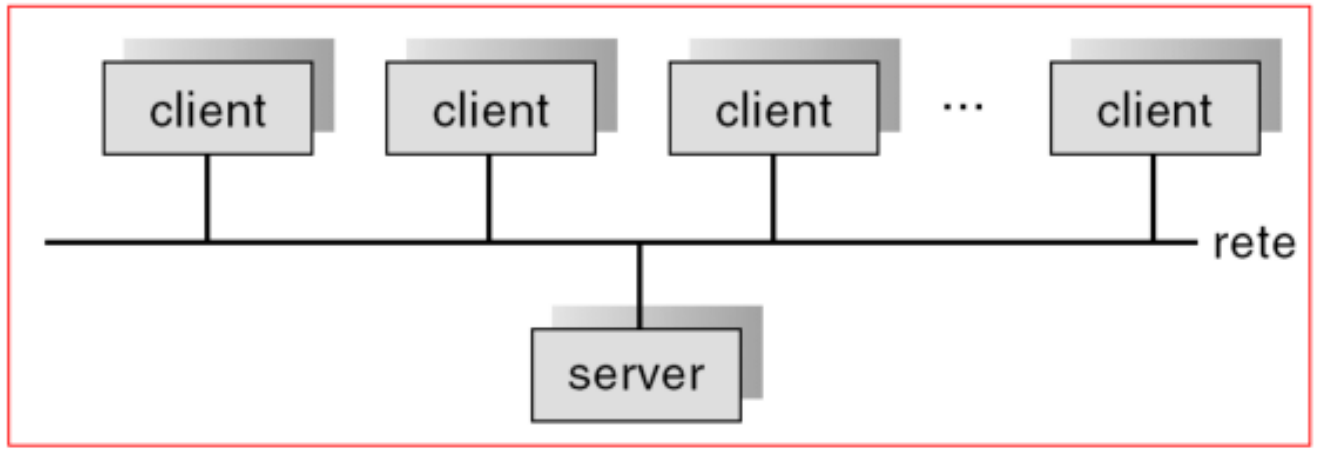
- Il calcolo viene distribuito tra diversi elaboratori **fisicamente distinti**
- Gli elaboratori possono essere eterogenei
- **Sistemi lascamente connessi** – ogni processore possiede una propria memoria locale; i processori comunicano tra loro mediante linee di comunicazione come bus ad alta velocità o linee telefoniche
- Vantaggi dei sistemi distribuiti
  - Condivisione delle risorse
    - Dati
    - Servizi
  - Rapidità di calcolo – distribuzione del carico
  - Affidabilità

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Sistemi Distribuiti 2/2

- **Necessitano di una infrastruttura di rete**
- Rete locale (Local area networks – *LAN*) o rete geografica (Wide area networks – *WAN*)
- Possono essere sistemi *client-server* o punto-a-punto (*peer-to-peer*) o
- Possono sfruttare *infrastrutture di cloud computing*



DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Cluster

Architettura con più computer fortemente connessi

- Capacità di elaborazione superiore ai sistemi SMP
  - Esecuzione contemporanea di un'applicazione su più pc
  - Richiede programmazione parallela (programmi con componenti eseguibili in parallelo)
- Economie di scala sulle periferiche
- Simmetrici o asimmetrici
- Affidabilità del sistema in caso di Guasti
  - Ogni pc è controllato da almeno un altro pc
  - Il quale recupera il lavoro in caso di guasto (Business Continuity)
- Usando computer disponibili sul mercato

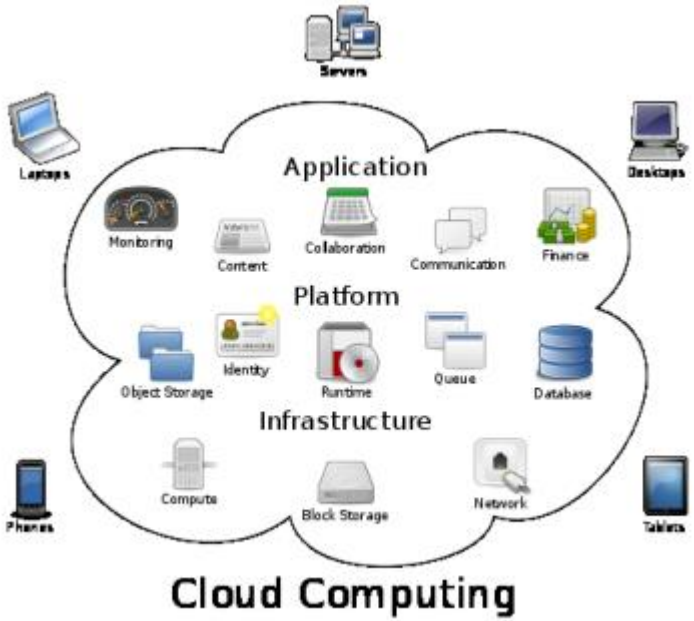
Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Cloud Computing

Un insieme di tecnologie informatiche che permettono l'utilizzo di risorse hardware (es. storage, CPU) o software distribuite e virtualizzate in Rete

- the cloud, in inglese - nuvola di risorse le cui caratteristiche non sono note all'utilizzatore
- Modello pay-as-you-go



Cloud computing =

- **SaaS** (Software as a Service)
- **PaaS** (Platform as a Service)
  - **CaaS** (Container as a Service)
  - **FaaS** (Function as a Service)
- **IaaS** (Infrastructure as a Service)

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: E. Tipologie di Sistemi di Elaborazione

Mainframe Classico

- Architettura orientata all'elaborazione di lavori non interattivi (job)
- Processore, memoria centrale (milioni di gigabyte), numerosi (1000) nastri/dischi, stampanti
- Elaborazione a lotti (batch)
- Riducono i tempi di processo raggruppando i job (processi) in batch (lotti) con necessità similari
- Esecuzione di numerosi lavori di routine alla volta, con prodigiose quantità di I/O e senza la presenza di alcun utente che interagisca con la macchina
- Sistemi monoprogrammati ➔ CPU sottoutilizzata
- Sistemi multiprogrammati
  - memoria centrale ripartita tra job (multiprogrammazione)
  - condivisione CPU (multitasking)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Mainframe Moderno (HPC)

Grandi server (spesso raffredati a liquido)

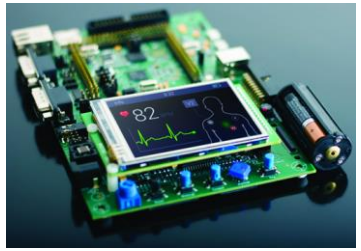
- Supportano molti utenti operanti contemporaneamente
- Alla base dei giganteschi server web centralizzati!
- CPU, memoria centrale, terminali, nastri/dischi, stampanti
- Elaborazione contemporanea flussi di attività (processi)
- Elaborazione di transazioni e condivisione del tempo macchina
- Sistemi multiutente
- ripartizione memoria tra processi (multiprogrammazione)
- condivisione CPU (multitasking)
- gestione CPU in condivisione di tempo (time sharing)

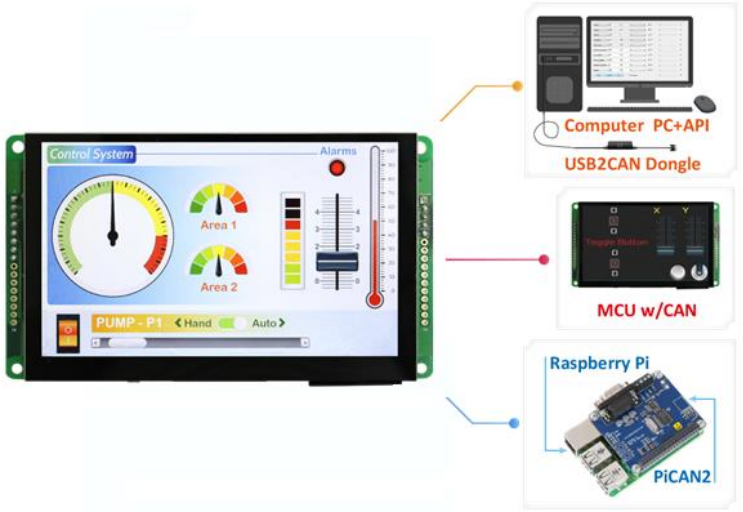
Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione

Sistema Dedicato (Embedded)





Sistemi di elaborazione dedicati a supportare una sola applicazione

- Ad esempio: elettrodomestici, sistemi hi-fi, motore automobile (CANBus), sistemi biomedicali, protesi, carte di credito, ecc..
- Tutto il software è su ROM
- Ridotte caratteristiche di prestazioni computazionali, memoria e periferiche
- Sistemi per SmartCard
- Sistemi operativi proprietari, JavaCard
- Hanno spesso caratteristiche di real-time e multi-tasking

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

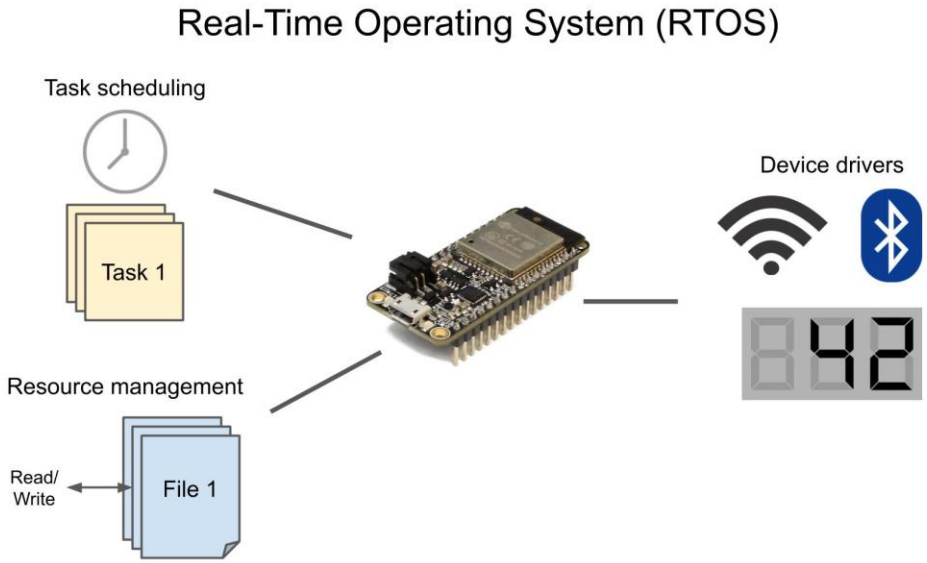
Sapienza Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione
Real-Time

- Risposta agli eventi in tempo reale
  - La risposta viene fornita rispettando rigorosi vincoli temporali
    - sistemi in tempo reale stretto (hard real-time)
    - sistemi in tempo reale lasco (soft real-time)
- Architettura con capacità di scambiare segnali

con il mondo esterno attraverso sensori e attuatori

- schede di acquisizione segnali (Input digitali/analogici), schede di attuazione controlli (Output digitali/analogici)



Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione

Sistemi Multimediali

Sistema con supporti avanzati per l'interazione multimediale

- Ad esempio i sistemi di controllo delle console giochi (Nintendo Wii, Microsoft X-box, Sony PlayStation, ecc..) ma anche PuzzleLinux
- La trasmissione dei dati deve attenersi a specifiche

frequenze

- Sistemi interattivi multiprocesso
- ripartizione memoria tra processi (multiprogrammazione)
- condivisione CPU (multitasking)
- gestione CPU in condivisione di tempo (time sharing)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: E. Tipologie di Sistemi di Elaborazione

PDA/Smartphone

Sistemi di elaborazione portatili e di dimensioni estremamente ridotte, orientati al supporto di attività personali (Personal Digital Assistant - PDA)

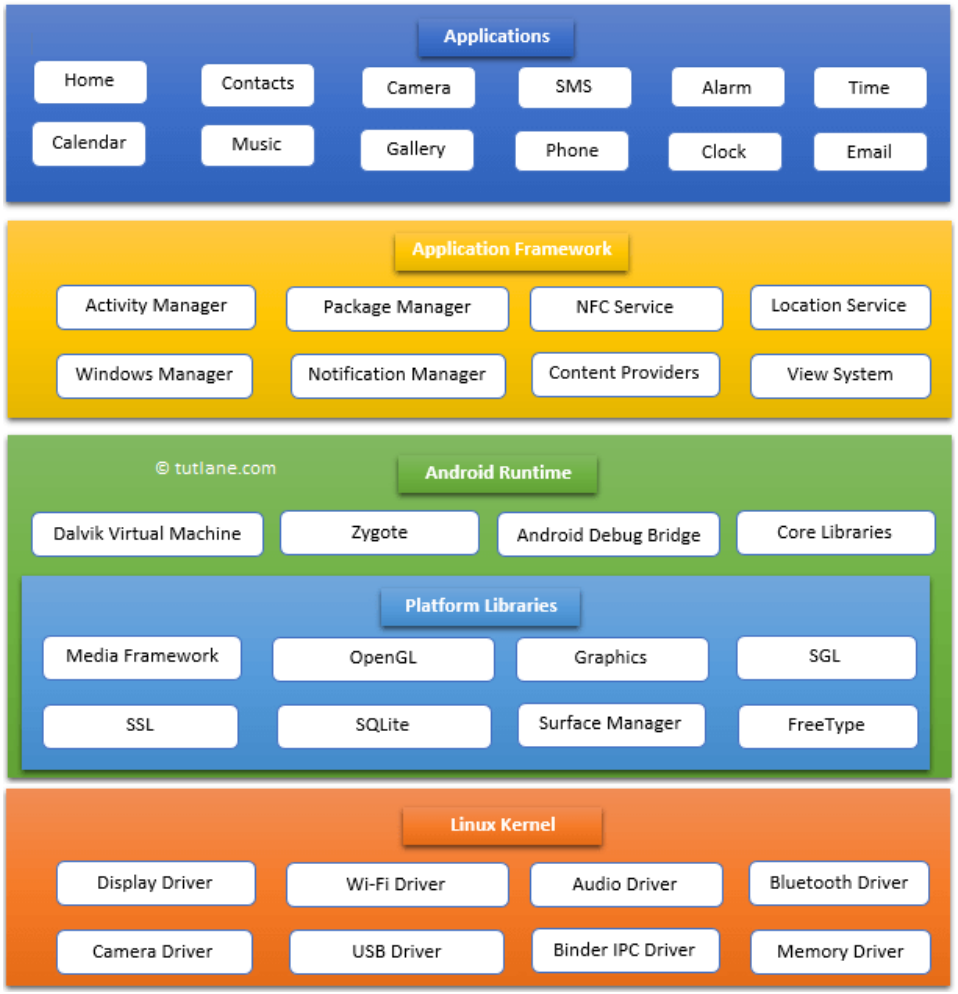
- Sistemi palmari
- Telefoni cellulari

Sistemi interattivi multiprocesso con

- Ridotto consumo di potenza
- Basso numero di processi

Gli smartphone sono una loro evoluzione (S.O. iOS e Android)

- Multi-core
- Interfacce multi-touch
- Prestazioni elevate



DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Microkernel

- A microkernel is a small OS core that provides the foundation for modular extensions.
- Big question is how small must a kernel be to qualify as a microkernel
  - *Must* drivers be in user space?
- In theory, this approach provides a high degree of flexibility and modularity.

Kernel Architecture

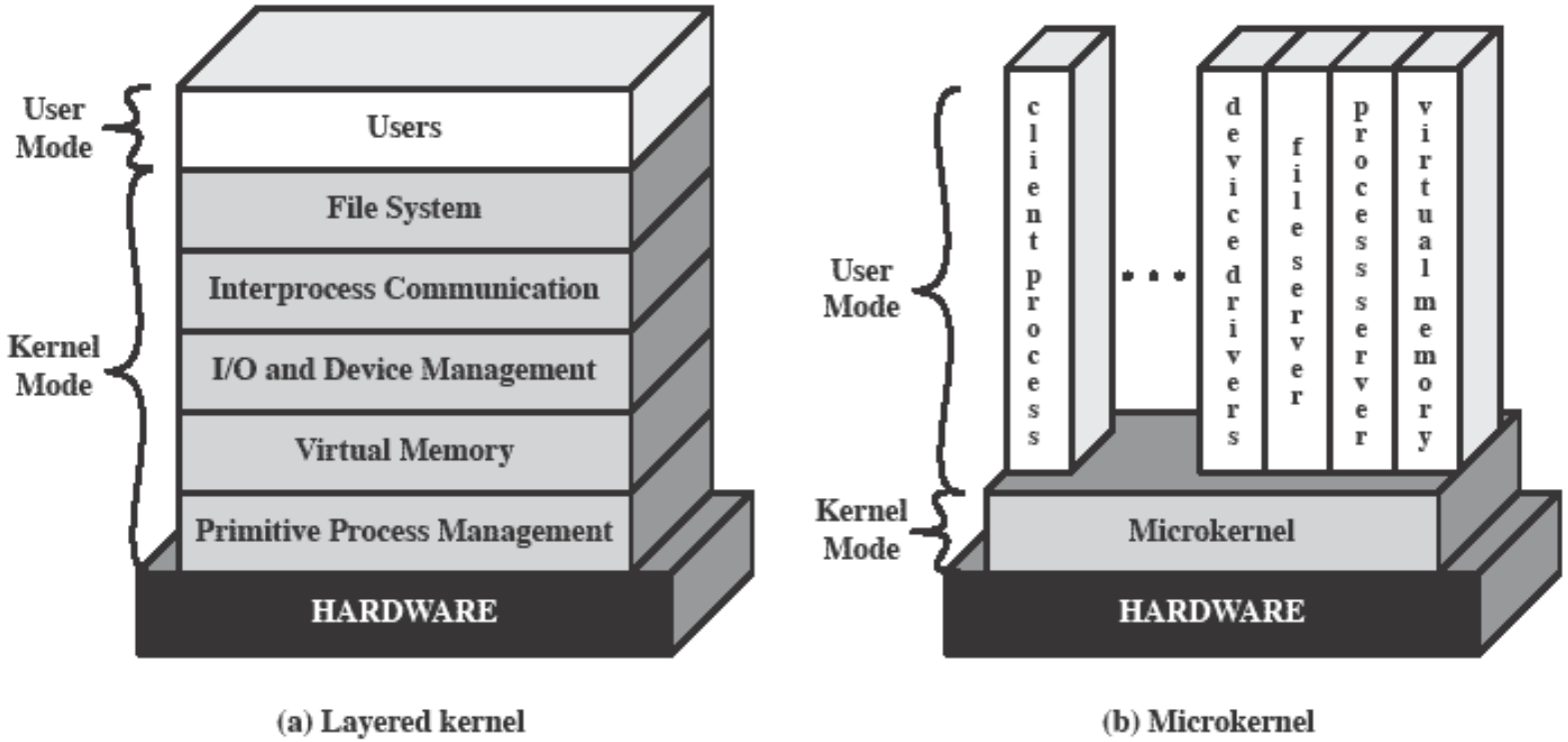


Figure 4.10 Kernel Architecture

Microkernel Design: Memory Management

- Low-level memory management - Mapping each virtual page to a physical page frame
  - Most memory management tasks occur in user space

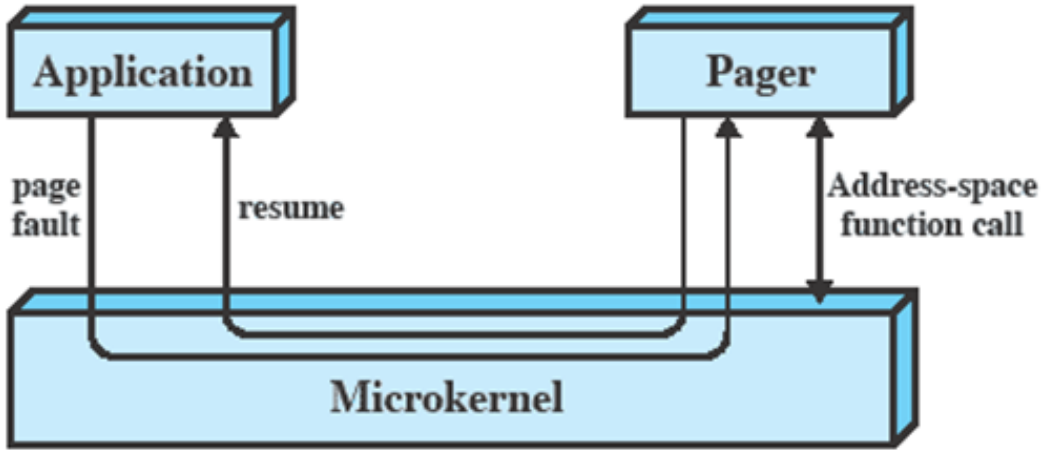


Figure 4.11 Page Fault Processing

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Microkernel Design: Interprocess Communication

- Communication between processes or threads in a microkernel OS is via messages.
- A message includes:
  - A header that identifies the sending and receiving process and
  - A body that contains direct data, a pointer to a block of data, or some control information about the process.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Microkernel Design: I/O and interrupt management

- Within a microkernel it is possible to handle hardware interrupts as messages and to include I/O ports in address spaces.
  - a particular user-level process is assigned to the interrupt and the kernel maintains the mapping.

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Benefits of a Microkernel Organization

- Uniform interfaces on requests made by a process.
- Extensibility
- Flexibility
- Portability
- Reliability
- Distributed System Support
- Object Oriented Operating Systems

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

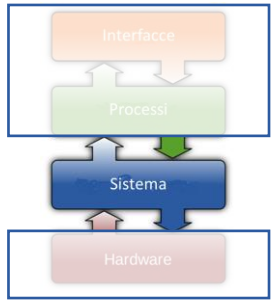
S.O.: Obiettivi, Funzioni, Servizi

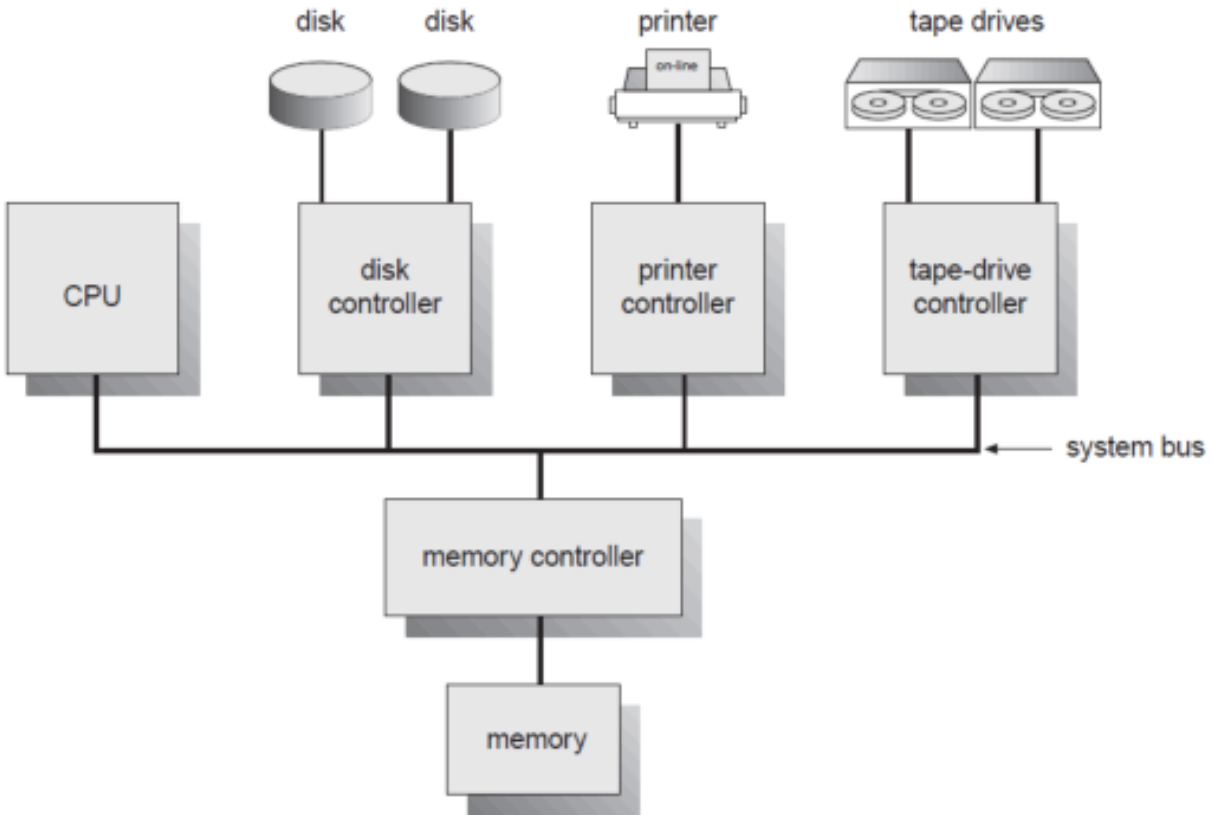
Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

Input/Output





Un calcolatore è composto da una CPU e diversi controllori di Dispositivi:

- Bus: mezzo condiviso di comunicazione
- RAM: luogo condiviso di allocazione dei dati da elaborare
- CPU: strumento condiviso di elaborazione

Lo spostamento dei dati da/verso la CPU prende il nome di **I/O**

- L'I/O avviene tra il dispositivo e il buffer locale del controller
- La CPU guida poi lo spostamento dei dati (che viaggiano sul BUS) dal **buffer** locale dei controller alla memoria, e viceversa

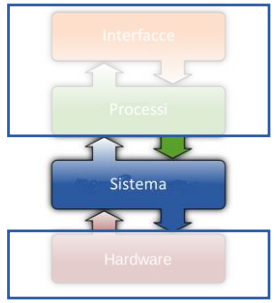
Rendere il sistema quanto meno impattato possibile dalla lentezza dei dispositivi periferici.

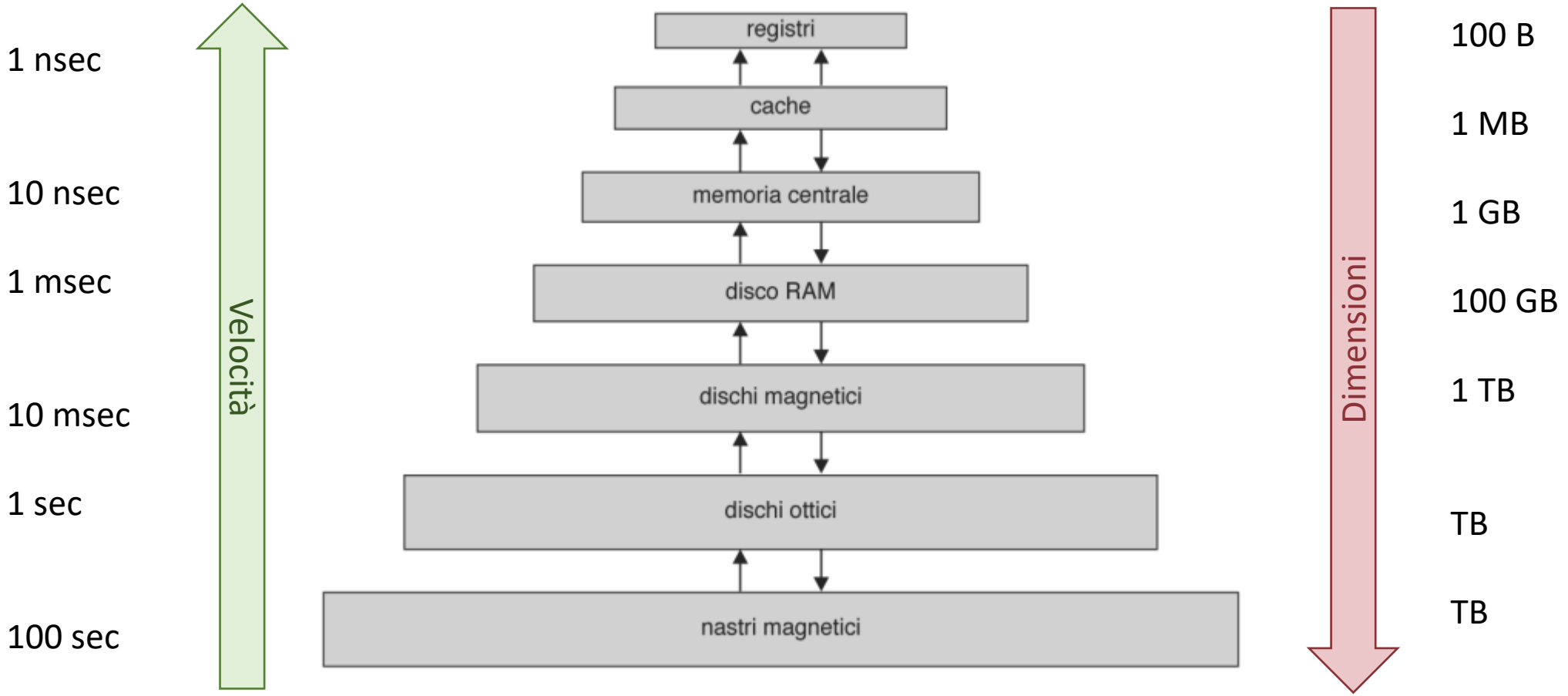
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

Input/Output

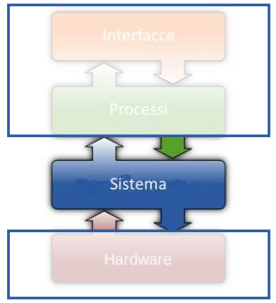


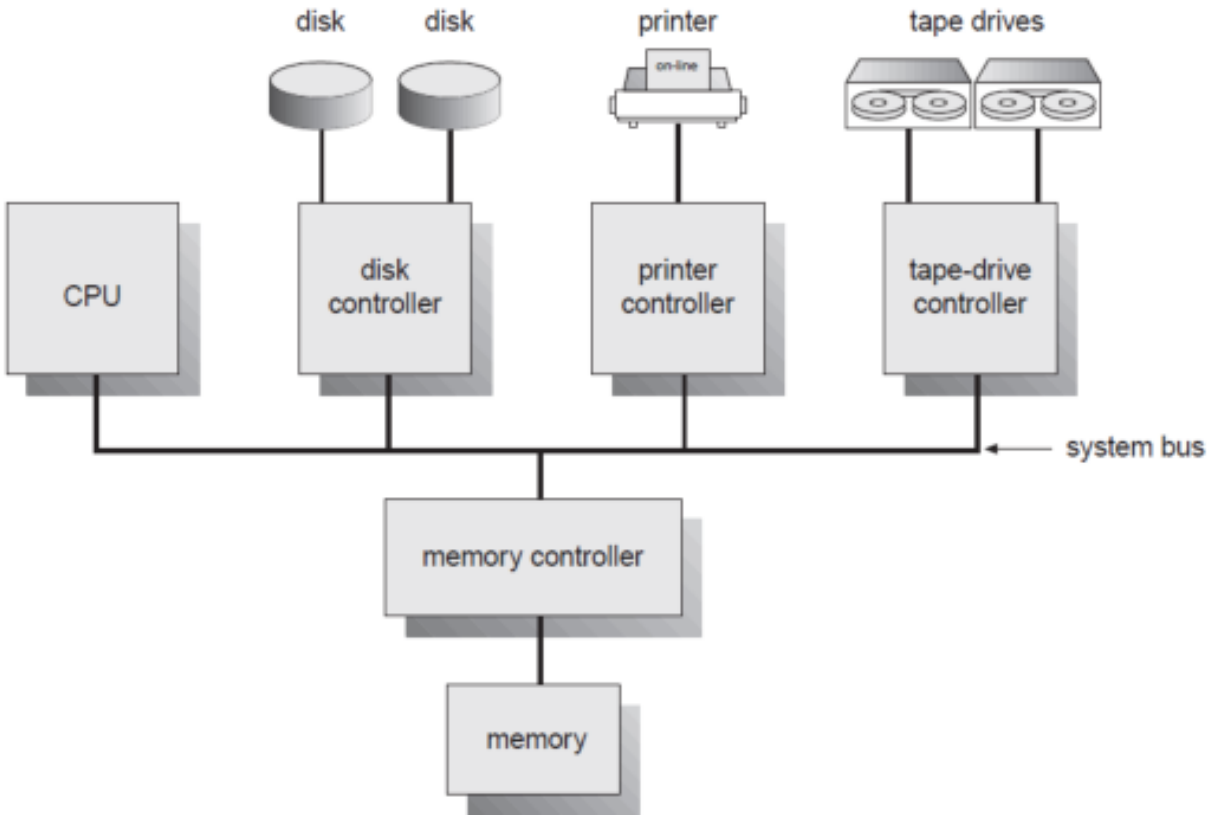


Cache: 1°, 2° livello specifiche per Core. 3° livello condiviso tra core (➔ Meltdown, Spectre).

Operating Systems: Obiettivi Funzioni Servizi

Input/Output





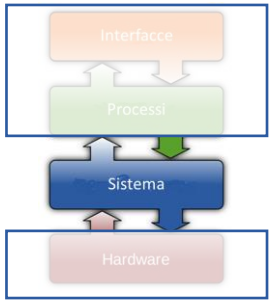
Tecniche di I/O:

1. I/O programmato
2. I/O gestito con interrupt
3. I/O con DMA
4. I/O con canali (processore separato)

Rendere il sistema quanto meno impattato possibile dalla lentezza dei dispositivi periferici.

Operating Systems: Obiettivi Funzioni Servizi

Input/Output

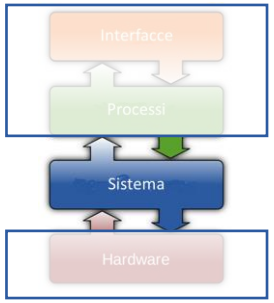


I/O programmato

- È tipicamente utilizzato da architetture di fascia bassa o al controllo industriale con basse necessità di I/O multiplo.
- Per ogni scrittura il processore esegue una istruzione di I/O
  - Per scrivere un blocco di n caratteri la CPU esegue n istruzioni di output carattere
- Per ricevere un dato la CPU legge in un ciclo stretto la porta di I/O in attesa del carattere
  - L'informazione di input disponibile viene tipicamente affidata ad un bit di controllo
- **Svantaggio**: non è ottimale nell'ipotesi in cui l'I/O di quantità di dati relativamente grosse occupi una parte rilevante del tempo macchina
  - Il microprocessore spenderà la maggior parte del suo tempo in operazioni di I/O

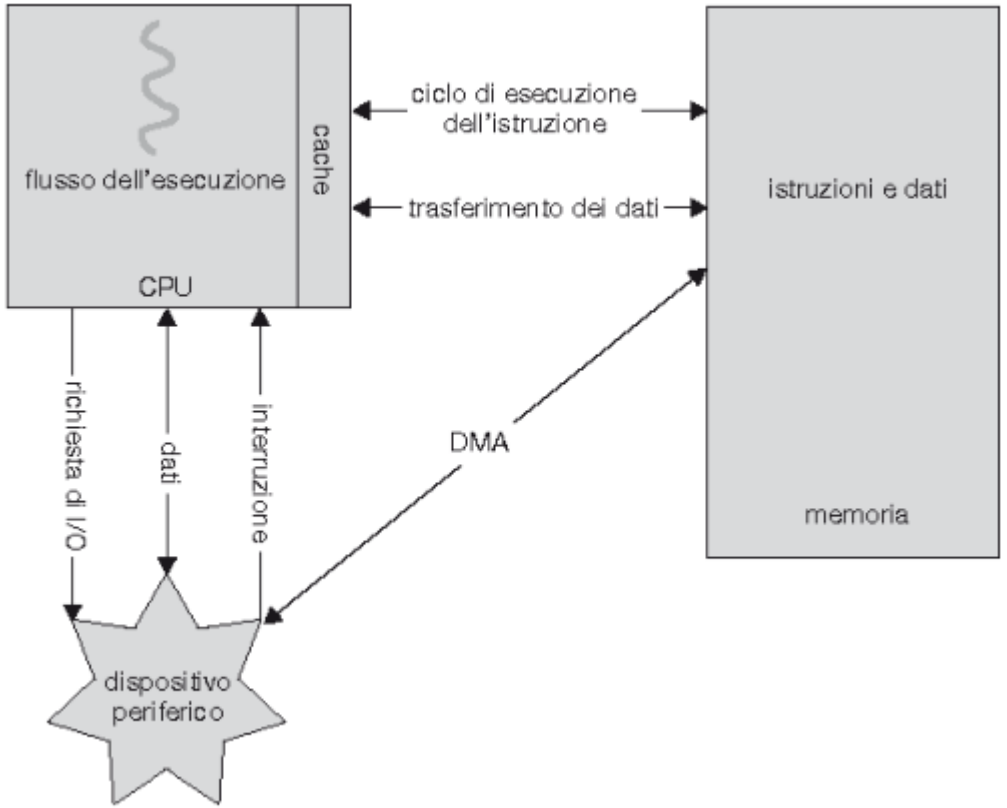
Operating Systems: Obiettivi Funzioni Servizi

Input/Output



Interrupt
Ogni controllore di dispositivo I/O dispone di una memoria interna (buffer)

- Per avviare un'operazione di read di un byte il controllore
- copia i dati sul buffer
- Informa il driver del dispositivo (tramite interrupt)
- Il driver cede il controllo al SO restituendo un puntatore ai dati nel buffer
- Oneroso per grosse dimensioni (molti byte)



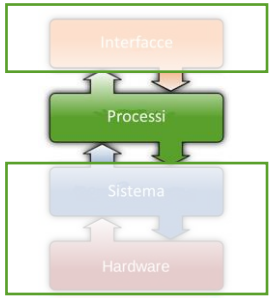
DMA (Direct Memory Access)
un volta impostato il buffer, il controllore del dispositivo I/O
trasferisce direttamente grandi porzioni di dati dal dispositivo alla memoria. È necessario un solo interrupt, quindi un solo interazione con la CPU per ogni blocco di byte.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

Virtualizzazione



Processo è un programma in esecuzione:

- Indipendente dagli altri programmi
- Che non deve interferire con gli altri programmi
- Che trae giovamento dalla gestione ottimizzata delle risorse
- Che deve sottostare alla allocazione e condivisione ordinata delle risorse rispetto a spazio/tempo

«Legum servi sumus ut liberi esse possimus»
[M. T. Cicerone]

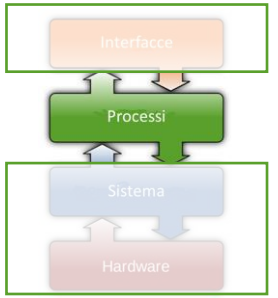
Rendere il sistema quanto più possibile «dedicato» all'esecuzione di ogni processo.

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

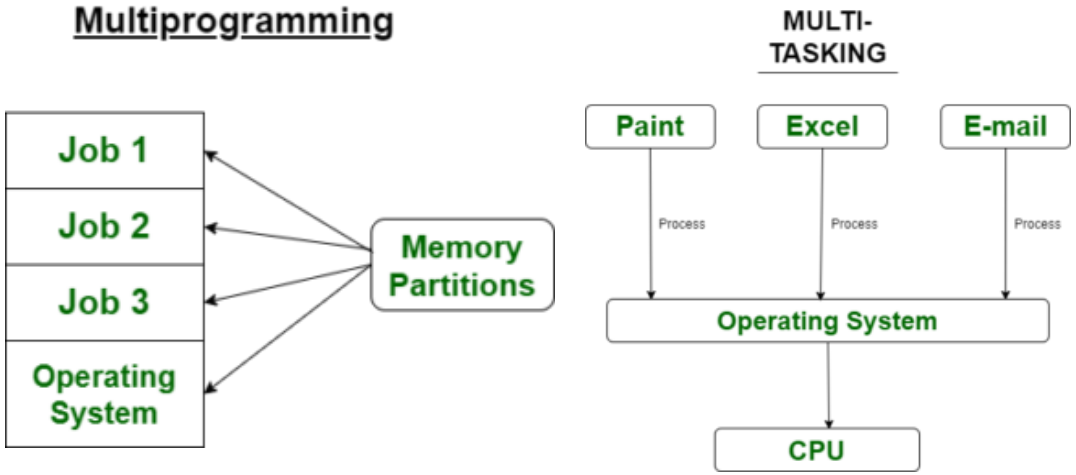
Operating Systems: Obiettivi Funzioni Servizi

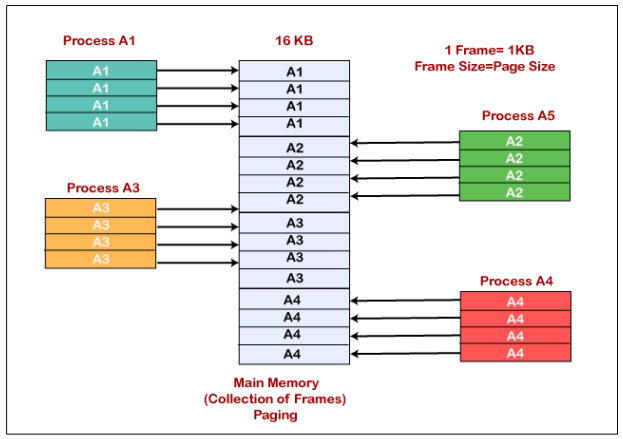
Virtualizzazione



Condivisione Temporale della CPU

- **Elaborazione Batch**: un processo alla volta dall'inizio alla fine della elaborazione, senza comunicazioni intermedia.
- **Multiprogrammazione**: scheduling dei job basata su un corretto spooling. Ogni processo rimane in esecuzione fintantoché non genera un interrupt (eccezione). L'I/O viene attivato contemporaneamente al **Context Switch**.
- **Time-Sharing** (**Multitasking**): l'alternanza dei processi nella CPU è assegnata ad una condivisione equa di tempo nella CPU fra processi ((Timer: registro scalante settato dal S.O. che genera un interrupt)





Condivisione Spaziale della RAM

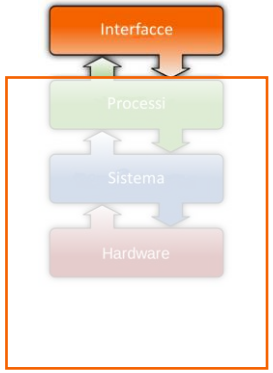
Ogni processo ha accesso solo alla porzione di memoria riservataglia dal S.O.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

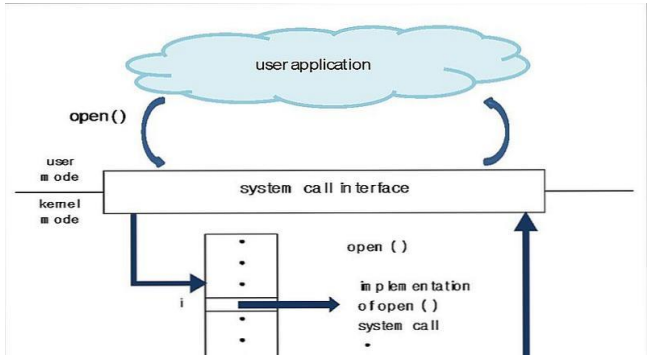
Astrazione

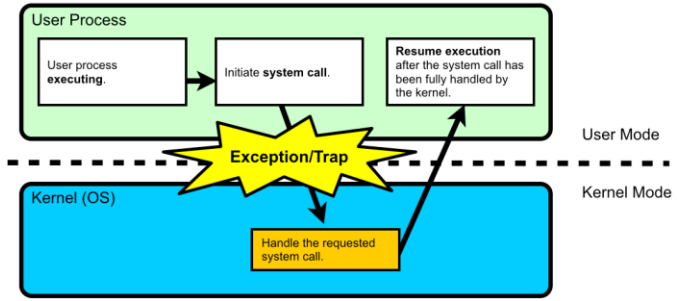


Semplificazione dell'uso dell'elaboratore (facilità di programmazione)

- **System Call**: esplicita richiesta da parte del programma di accedere a determinati servizi del S.O. Interfaccia tra processo e S.O..

- **Trap**: routine eseguite dal S.O. a seguito di un interrupt software dovuto a:
  - Errore nella computazione
  - Tentativo di eseguire un commando privilegiatoessendo in modalità utente

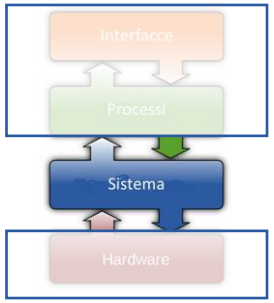




Alzare il più possibile il livello di astrazione dei componenti del sistema di elaborazione.

Operating Systems: Obiettivi Funzioni Servizi

Error Detection



Capacità di rilevare errori nella CPU o nei dispositive, Quali:

- Divisione per 0
- Fine della carta
- Guasto di una connessione di rete

Per ogni errore identificato (interrupt) è necessario eseguire una appropriata azione di recovery (trap)

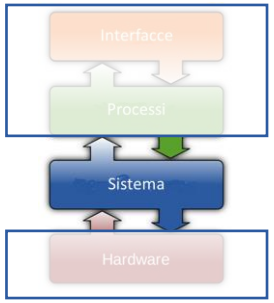
Impedire che il sistema vada in blocco.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

Dual-Mode



- Una volta avviato il sistema operativo, la CPU può operare in due modalità
  1. **User mode** – la CPU sta eseguendo codice di un utente
  2. **Kernel mode** (anche supervisor mode, system mode, monitor mode) – la CPU sta eseguendo codice del sistema operativo

- La CPU ha un **Mode bit** nel *registro di stato* (Process Status Word o PSW) che indica in quale modo si trova: kernel (0) o user (1)

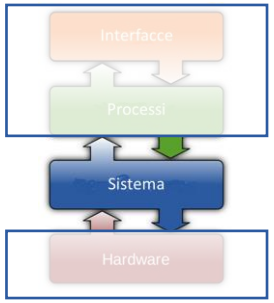
Rendere il sistema quanto più possibile scevro da esecuzioni pericolose.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

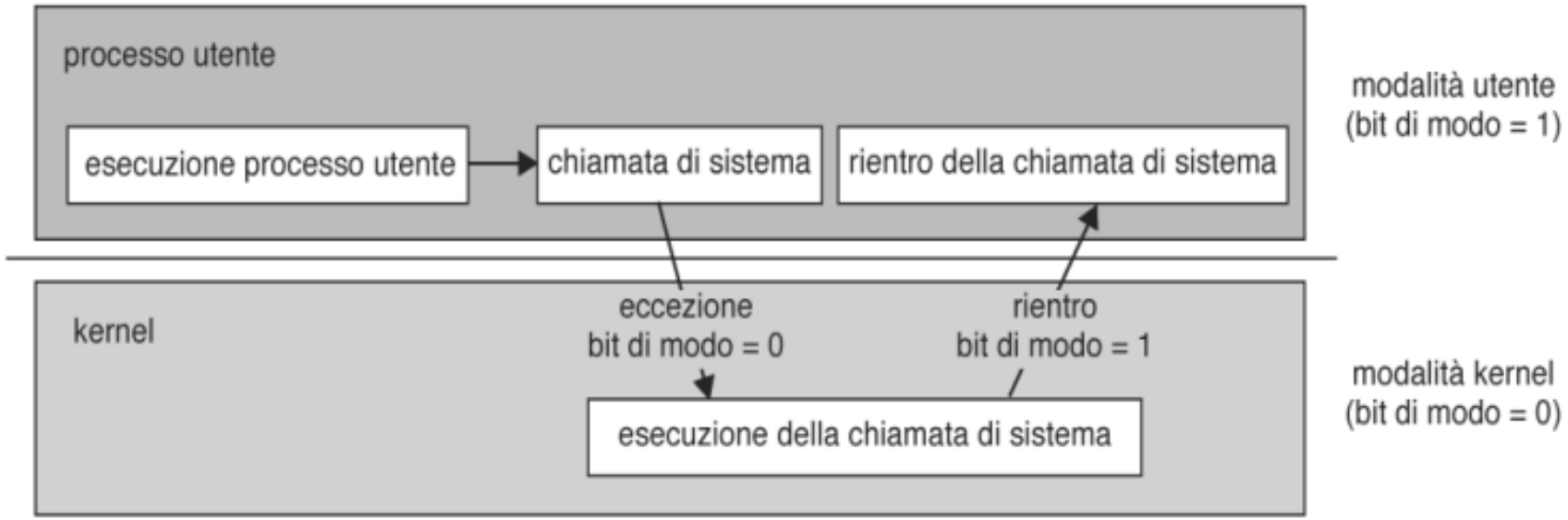
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

Dual-Mode

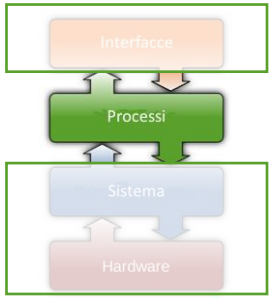


- Protezione dei registri da accessi erronei/intenzionali
- L'accesso completo all'hardware solo in modo kernel
- Passaggio controllato da modo utente a modo kernel
  - interruzioni
  - chiamate di sistema (system call), trap/fault



Operating Systems: Obiettivi Funzioni Servizi

Process Management/Memory Management



Gestione dei Processi in relazione alla CPU (esecuzione)

- Creazione e terminazione dei processi
- Sospensione e riattivazione dei processi
- Schedulazione dei processi
- Sincronizzazione tra processi
- Gestione di situazioni di stallo (deadlock)
- Comunicazioni tra processi (memoria condivisa o scambio di messaggi)

Gestione dei Processi in relazione alla Memoria (allocazione)

- Multiprogrammazione
- Allocazione e deallocazione della memoria ai processi
- Caricamento e scaricamento di processi e di loro porzioni in memoria centrale
- Protezione della memoria centrale

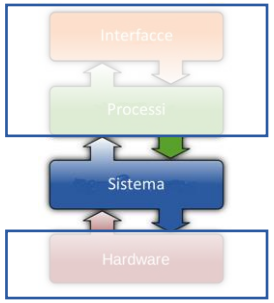
Creare l'ambiente di esecuzione per i programmi.

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

Input/Output (digressione)

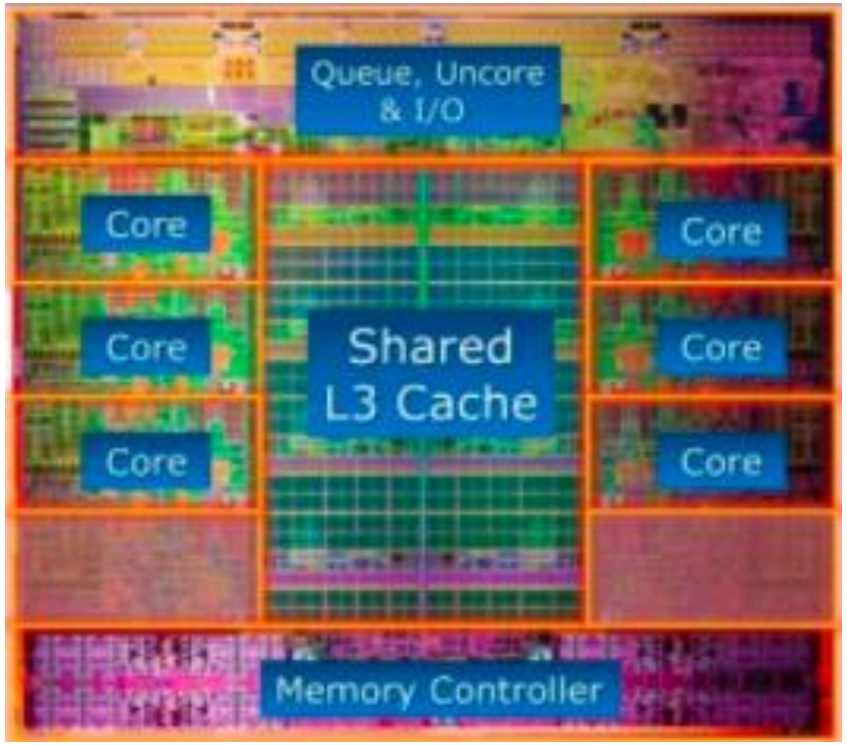


Architettura Intel Core i7

Cache di 3° livello tra i core contenuti nella stessa CPU.

Non previsti meccanismi di difesa da parte della CPU.

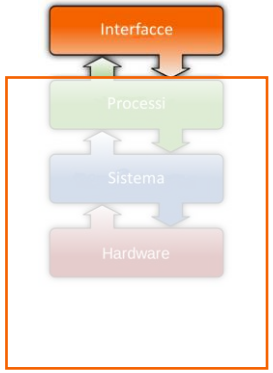
Il S.O. non può impedire che il processo eseguito sul core A possa accedere allo spazio di cache L3 usato dal processo eseguito sul core B.



Cache: 1°, 2° livello specifiche per Core. 3° livello condiviso tra core (➔ Meltdown, Spectre).

Operating Systems: Obiettivi Funzioni Servizi

File System



File: contenitore di informazioni/dati in formato digitale

Nasconde all'utente l'organizzazione della memoria:

- File e directory
- Creazione e cancellazione
- Lettura e scrittura
- Copiatura
- Ricerca
- Salvataggio e ripristino
- Protezione e sicurezza (diritti di accesso)

In Unix "Everything is a File".

1. - : regular file
2. d : directory
3. c : character device file
4. b : block device file
5. s : local socket file
6. p : named pipe
7. l : symbolic link

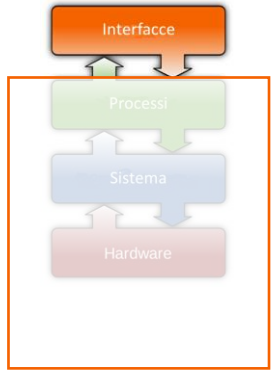
Il file è una pregevole astrazione del S.O..

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

File System



/ "root"

/bin
"essential user command binaries"
- bash
- cat
- chmod
- cp
- date
- echo
- grep
- gunzip
- gzip
- hostname
- kill
- less
- ln
- ls
- mkdir
- more
- mount
- mv
- nano
- open
- ping
- ps
- pwd
- rm
- sh
- su
- tar
- touch
- umount
- uname

/etc
"configuration files for the system"
- crontab
- cups
- fonts
- fstab
- host.conf
- hostname
- hosts
- hosts.allow
- hosts.deny
- init
- init.d
- issue
- machine-id
- mtab
- mtools.conf
- nanorc
- networks
- passwd
- profile
- protocols
- resolv.conf
- rpc
- securetty
- services
- shells
- timezone

/sbin
"essential system binaries"
- fdisk
- fsck
- getty
- halt
- ifconfig
- init
- mkfs
- mkswap
- reboot
- route

/usr
"read-only user application support data & binaries"

- **/usr/bin**
  "most user commands"
- **/usr/include**
  "standard include files for 'C' code"
- **/usr/lib**
  "obj, bin, lib files for coding & packages"
- **/usr/local**
  "local software"
  - /usr/local/bin
  - /usr/local/lib
  - /usr/local/man
  - /usr/local/sbin
  - /usr/local/share
- **/usr/share**
  "static data sharable accross all architectures"
  - /usr/share/man
    "manual pages"

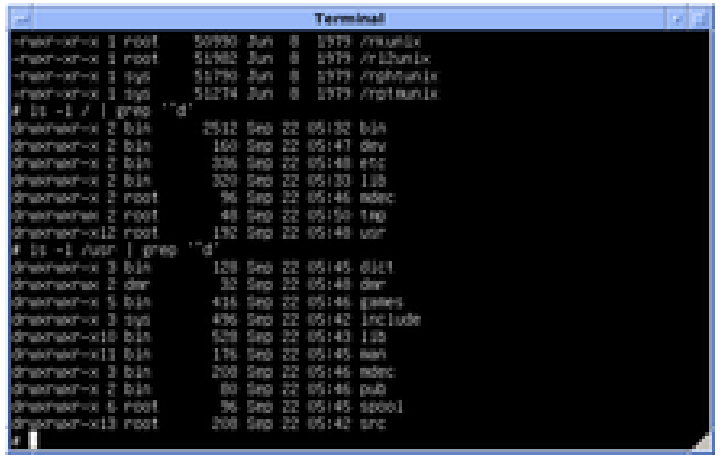
/var
"variable data files"

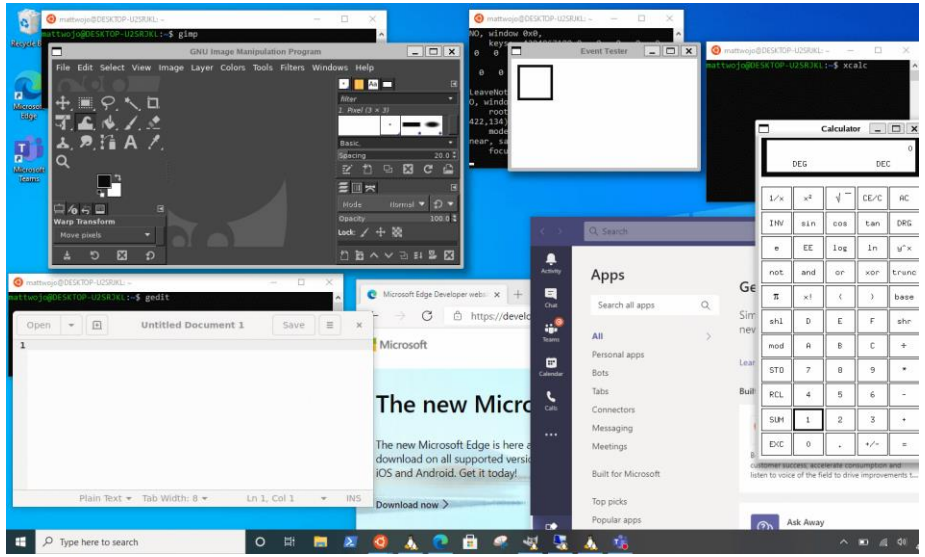
- **/var/cache**
  "application cache data"
- **/var/lib**
  "data modified as programmes run"
- **/var/lock**
  "lock files to track resources in use"
- **/var/log**
  "log files"
- **/var/opt**
  "variable data for installed packages"
- **/var/spool**
  "tasks waiting to be processed"
  - /var/spool/cron
  - /var/spool/cups
  - /var/spool/mail
- **/var/tmp**
  "temporary files saved between reboots"

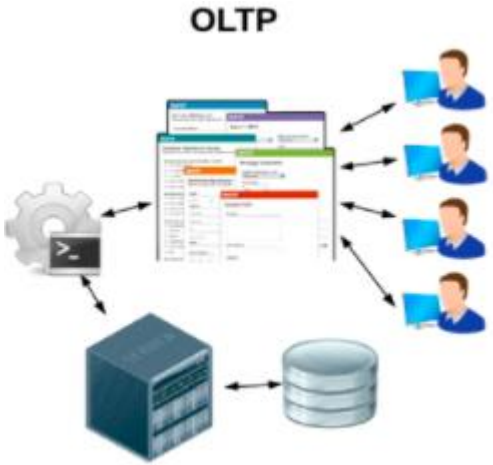
- **/dev**
  "device files incl. /dev/null"
- **/home**
  "user home directories"
- **/lib**
  "libraries & kernel modules"
- **/mnt**
  "mount files for temporary filesystems"
- **/opt**
  "optional software applications"
- **/proc**
  "process & kernel information files"
- **/root**
  "home dir. for the root user"

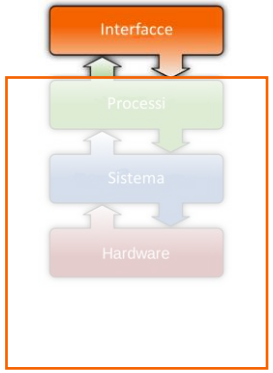
Operating Systems: Obiettivi Funzioni Servizi

Shell/GUI/Transaction







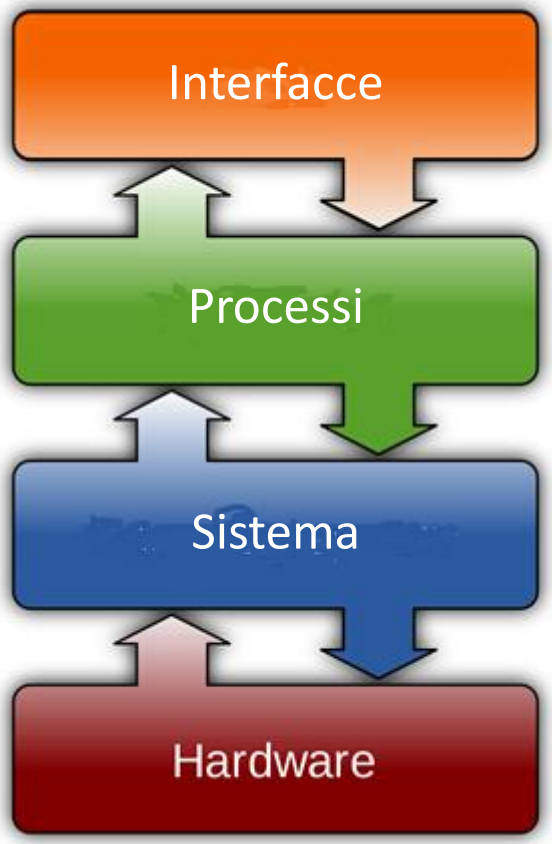


Rendere il sistema «user-friendly» per gli utenti, secondo necessità.

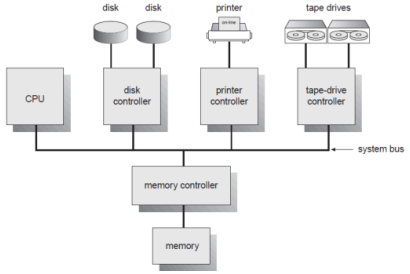
DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

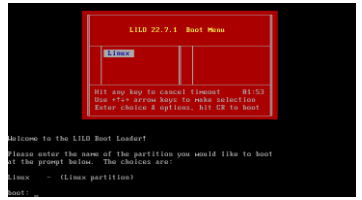
Operating Systems: Introduzione ai Layers



Obiettivi	Funzioni	Servizi
Astrazione	File System Shell/GUI/Transaction	Authentication/Authorization Accounting
Virtualizzazione	Process Management Memory Management	IPC System Calls
Input/Output	Error Detection Dual-Mode	Boot Interrupt Handling



	B	C
	Formulas	Formula Results
	=1/0	#DIV/0!
	=A2/A3	#DIV/0!
	=QUOTIENT(A2,A3)	#DIV/0!

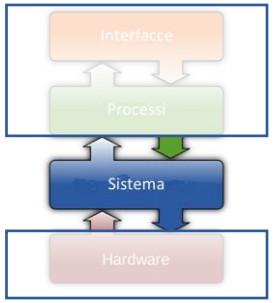


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

Boot



I passi sono:

- Accensione
- Esecuzione del BIOS (in ROM o EEPROM) – *Basic I/O System*
  - Inizializza tutti i registri della CPU, i controllori delle periferiche e la memoria centrale
- Esecuzione del *bootstrap program o bootstrap loader*
  - caricamento del kernel del SO da una unità di memoria ed esecuzione
- Controllo del sistema da parte del SO

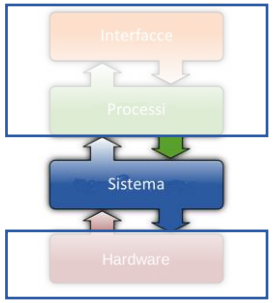
Permettere al sistema di ripartire in ogni momento.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

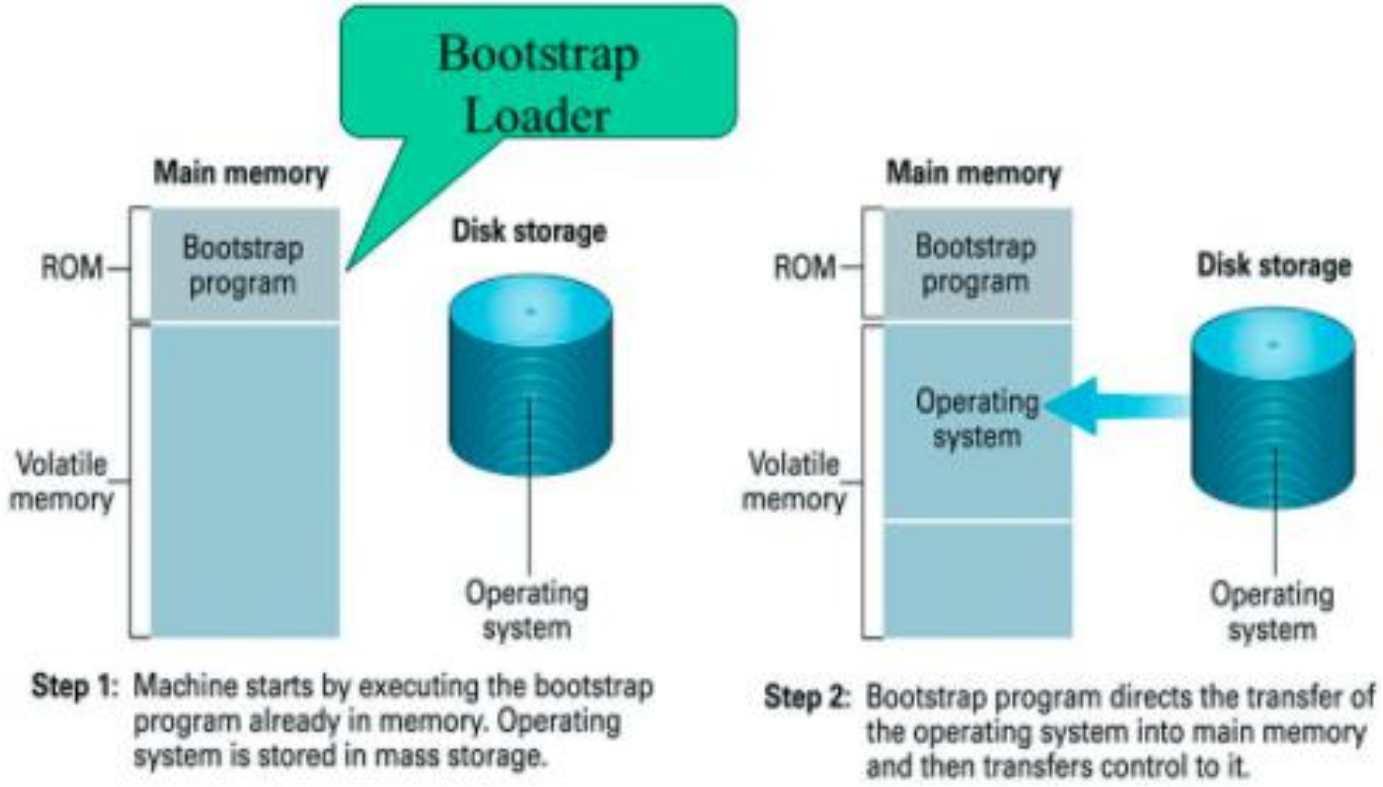
Boot



Posizionamento del S.O.

In generale, il SO è nel disco (in una partizione detta attiva) ed il bootstrap program è in memoria ROM o EEPROM

- Il BIOS cerca periferiche avviabili e dà il controllo al bootstrap program
- Il bootstrap program si avvale di un frammento di codice (che sà in quale parte del disco si trova il SO) che risiede in un blocco del disco ad una posizione fissa (tipicamente 0) – boot block o master boot record (MBR)
- La partizione del disco che contiene il boot block è detta di avvio – boot partition

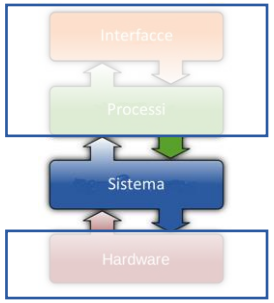


Permettere al sistema di ripartire in ogni momento.

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI
SAPIENZA UNIVERSITÀ DI ROMA

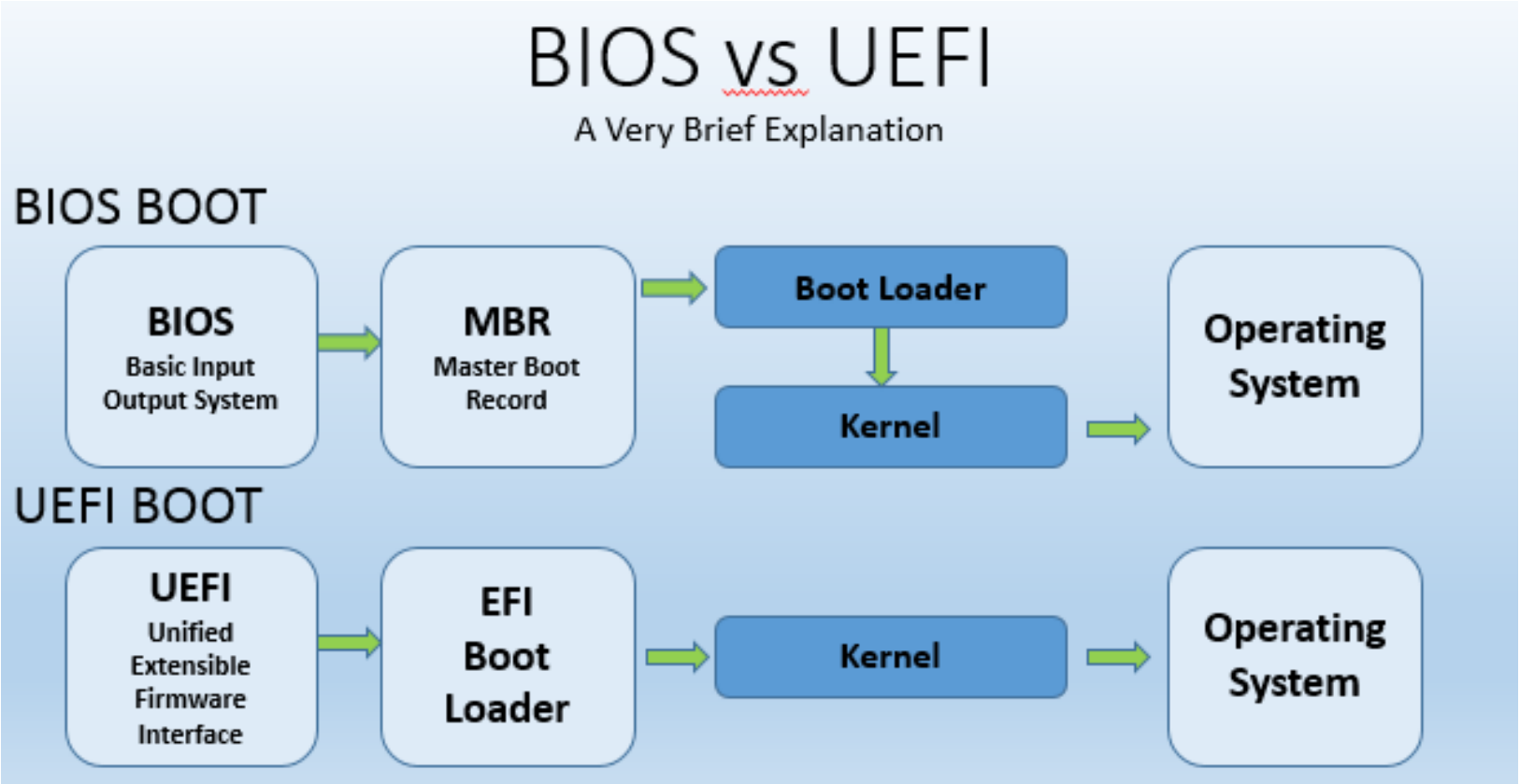
Operating Systems: Obiettivi Funzioni Servizi

Boot



Posizionamento del S.O.

- I moderni sistemi UEFI hanno una partizione apposite dedicata alle immagini di S.O.



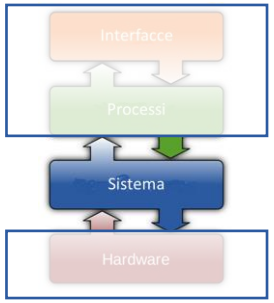
Permettere al sistema di ripartire in ogni momento.

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

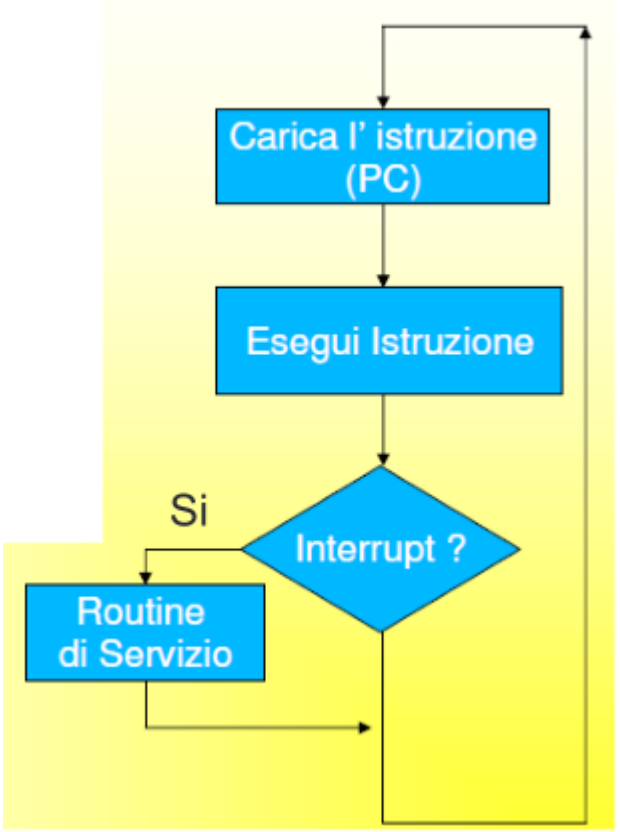
SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

Interrupt Handling



- Il sistema esegue sequenzialmente le istruzioni
- All'arrivo di un interrupt (I/O guidato da interrupt), la CPU invoca un "gestore" (*routine di servizio*) opportuno attraverso l'**interrupt vector**
  - Il **vettore di interrupt contiene gli indirizzi in memoria di tutte le routine di servizio**
- L'HW deve salvare l'indirizzo dell'istruzione interrotta, in modo tale che, una volta ultimata la gestione di interrupt, possa essere ripristinato.
- NOTA: **Interrupt in arrivo sono disabilitati mentre un altro interrupt viene gestito**, per evitare che vadano perduti

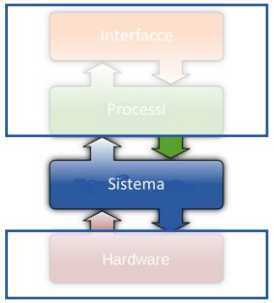


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

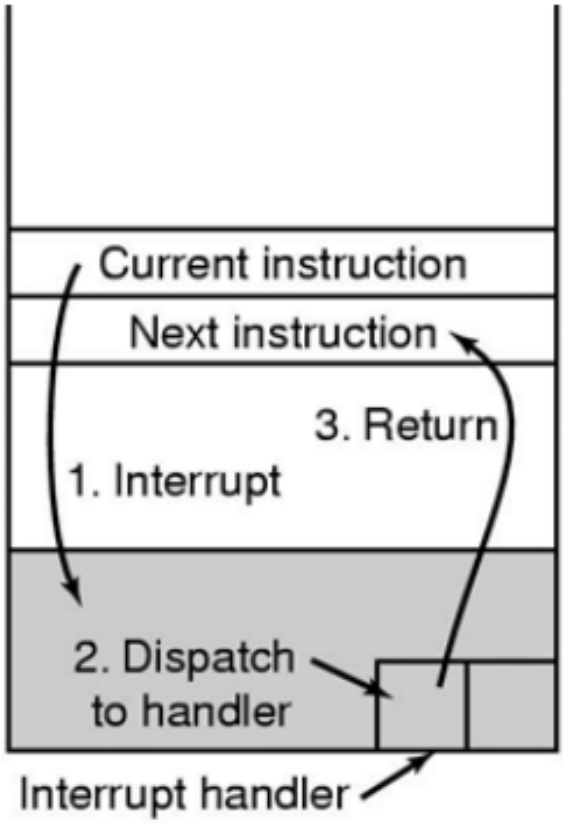
Operating Systems: Obiettivi Funzioni Servizi

Interrupt Handling

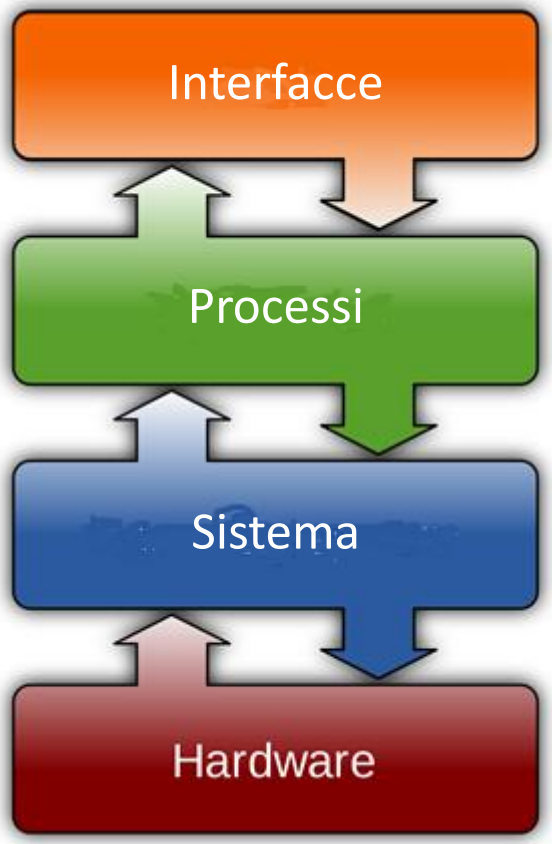


Passi necessari alla gestione di una interruzione:

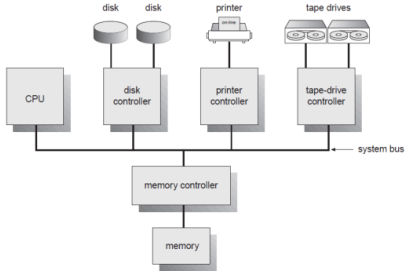
1. CPU rileva l'interruzione e legge l'indirizzo del dispositivo (*device*) sul bus
   - salvataggio dei registri PC e PSW (stato della CPU) sullo stack, passaggio in modo kernel
2. L'indirizzo del dispositivo viene usato come indice nella tabella dei gestori delle interruzioni (*interrupt vector*)
3. Il gestore selezionato prende il controllo e svolge le operazioni necessarie
4. Quando il gestore termina:
   - ripristino PC e PSW , ritorno in modo utente
   - si esegue l'istruzione successiva a quella interrotta



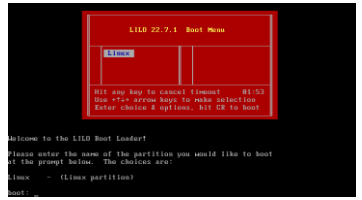
Operating Systems: Introduzione ai Layers



Obiettivi	Funzioni	Servizi
Astrazione	File System Shell/GUI/Transaction	Authentication/Authorization Accounting
Virtualizzazione	Process Management Memory Management	IPC System Calls
Input/Output	Error Detection Dual-Mode	Boot Interrupt Handling



	B	C
	Formulas	Formula Results
	=1/0	#DIV/0!
	=A2/A3	#DIV/0!
	=QUOTIENT(A2,A3)	#DIV/0!

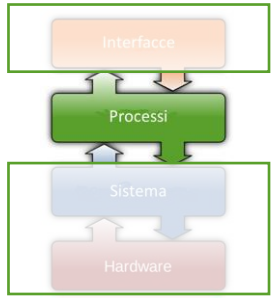


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

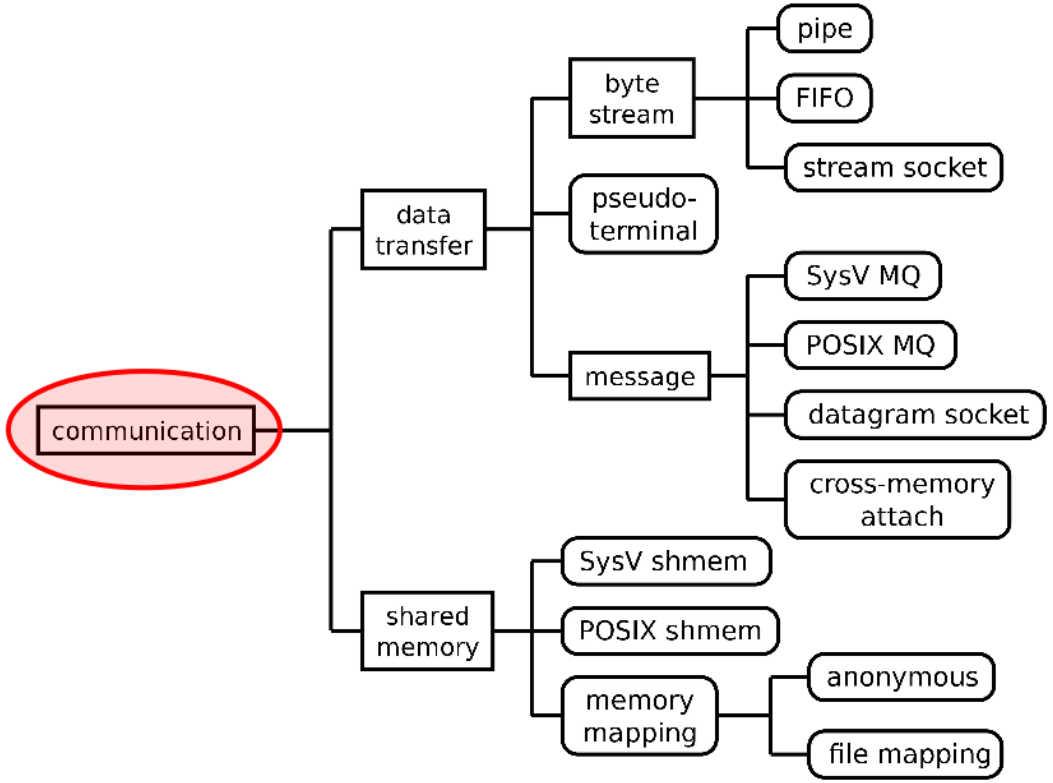
SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

IPC



Communication



Inter-Process Communication

- **Segnale** (signals), per eventi asincroni
- **Pipe**, per reindirizzare il risultato della elaborazione
- **Socket**, scambio di datagram
- **Code di Messaggi** (Message Queue), per comunicazioni in multicast
- **Memoria Condivisa** (Shared Memory), per condividere dati fra processi con trust reciproco
- **Semaforo** (Semaphore), per coordinare processi che lavorano su una stessa risorsa (-> critical region)

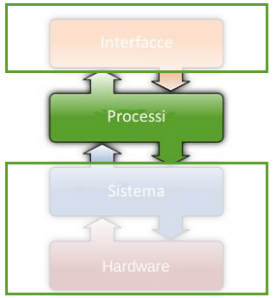
Permettere ai processi di comunicare fra di loro.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

IPC: Signal in Unix



Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

The signals **SIGKILL** and **SIGSTOP** cannot be caught, blocked, or ignored.

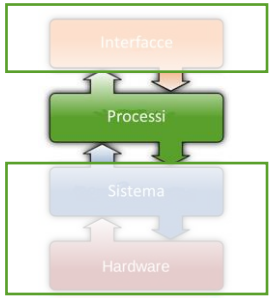
Permettere ai processi di comunicare fra di loro.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

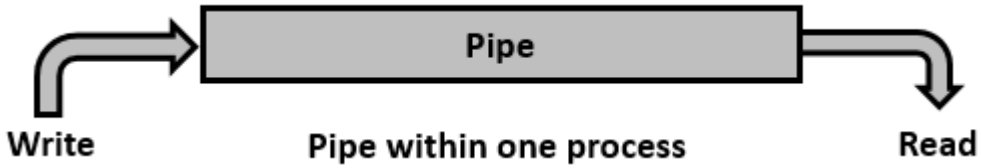
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

IPC: Pipe (datagram FIFO) in Unix

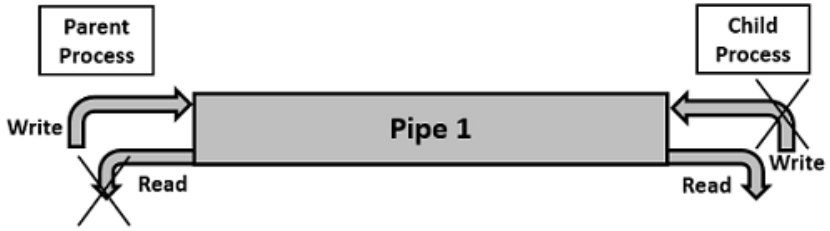


Pipe is one-way communication
Flusso (stream) Unidirezionale

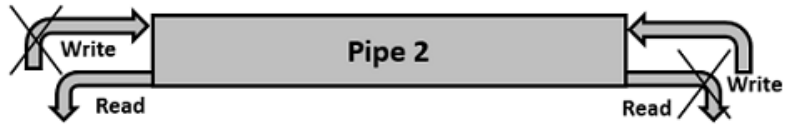



```
rishabh@rishabh:~/GFG$ ls -l | more
total 28
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo1
-rw-rw-r-- 1 rishabh rishabh   26 Jan 25 23:03 demo1.txt
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo2
-rw-rw-r-- 1 rishabh rishabh    0 Jan 25 23:04 demo2.txt
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo3
-rw-rw-r-- 1 rishabh rishabh    0 Jan 25 23:04 demo.txt
-rw-rw-r-- 1 rishabh rishabh  123 Jan 26 16:02 sample1.txt
-rw-rw-r-- 1 rishabh rishabh   44 Jan 26 15:52 sample2.txt
-rw-rw-r-- 1 rishabh rishabh    0 Jan 26 00:12 sample3.txt
-rw-rw-r-- 1 rishabh rishabh   26 Jan 25 23:03 sample.txt
```

Esempio di pipe tra comandi shell



Two-way Communication Using Pipes
(combinando 2 pipe)



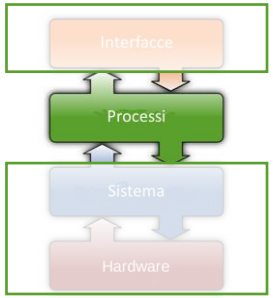
Permettere a 2 processi di scambiarsi informazioni fra loro.

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

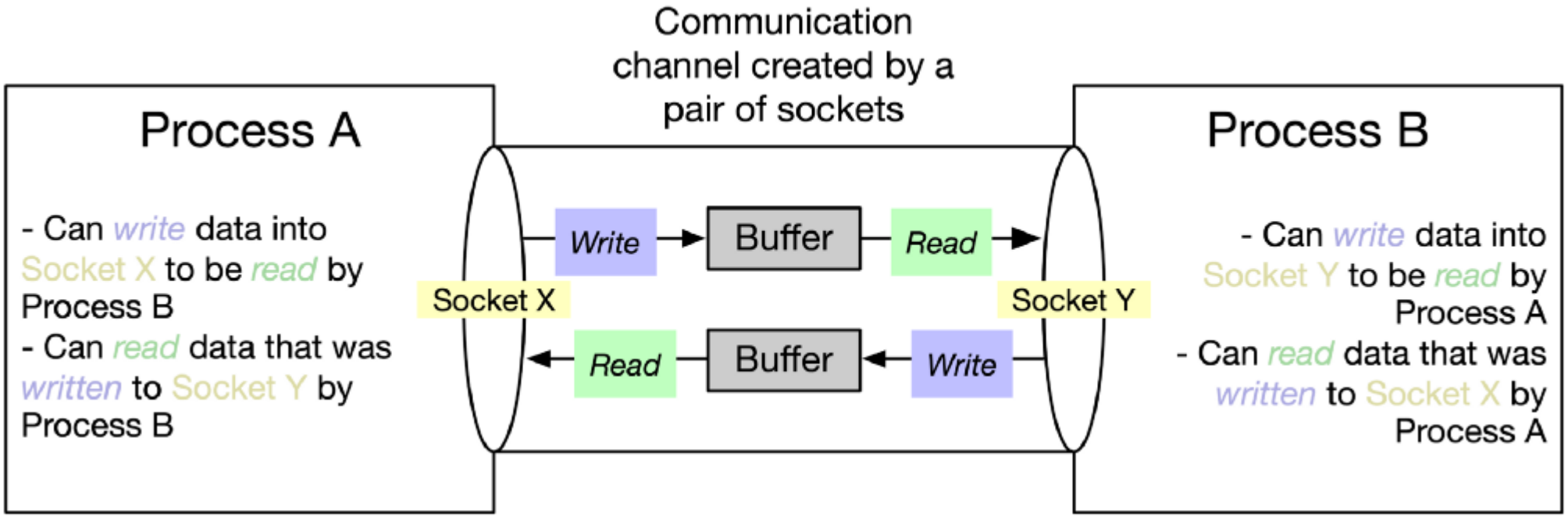
SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

IPC: Socket (stream/datagram) in Unix



One-way communication
Flusso (stream) o blocchi di dati (Datagram) Unidirezionale

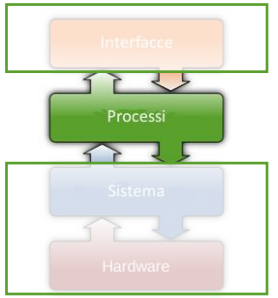


Permettere a 2 processi di scambiarsi informazioni (anche di notevoli dimensioni) fra loro.

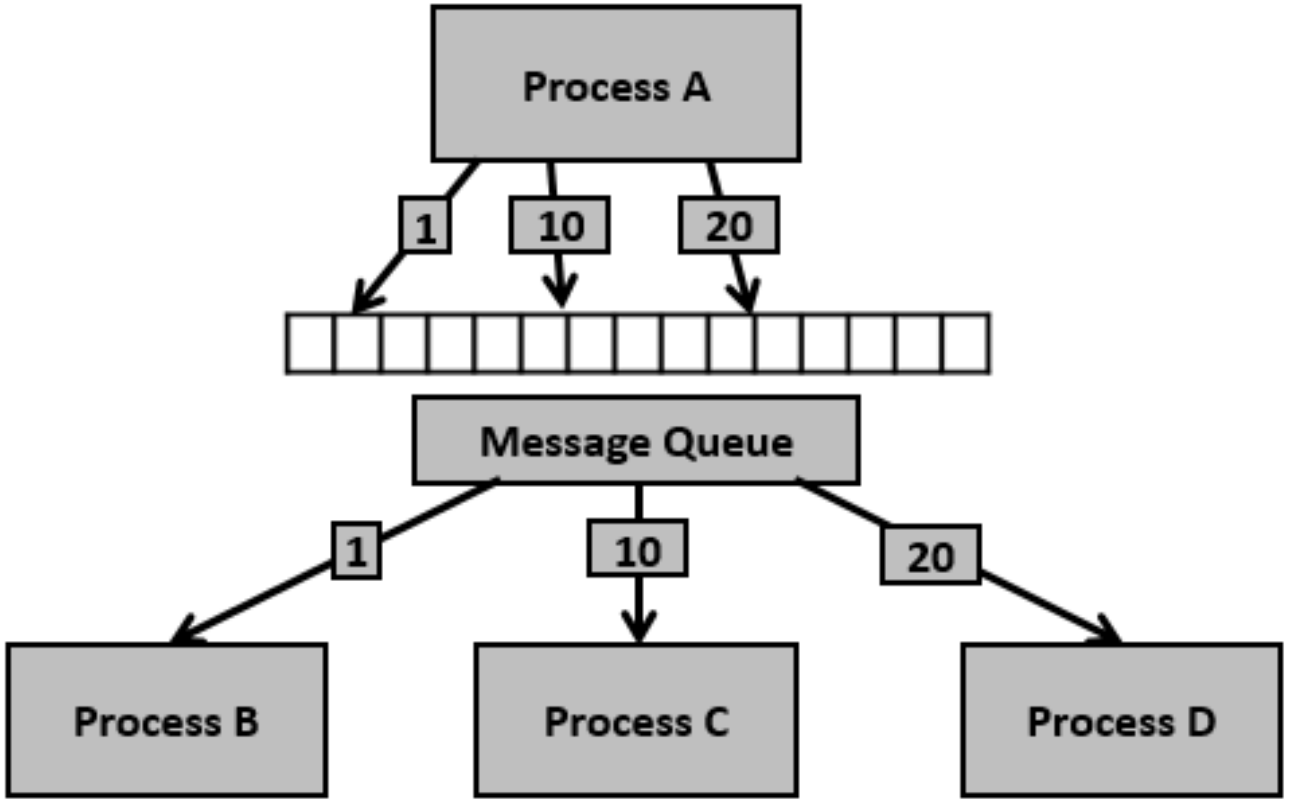
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

IPC: Message Queue in Unix



Multicast communication

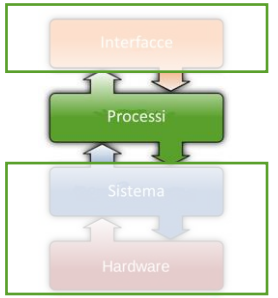


Permettere a molteplici processi di scambiarsi informazioni non riservate.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

IPC: Shared Memory

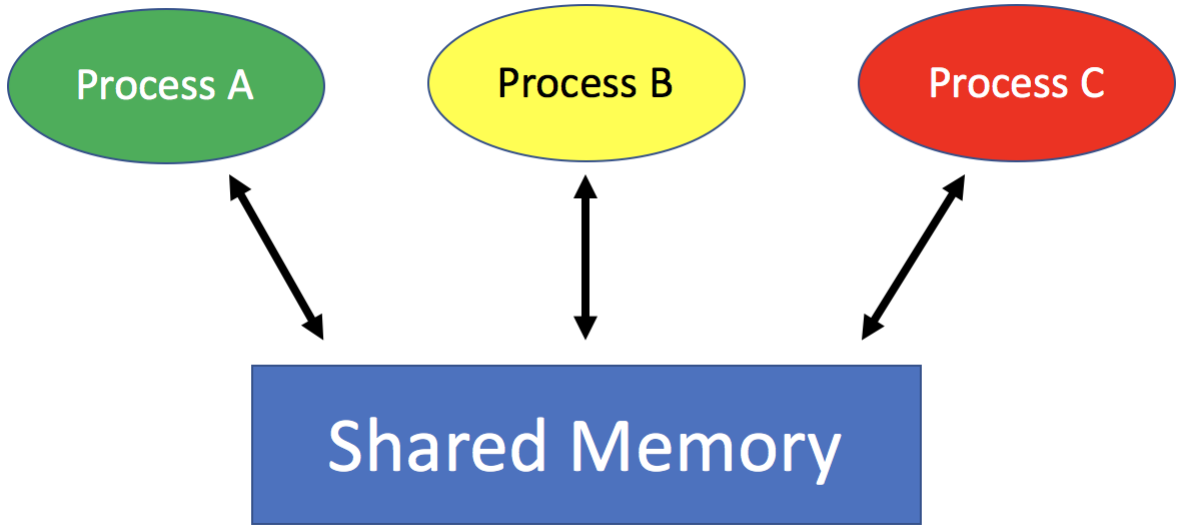


Shared Memory: memoria condivisa fra processi

SHMMAX: massima dimensione (in Byte) di un singolo segmento di memoria condivisa

SHMMNI: numero massimo di segmenti di memoria condivisa presenti sul Sistema

SHMALL: numero totale di pagine di memoria condivisa



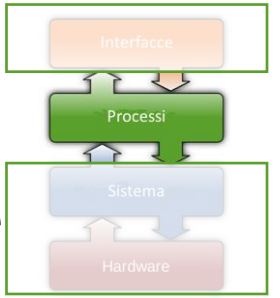
Alloca porzioni di memoria condivisa fra processi.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

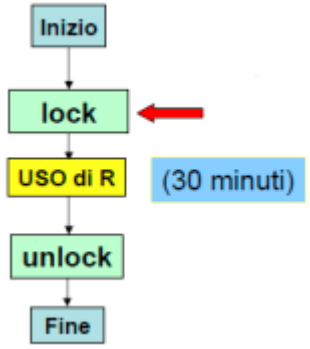
Operating Systems: Obiettivi Funzioni Servizi

IPC: Semaphore in Unix

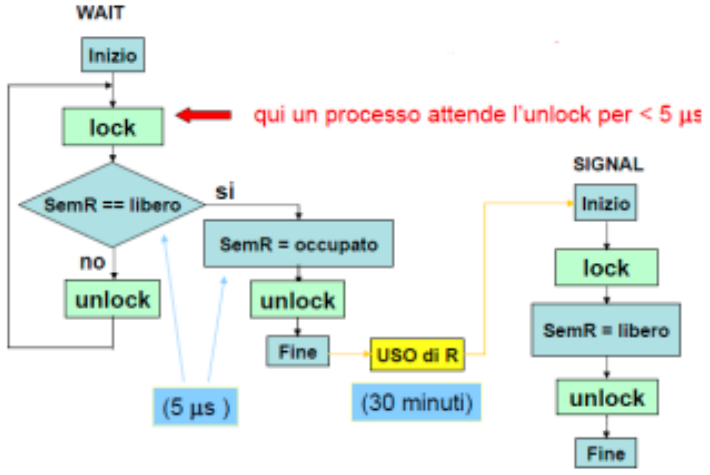


Sezione Critica: operazioni (istruzioni eseguite da un processo) su variabili di stato di una risorsa comune

Busy Waiting: processo esegue NOP in attesa che la risorsa bloccata (lock()) venga liberate dal processo che la usa (unlock())



Semaforo: Meccanismo di gestione di una Sezione Critica, facente uso di wait() (al posto di lock()) e signal() (al posto di unlock()). La risorsa è guardata allora da un semaforo che inizialmente vale 1, per indicare lo stato di risorsa libera. Il valore del semaforo binario è infatti un indicatore che indica se la risorsa è libera (1) o occupata (0)

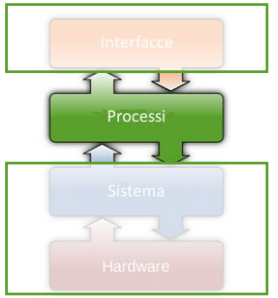


Protegge una risorsa da accessi concorrenti. e risolve in modo efficiente in un ambiente multiprogrammato.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

System Calls



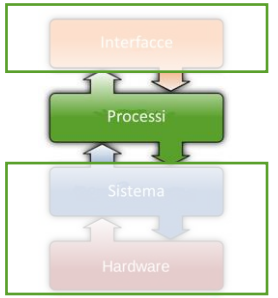
- **Chiamata di sistema**: interazione elementare tra un programma utente e il SO
- Con le chiamate di sistema i programmi utente accedono ai **servizi del SO** disponibili come istruzioni in:
  - linguaggio **assembler**
  - linguaggi di programmazione di più alto livello come **C e C++**
    - permettono alle chiamate di sistema di avvenire direttamente
  - **Java non lo permette**
    - poiché una chiamata di sistema è specifica del SO e dà come risultato un codice specifico della piattaforma
    - tuttavia se richiesto, è possibile attraverso *metodi nativi*
      - Java può richiamare metodi scritti in un altro linguaggio (C o C++) che eseguono la chiamata di sistema

Offrire ai processi funzionalità offerte dal sistema tramite una interfaccia semplice.

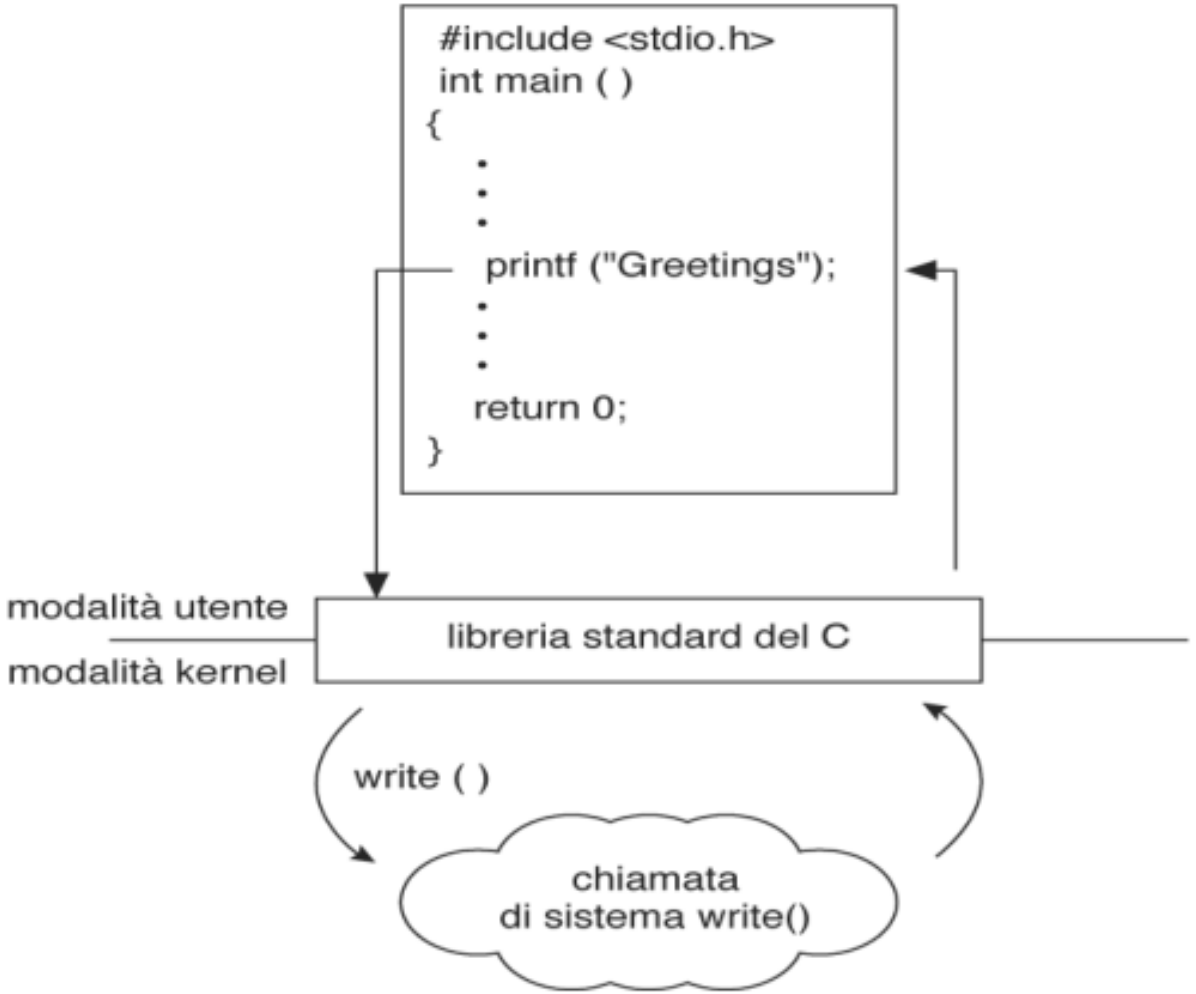
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

System Calls

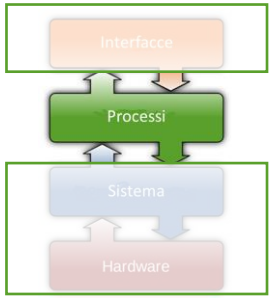


- Un programma C che invoca la chiamata di libreria **printf()**, che a sua volta chiama la funzione di sistema **write()**



Operating Systems: Obiettivi Funzioni Servizi

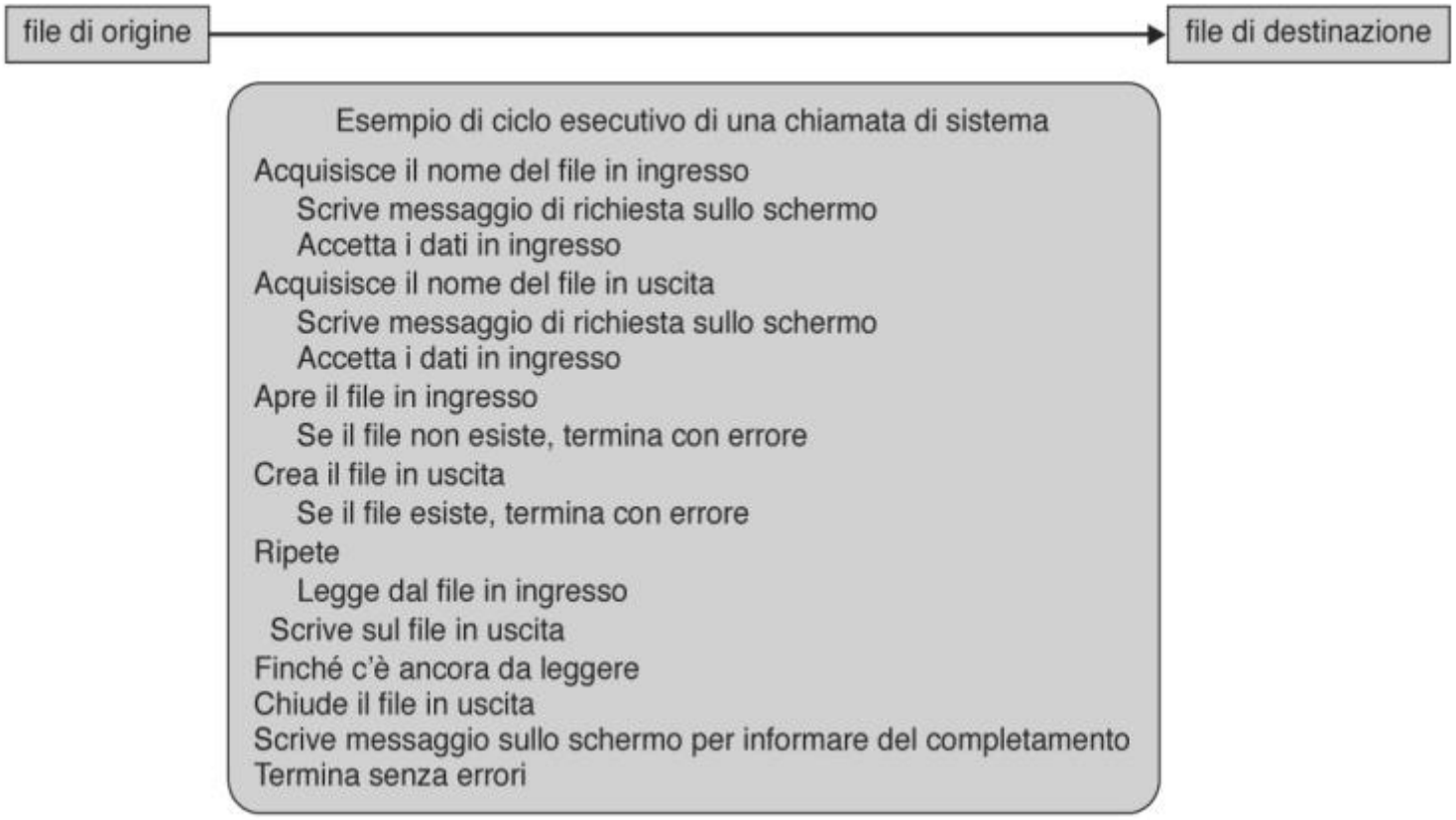
System Calls



Esempio: programma per copiare un file in un altro file

Richiede diverse chiamate di sistema per:

- Ricavare i nomi dei due file
- Tramite richiesta del nome
- Tramite browser dei file (molte più chiamate)
- Aprire il file di lettura e creare il file di scrittura
- Possibili errore: non esiste file input o accesso negato (come risolvo? Avviso l'utente e termino?)
- Possibili errore: esiste già file output (come risolvo? sovrascrivo?)
- Lettura e scrittura
- Possibili errori: end of file, memoria insufficiente per scivere
- Chiusura dei file, avviso utente, terminazione programma

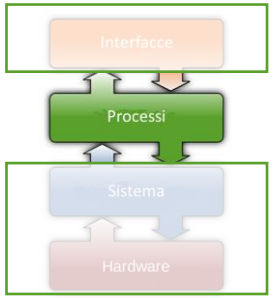


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

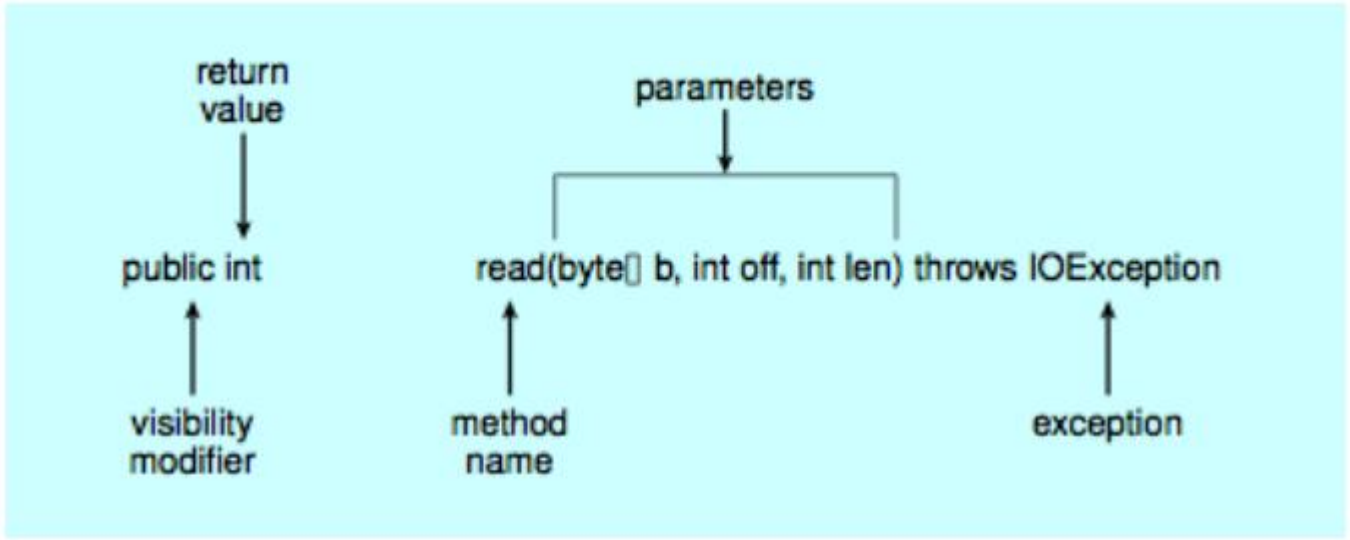
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

System Calls



Il **programmatore** tipicamente **non** invoca le **chiamate di sistema** ma delle funzioni messe a disposizione da un API. Es. metodo Java read()



byte[] b – il buffer di destinazione dei dati letti

int off - offset iniziale in b dove vengono posti i dati

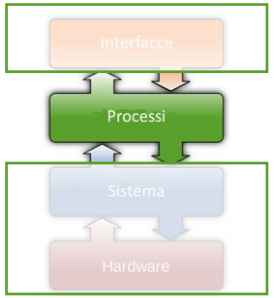
int len – il massimo numero di byte da leggere

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

System Calls; scambio di parametri ed esecuzione



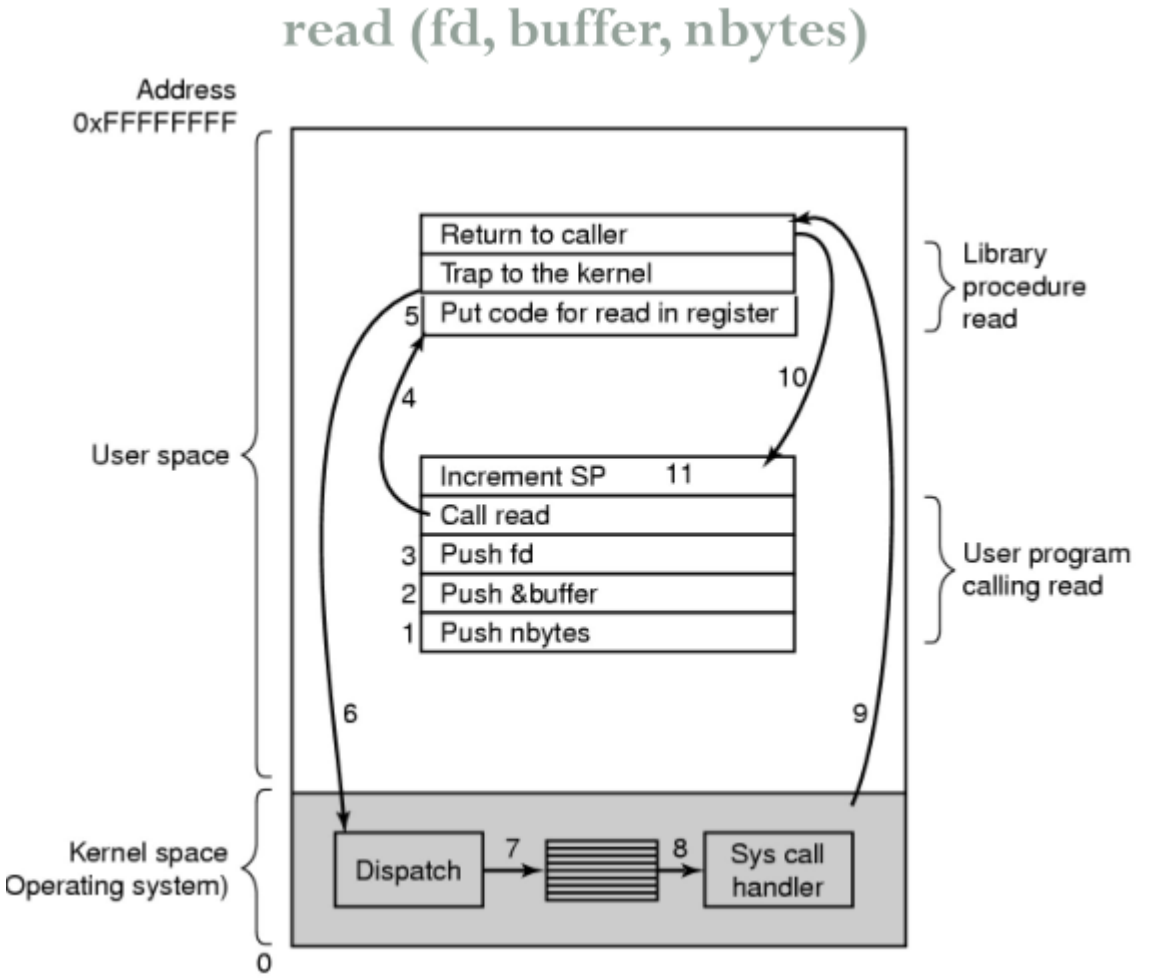
Le chiamate di sistema possono richiedere lo scambio di informazioni (parametri) tra un programma e il SO
Lo scambio dei parametri avviene in generale in 3 modi:

1. Attraverso registry: attuabile solo se il numero dei parametri non supera quello dei registry

2. Attraverso una memoria provvisoria detta pila (stack) posti dal programma e prelevati dal SO

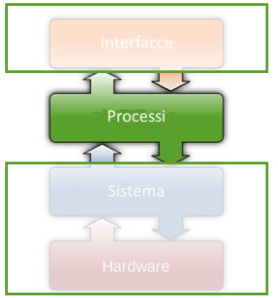
(non limitano il numero o la lunghezza dei parametri)

3. Attraverso una tabella in memoria, e l'indirizzo X della tabella viene passato come parametro in un registro



Operating Systems: Obiettivi Funzioni Servizi

System Calls: esecuzione



Dettaglio dell'esecuzione di `read (fd, buffer, nbytes)`
1-3. Salvataggio dei parametri sullo stack

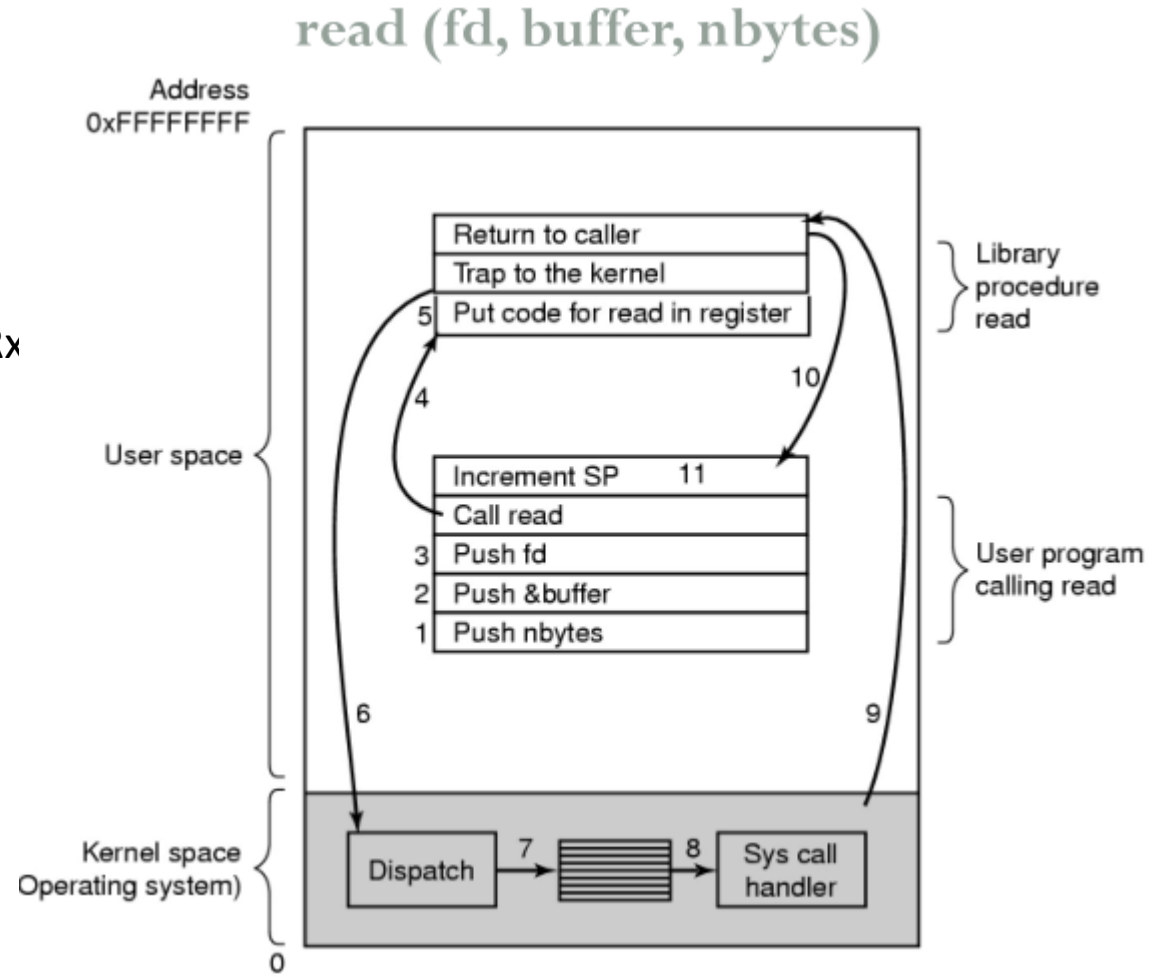
4. Chiamata della funzione di libreria read

5. Caricamento del codice della system call in un registro fissato Rx

6. Esecuzione TRAP
- passaggio in kernel mode, salto al codice del dispatcher

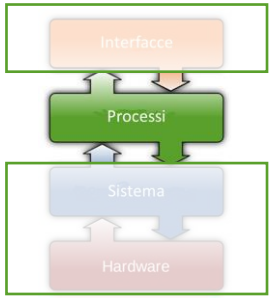
7-8. Selezione della system call secondo il codice in Rx ed invocazione del gestore appropriato
9. Ritorno alla funzione di libreria
- ripristino user mode, caricamento del program counter PC

10-11. Ritorno al codice utente (nel modo usuale) ed incremento dello stack pointer SP per rimuovere I parametri della chiamata a read

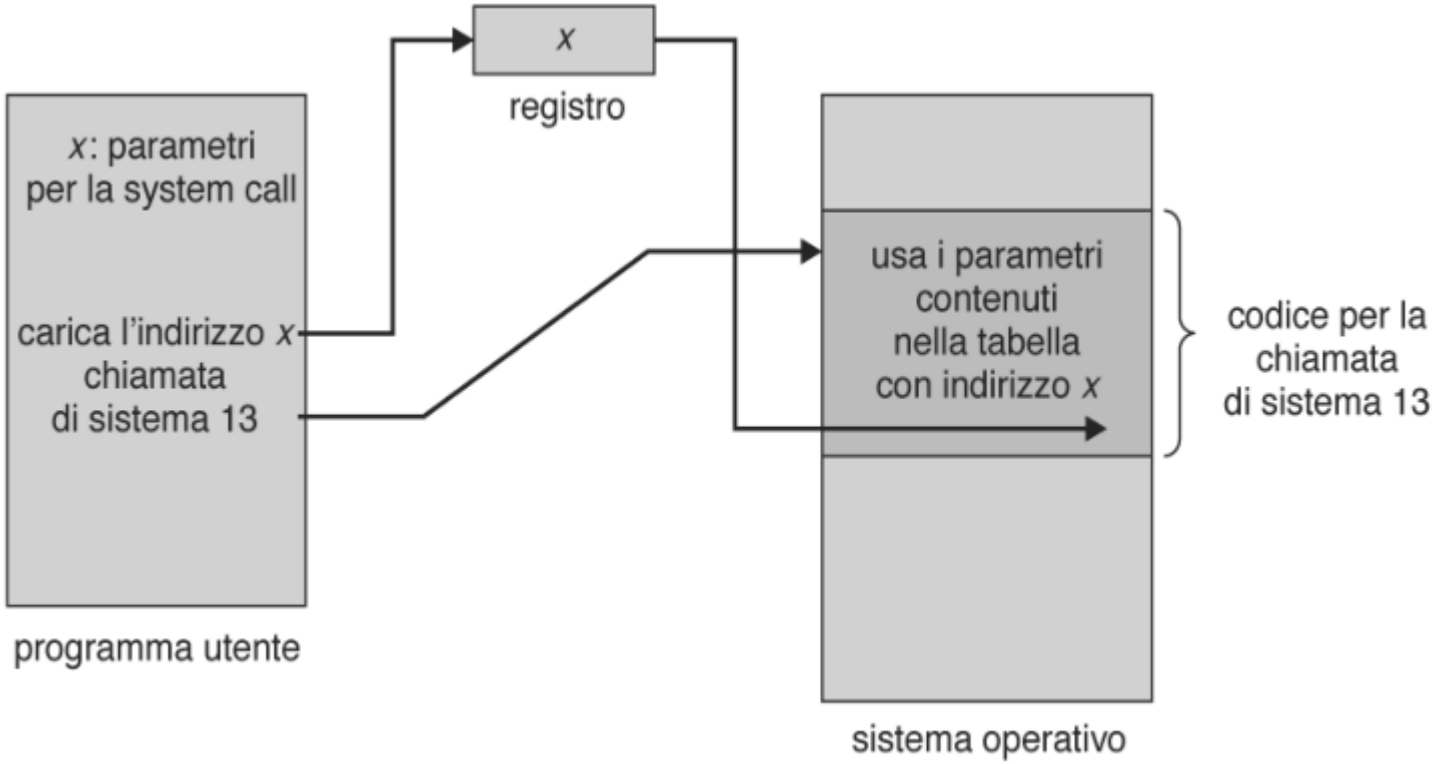


Operating Systems: Obiettivi Funzioni Servizi

System Calls: esecuzione



3. Attraverso una tabella in memoria, e l'indirizzo X della tabella viene passato come parametro in un registro

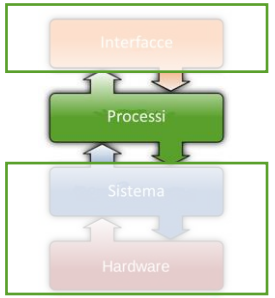


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

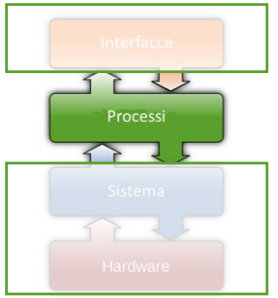
System Calls: classificazione



- Controllo dei processi
  - terminazione normale e anormale
  - caricamento, esecuzione
  - creazione e arresto di un processo
  - esame e impostazione degli attributi di un processo
  - attesa per il tempo indicato
  - attesa e segnalazione di un evento
  - assegnazione e rilascio di memoria
- Gestione dei file
  - creazione e cancellazione di file
  - apertura, chiusura
  - lettura, scrittura, posizionamento
  - esame e impostazione degli attributi di un file
- Gestione dei dispositivi
  - richiesta e rilascio di un dispositivo
  - lettura, scrittura, posizionamento
  - esame e impostazione degli attributi di un dispositivo
  - inserimento logico ed esclusione logica di un dispositivo
- Gestione delle informazioni
  - esame e impostazione dell'ora e della data
  - esame e impostazione dei dati del sistema
  - esame e impostazione degli attributi dei processi, file e dispositivi
- Comunicazione
  - creazione e chiusura di una connessione
  - invio e ricezione di messaggi
  - informazioni sullo stato di un trasferimento
  - inserimento ed esclusione di dispositivi remoti

Operating Systems: Obiettivi Funzioni Servizi

System Calls: API di Unix



Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

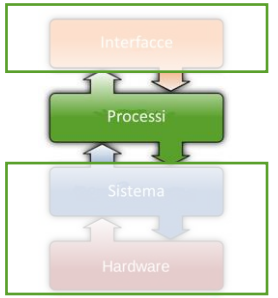
Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

System Calls: API di Windows

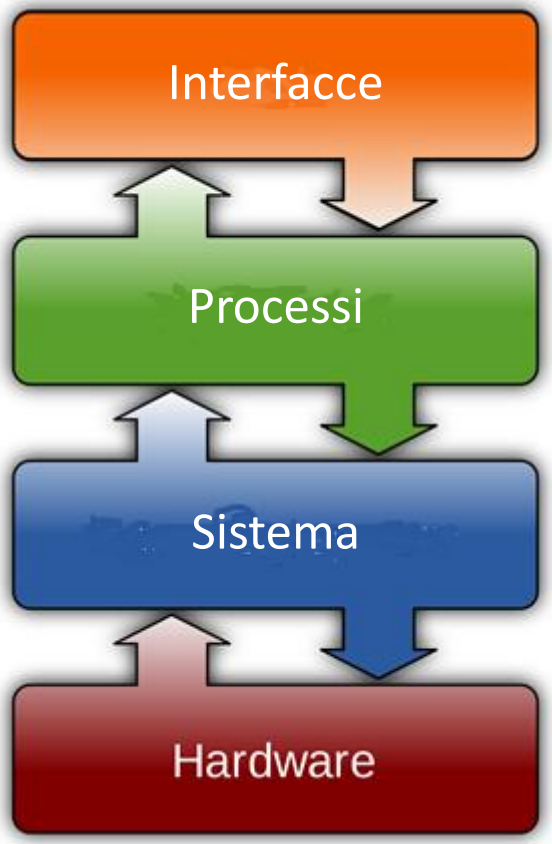


UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

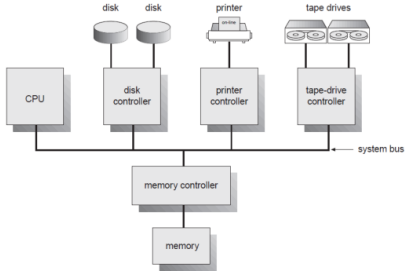
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

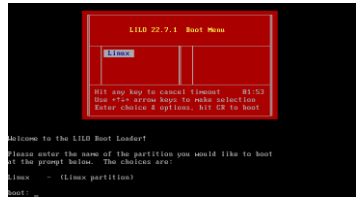
Operating Systems: Introduzione ai Layers



Obiettivi	Funzioni	Servizi
Astrazione	File System Shell/GUI/Transaction	Authentication/Authorization Accounting
Virtualizzazione	Process Management Memory Management	IPC System Calls
Input/Output	Error Detection Dual-Mode	Boot Interrupt Handling



	B	C
	Formulas	Formula Results
	=1/0	#DIV/0!
	=A2/A3	#DIV/0!
	=QUOTIENT(A2,A3)	#DIV/0!

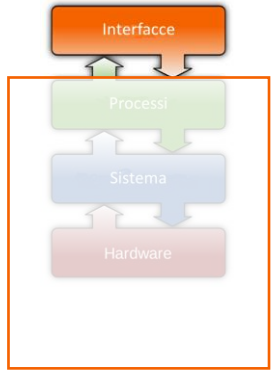


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

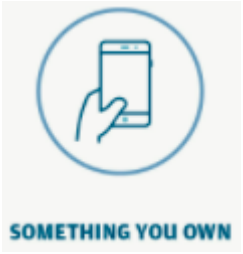
Operating Systems: Obiettivi Funzioni Servizi

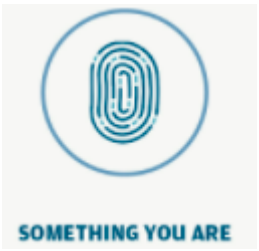
Identification/Authentication/Authorization/Accounting



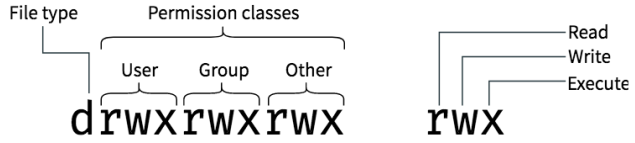
1. Identification

2. Authentication





3. Authorization



Resource

4. Accountability

```
# cat /var/log/secure | grep "authentication failure"

Nov 12 12:20:38 ip-192-168-2-150 su: pam_unix(su-l:auth):
authentication failure; logname=ec2-user uid=1000 euid=0 tty=pts/0
ruser=ec2-user rhost=  user=mytestuser

Nov 12 12:20:53 ip-192-168-2-150 su: pam_unix(su-l:auth):
authentication failure; logname=ec2-user uid=1000 euid=0 tty=pts/0
ruser=ec2-user rhost=  user=mytestuser

Nov 12 12:21:07 ip-192-168-2-150 su: pam_unix(su-l:auth):
authentication failure; logname=ec2-user uid=1000 euid=0 tty=pts/0
ruser=ec2-user rhost=  user=mytestuser
```

Consentire l'accesso al sistema ed alle sue risorse solo agli utenti che ne hanno diritto.

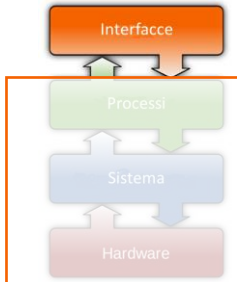
DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Obiettivi Funzioni Servizi

User Authentication/Authorization

/etc/passwd ha tipicamente permessi di file system che gli permettono di essere leggibile da tutti gli utenti del sistema (world-readable), anche se può essere modificato solo dal superutente o utilizzando alcuni comandi privilegiati a scopo speciale



nish:x:501:501:Nishant Rastogi:/home/nish:/bin/bash

1 2 3 4 5 6 7

1. Username (login name)
2. Password («x» indica che è nel file /etc/shadow)
3. UID (User Identifier)
4. GID (Group Identifier)
5. General Electric Comprehensive Operating Supervisor (Nome complete, Edificio, stanza, telefono d'ufficio, telefono di casa, indirizzo email, etc)
6. Home directory
7. Shell di avvio ad ogni nuova connessione

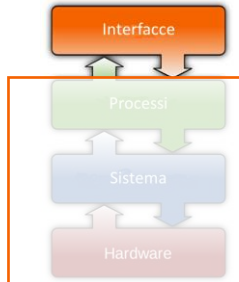
Consentire l'accesso al sistema solo agli utenti che ne hanno diritto.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

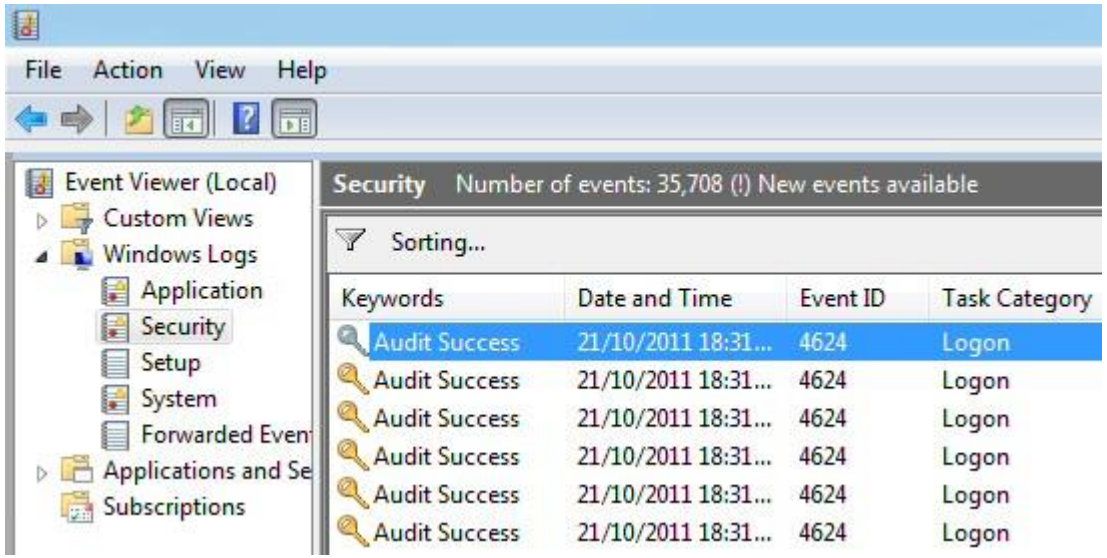
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

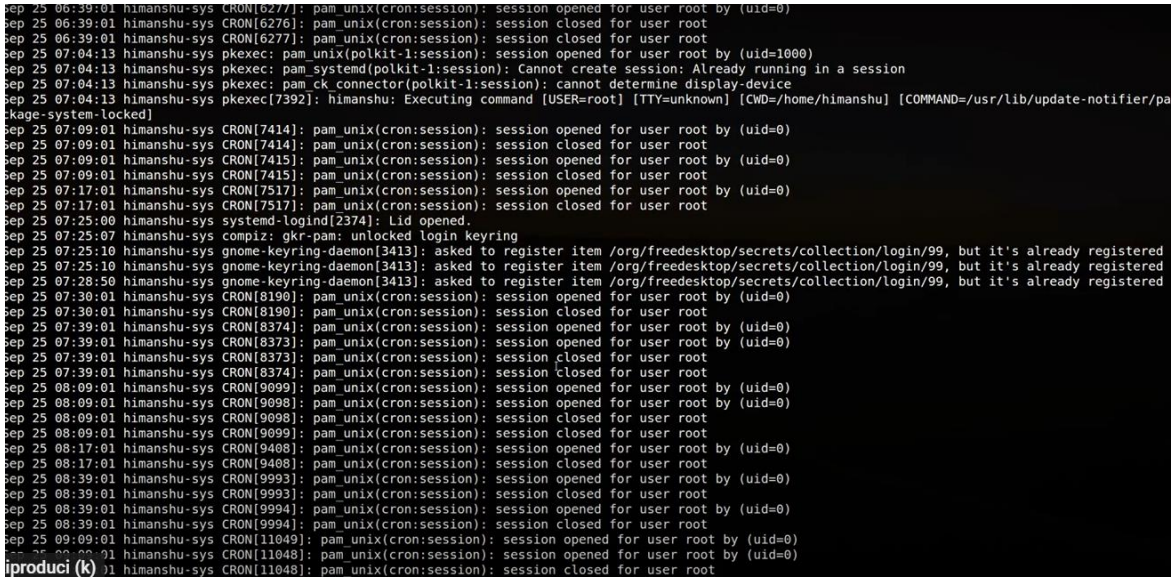
Accounting



Windows Security Log



Linux /var/log/auth.log



File di rendicontazione: /var/log/
Utmp, btmp, wtmp

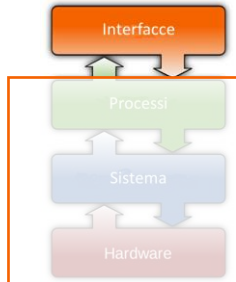
Registrare gli accessi al sistema e poterli rendicontare (se necessario).

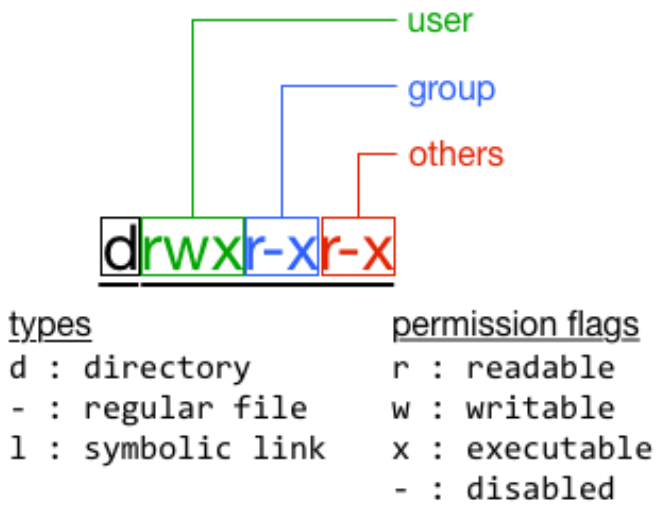
Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

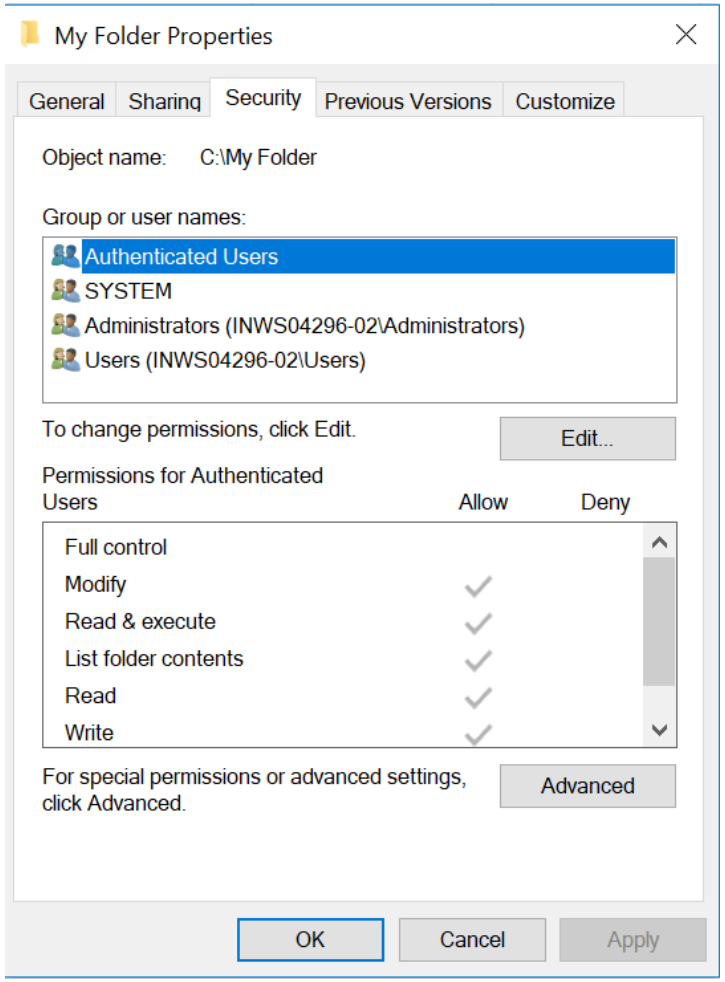
Sapienza Università di Roma

Operating Systems: Obiettivi Funzioni Servizi

File Authorization







Consentire l'accesso al sistema ed alle sue risorse solo agli utenti che ne hanno diritto.