

Laboratorio di Informatica di Base

Laurea in Informatica

Docente: *Massimo Merro*
profs.sci.univr.it/~merro

Lucidi a cura di
Andrea Colombari, Carlo Drioli, Andrea Fusiello e Barbara
Oliboni

Lezione 1

L'elaboratore elettronico

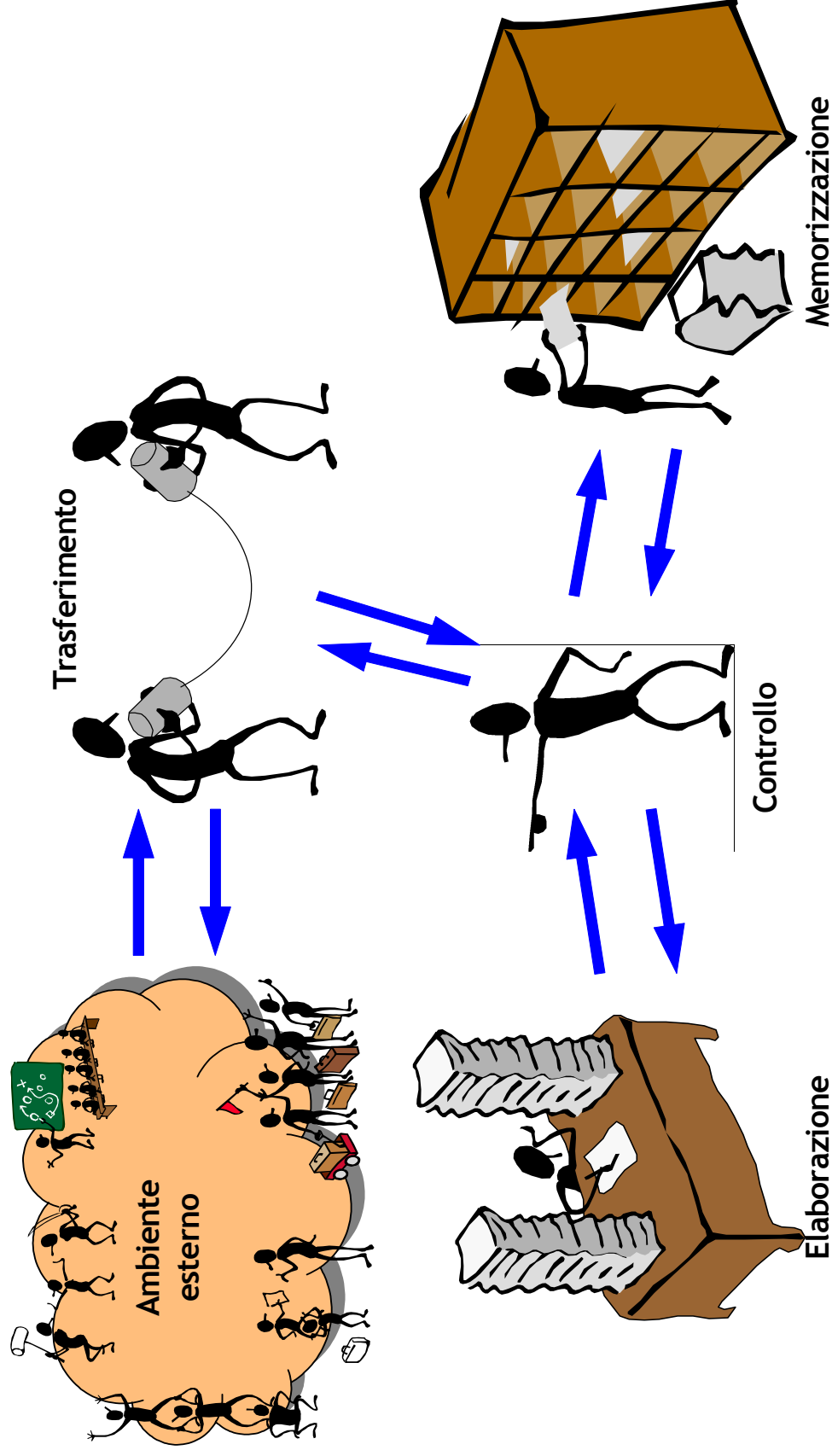
Materiale tratto dai lucidi ufficiali a corredo del testo:

D. Sciuto, G. Buonanno e L. Mari
“Introduzione ai sistemi informatici”

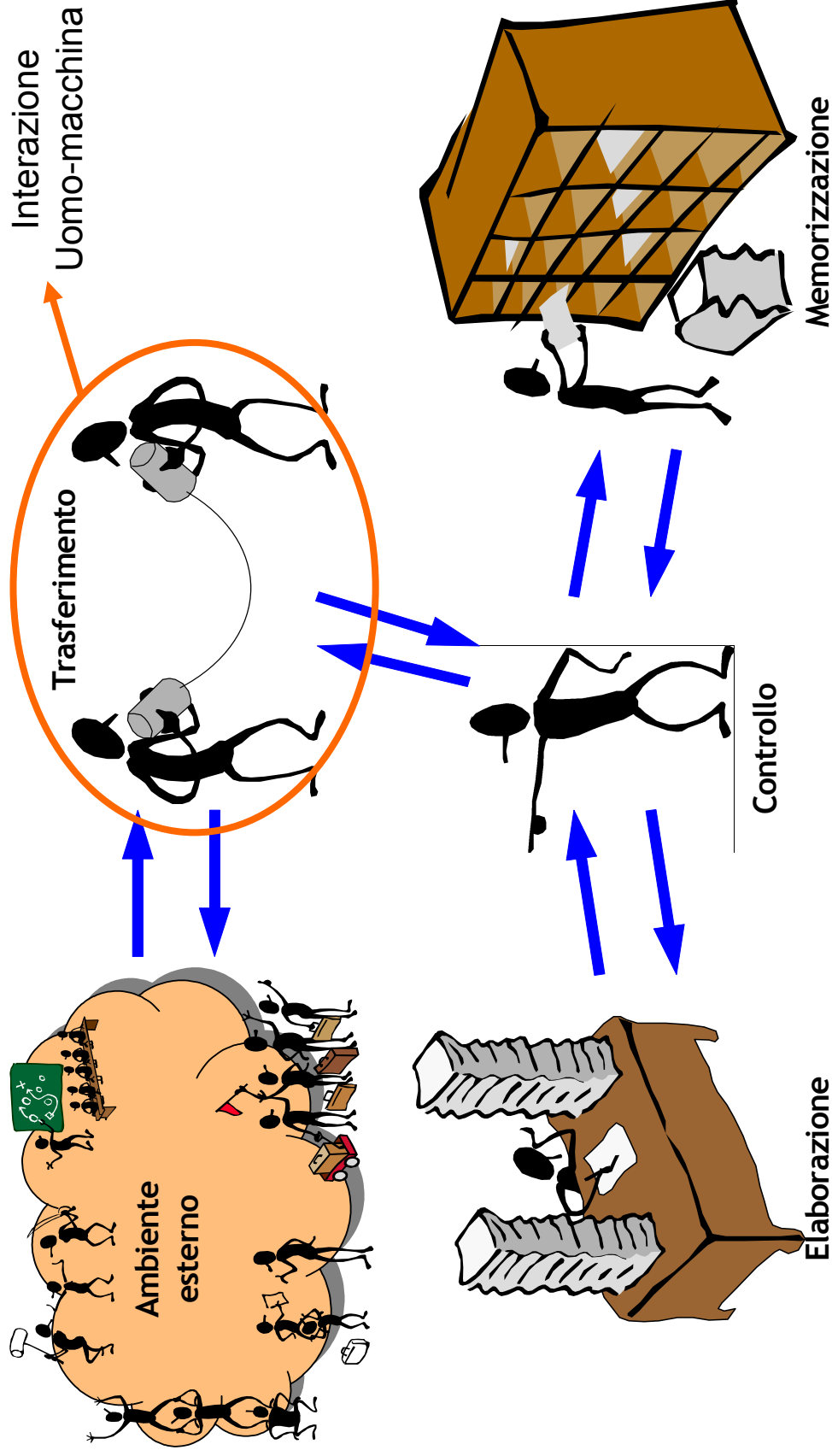
2005 - McGrawHill



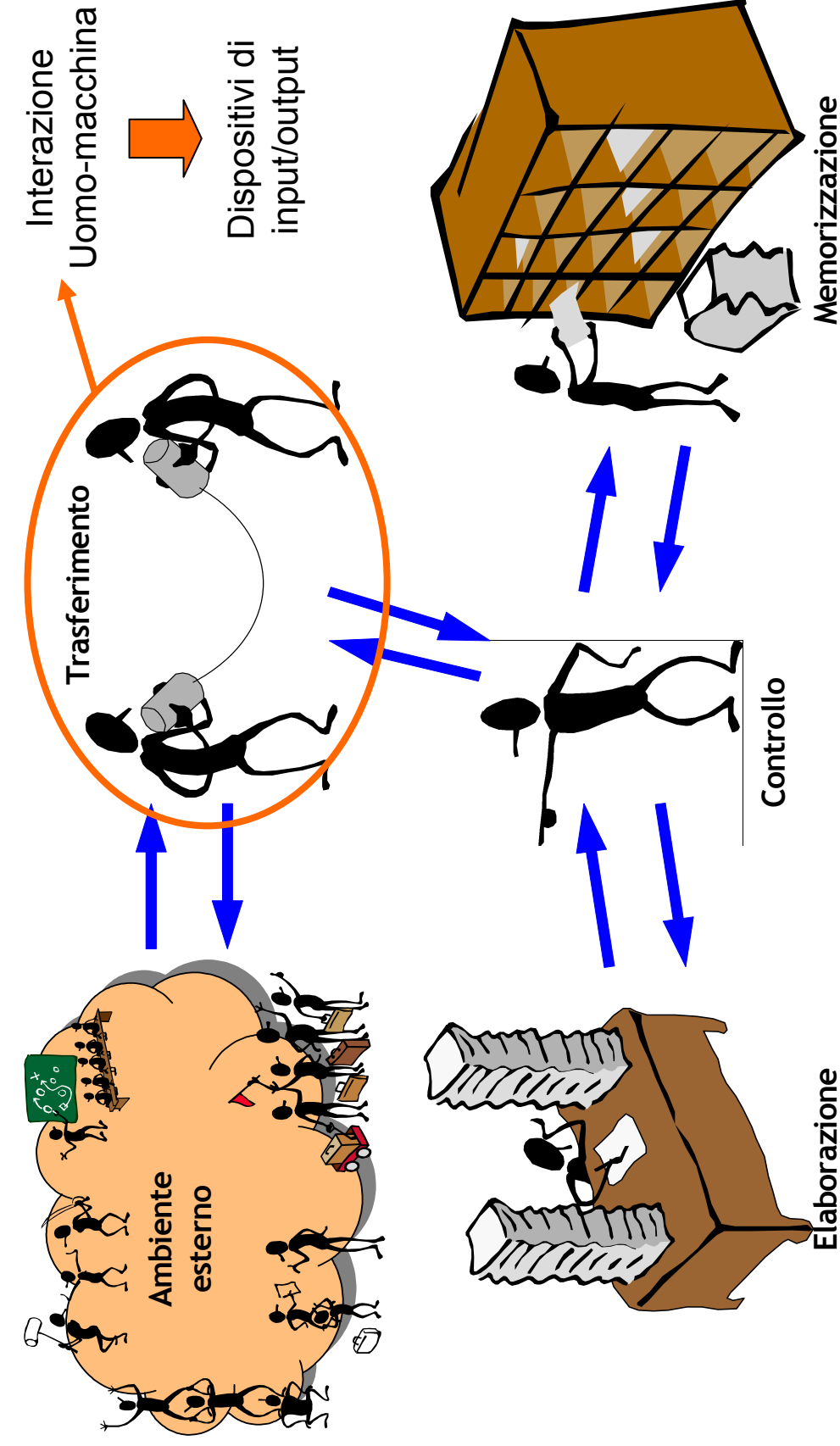
L'elaboratore elettronico



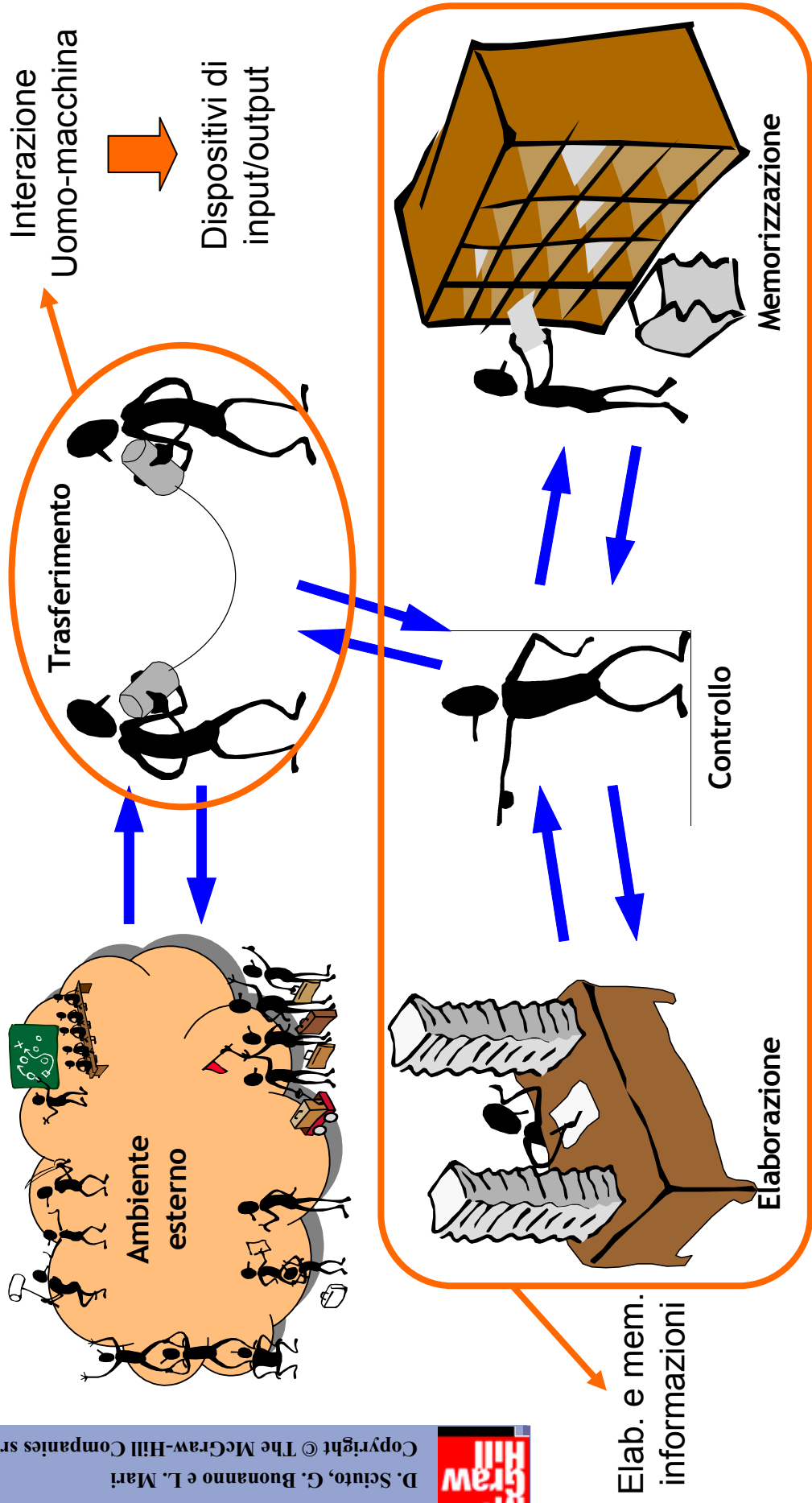
L'elaboratore elettronico



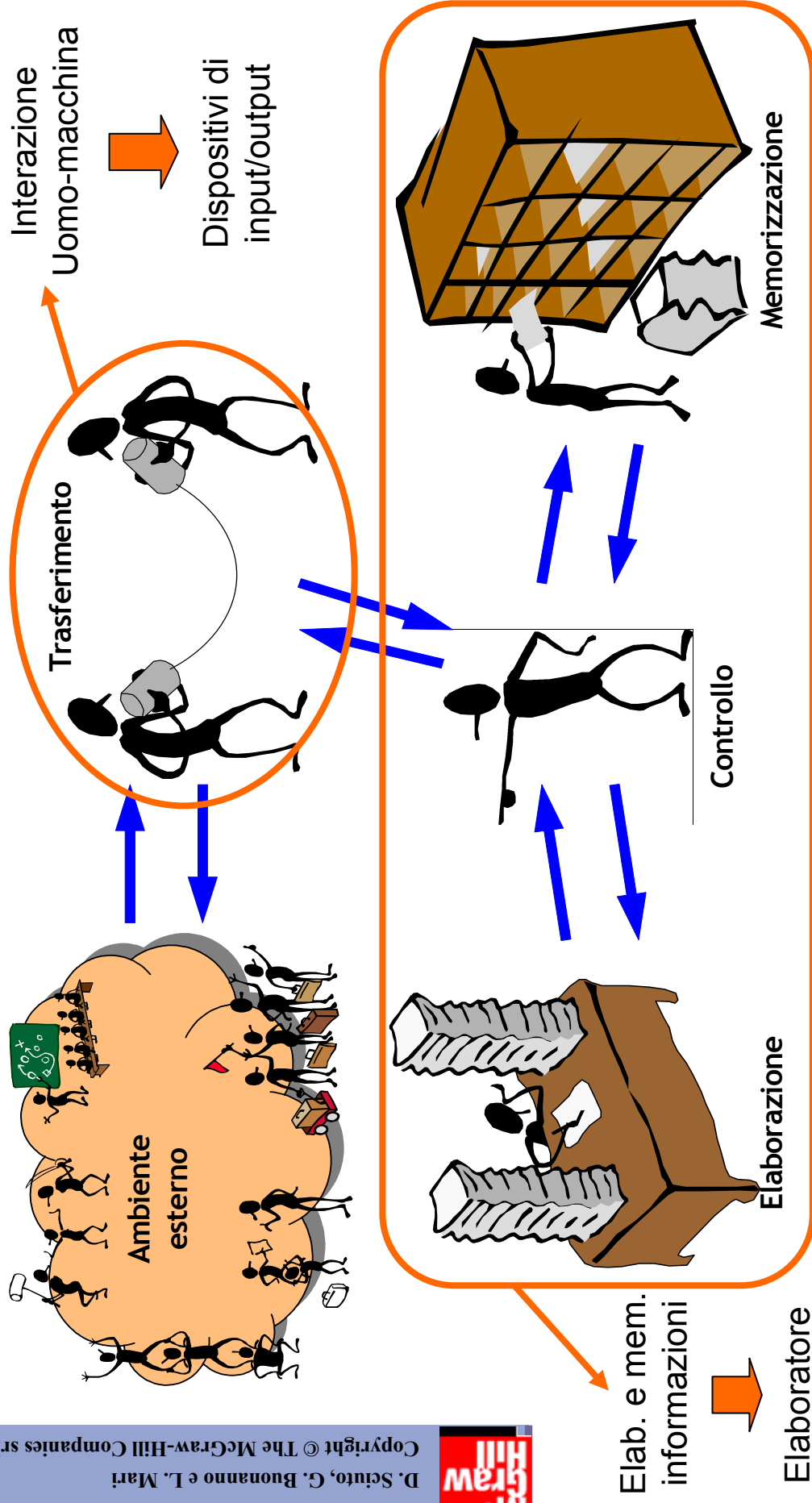
L'elaboratore elettronico



L'elaboratore elettronico



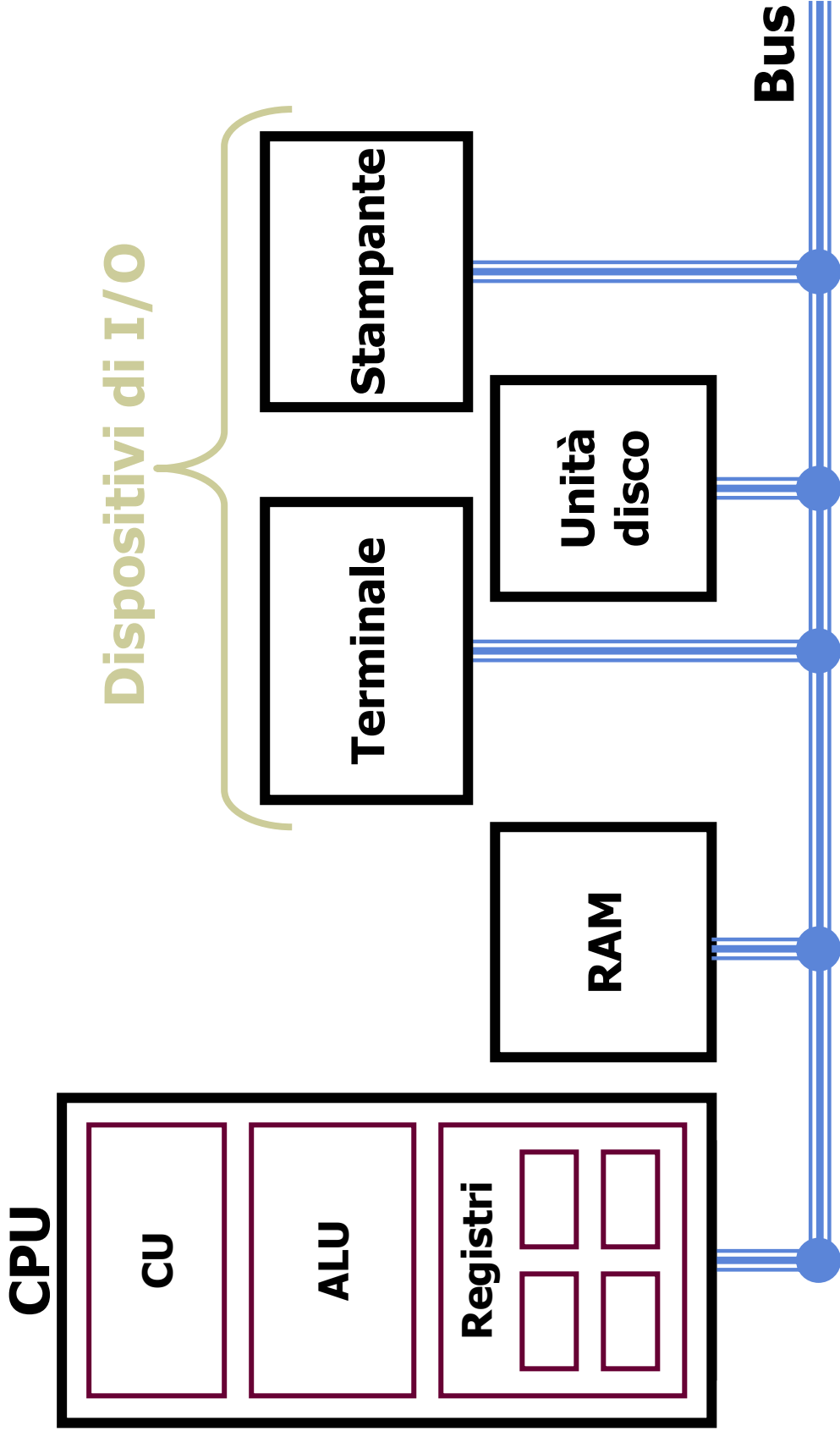
L'elaboratore elettronico



L'elaboratore elettronico (2)

- Un *elaboratore elettronico* deve essere in grado di:
 - eseguire istruzioni su dati e controllare il flusso dell'esecuzione
 - **CPU**, Central Processing Unit:
 - Unità logico aritmetica (**ALU**, Arithmetic Logic Unit)
 - Unità di controllo (**CU**, Control Unit)
 - **Registri**
 - memorizzare le istruzioni e i dati su cui esse operano
 - memoria centrale (**RAM**, Random Access Memory) e **unità disco** (o memoria di massa)
 - interagire con gli utenti e con eventuali altri sistemi
 - dispositivi di ingresso/uscita, i.e. input/output (**I/O**)

L'architettura di Von Neumann



L'architettura di Von Neumann

CPU

Controllo del flusso di esecuzione.

Esecuzione delle istruzioni

Registri



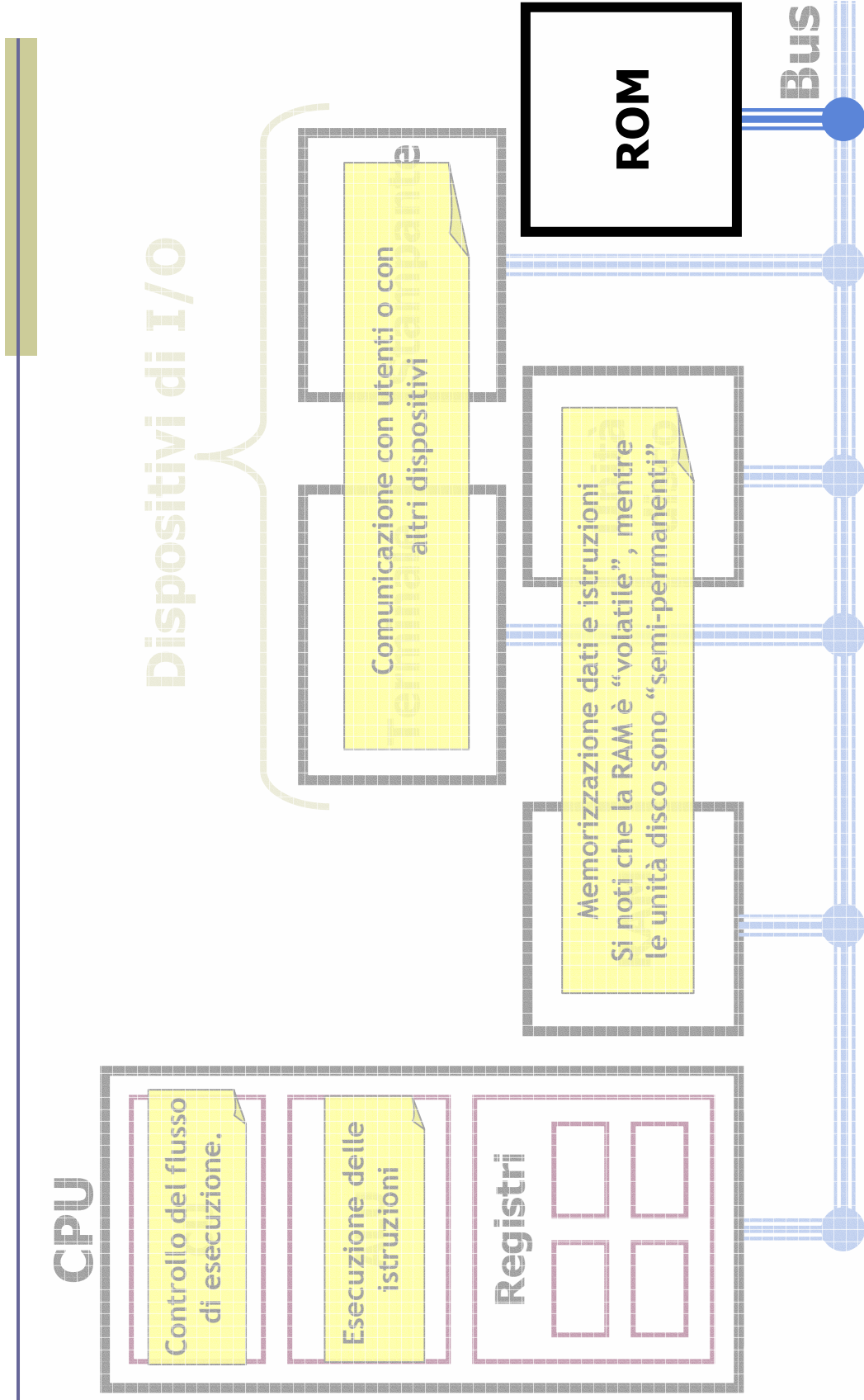
Dispositivi di I/O

Comunicazione con utenti o con altri dispositivi

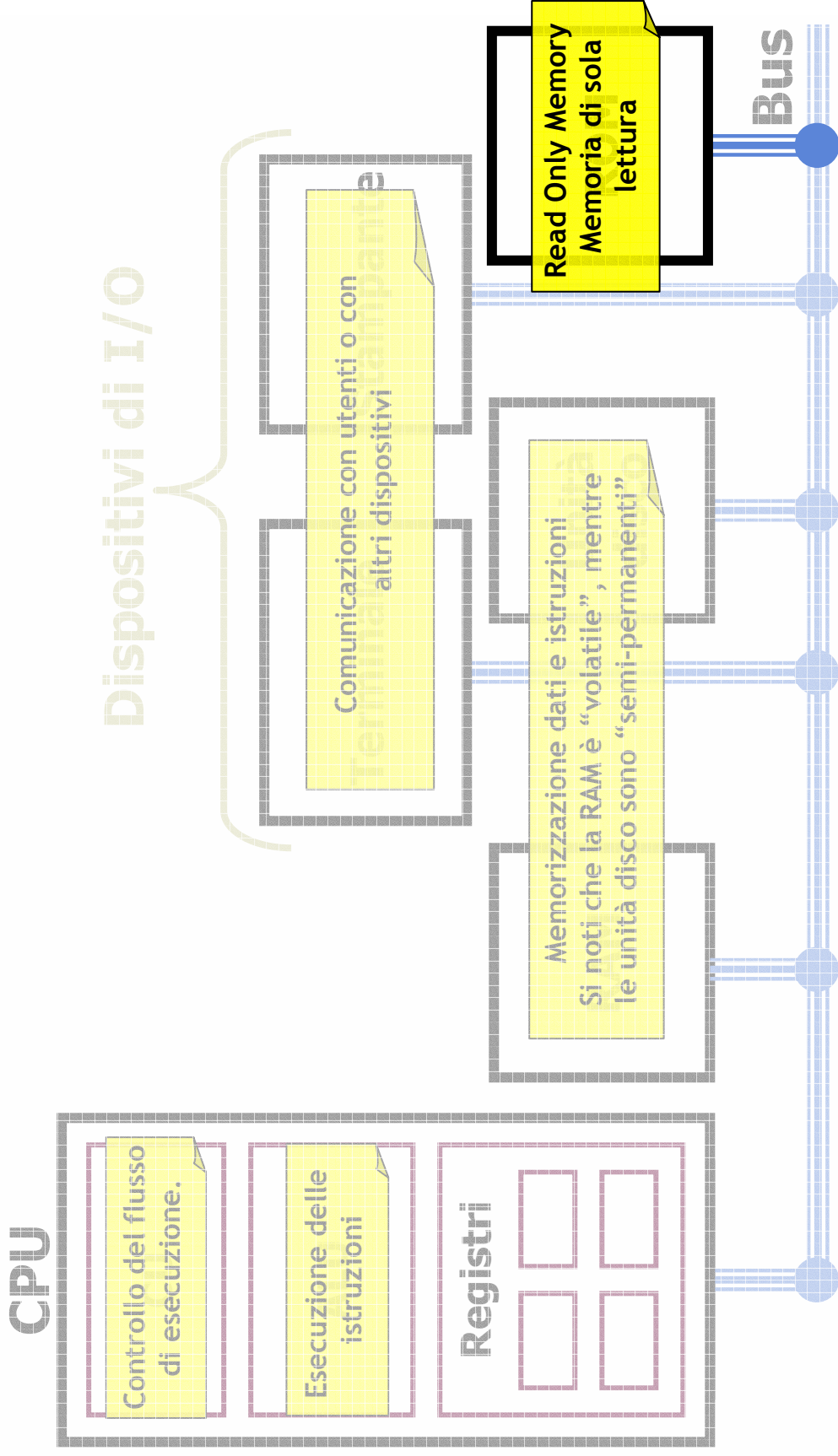
Memorizzazione dati e istruzioni
Si noti che la RAM è "volatile", mentre le unità disco sono "semi-permanenti"

Bus

L'architettura di Von Neumann



L'architettura di Von Neumann



Il bootstrap

- La fase di accensione di calcolatore è detta *bootstrap*
- Un calcolatore è fatto per eseguire delle istruzioni che si trovano in memoria centrale
 - PROBLEMA:
All'accensione la RAM è vuota!
 - SOLUZIONE:
Utilizzo di una ROM in cui è “fuso” un programma di bootstrap che viene eseguito all'accensione
- Il programma di bootstrap ha il compito di caricare in memoria centrale il **Sistema Operativo (S.O.)** a partire da un'unità disco

Il Sistema Operativo

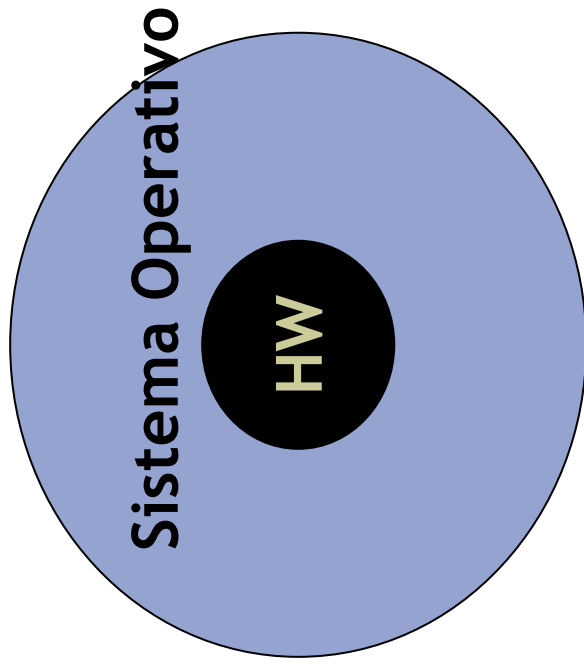


Materiale tratto dai lucidi ufficiali a corredo del testo:

D. Sciuto, G. Buonanno e L. Mari
“Introduzione ai sistemi informatici”

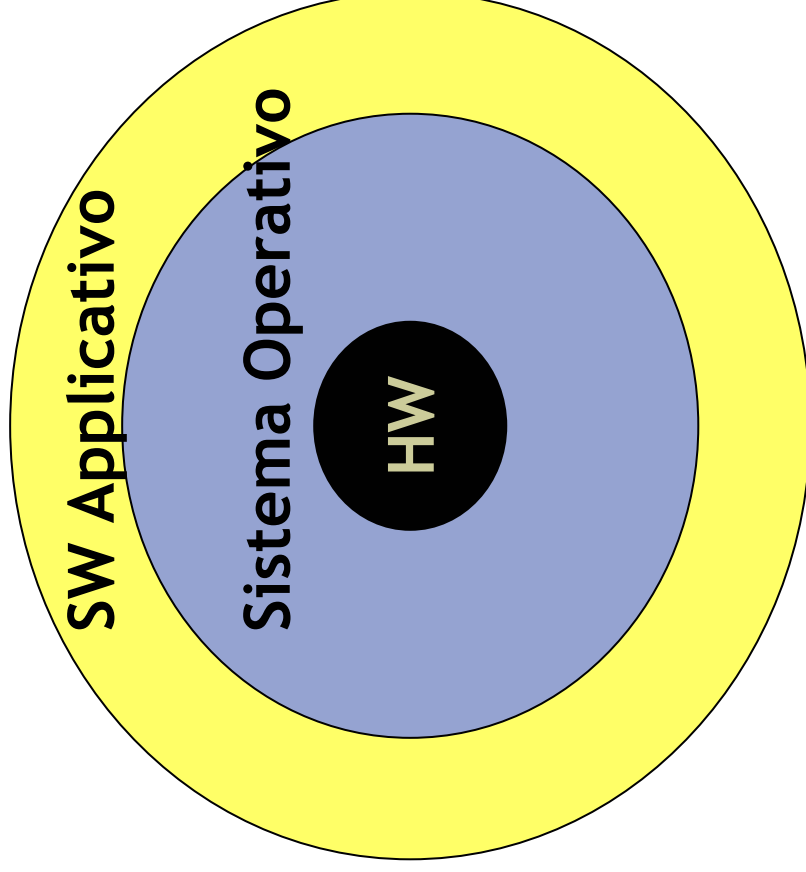
2005 - McGrawHill

Il sistema operativo



- Il S.O. come necessario **intermediario**

Il sistema operativo

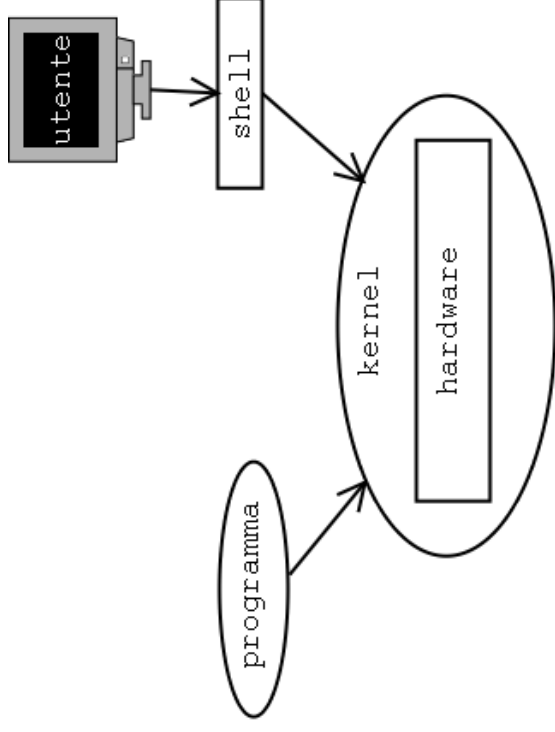


- Il S.O. come necessario **intermediario**
- SW = Sistema Operativo + SW applicativo

Il sistema operativo

■ Il sistema operativo **fornisce dei servizi** ai programmi applicativi e agli utenti rendendo utilizzabili le **risorse fisiche** presenti nel sistema di calcolo.

■ Il sistema operativo può essere inteso come uno strumento che **virtualizza** le caratteristiche dell'hardware sottostante, offrendo di esso la visione di una **macchina astratta** più potente e più semplice da utilizzare di quella fisicamente disponibile.



Il sistema operativo: funzioni

- Il S.O. deve svolgere delle funzioni di base:
 - Esecuzione di applicazioni
 - Accesso ai dispositivi di ingresso/uscita
 - Archiviazione di dati e programmi
 - Controllo di accesso
 - Gestione di tempi e costi di utilizzo (contabilizzazione)
 - Gestione dei malfunzionamenti

Il sistema operativo: elementi

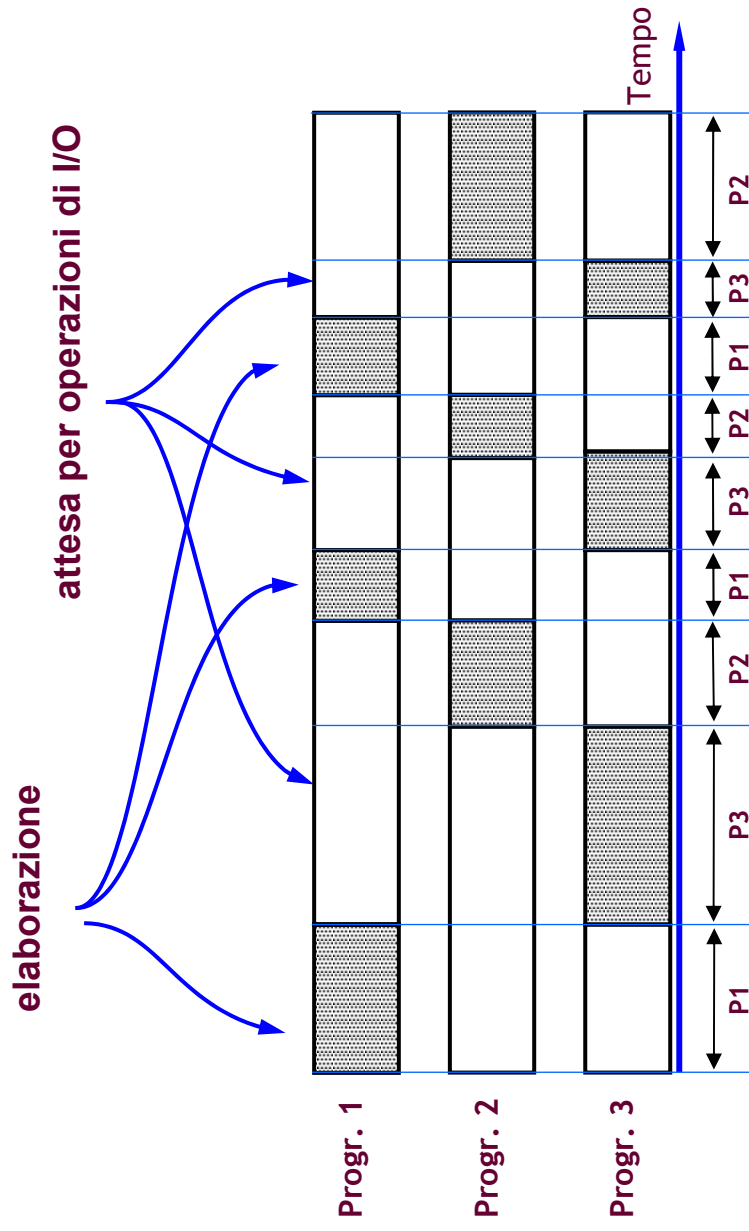
- Un S.O. tipico è costituito dai seguenti elementi:
 - Sistema di gestione del **processore**
 - Sistema di gestione della **memoria**
 - Sistema di gestione delle **periferiche (I/O)**
 - Sistema di gestione dei file (**file system**)
 - Sistema di gestione degli utenti e dei relativi comandi (**interprete comandi o shell**)
 - Sistema di gestione della **rete**.

Il sistema operativo: tipologie

- L'evoluzione dei sistemi di calcolo ha visto succedersi varie tipologie di S.O.:
 - Sistemi batch (esecuzione non interattiva di task)
 - Sistemi interattivi (interazione con l'utente)
 - Sistemi uniprogrammati e monoutente (1 solo l'utente, 1 solo programma utente per volta in esecuzione)
 - Sistemi time-sharing: interattivi, multiprogrammati e multiutente (più utenti e più programmi utente in esecuzione)

Il sistema operativo: tipologie (2)

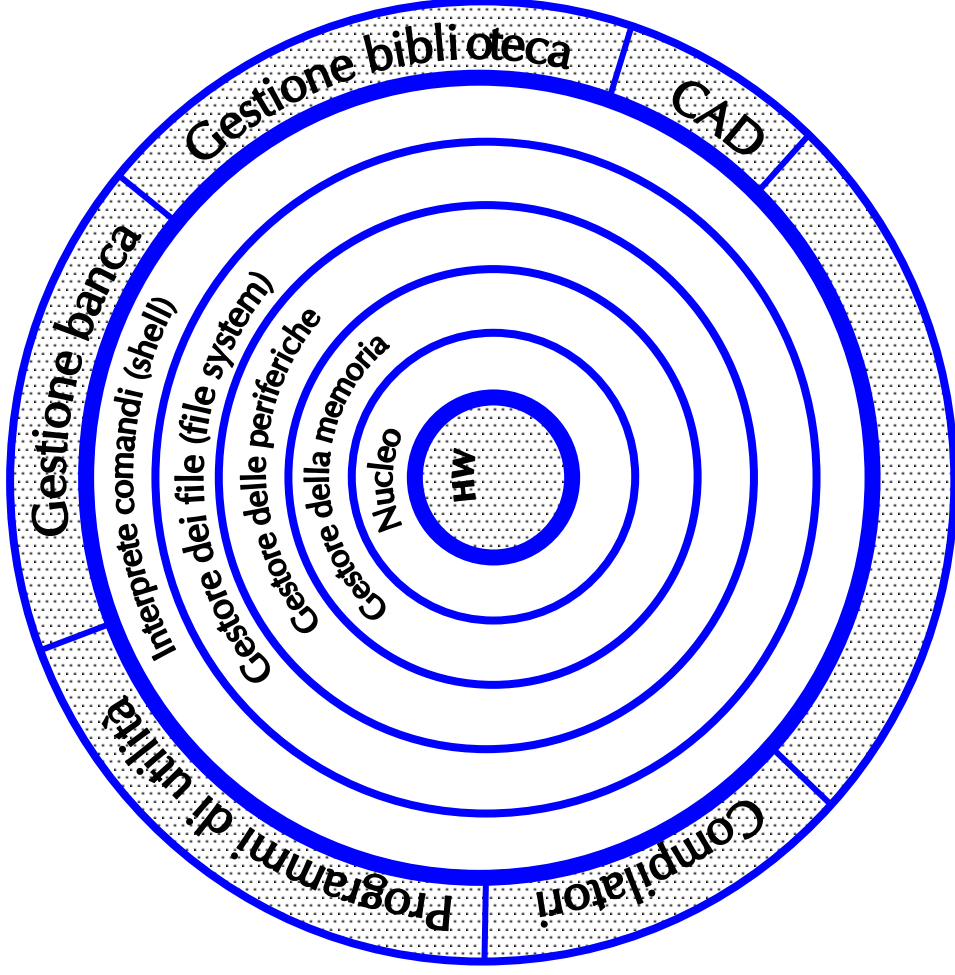
Time-sharing



Il sistema operativo: organizzazione

■ Un classico modello di organizzazione del S.O. è quello a “strati”, detto anche a “buccia di cipolla”:

- Determina una gerarchia di macchine virtuali (mv)
- La mv al livello N è determinata dall’hardware e dagli strati del S.O. fino a quel livello



Il nucleo (kernel)

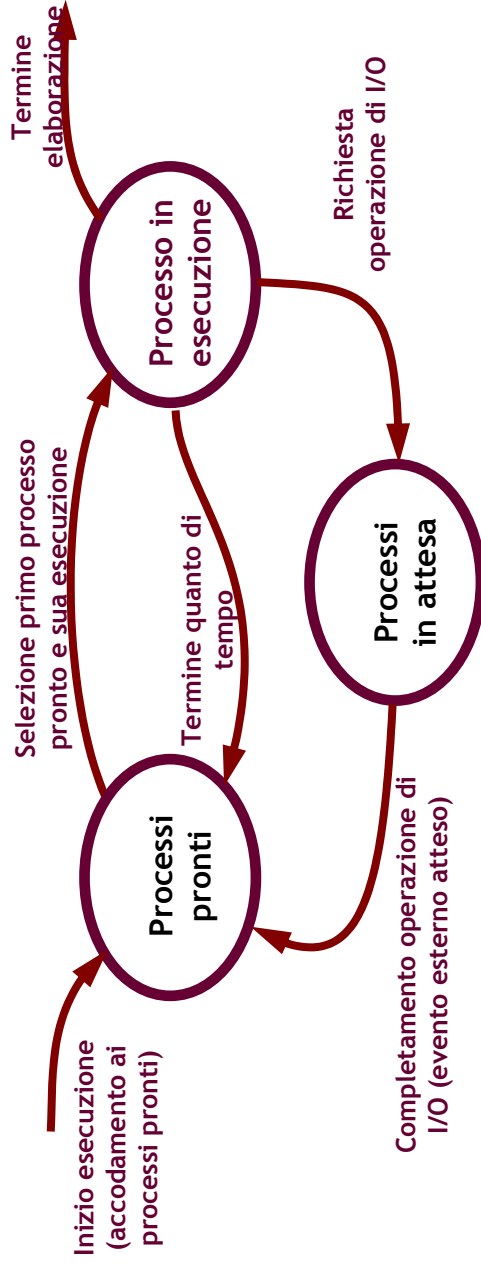
- Per la gestione dei processi il S.O. deve:
 - Interagire direttamente con l'**hardware**.
 - Occuparsi dell'esecuzione dei programmi e della risposta agli eventi esterni generati dalle **unità periferiche di I/O**.
 - Gestire i **processi** corrispondenti ai programmi che sono contemporaneamente attivi.
 - Gestire il **contesto di esecuzione** dei vari processi.
 - Attuare una politica di alternanza (**scheduling**) nell'accesso alla CPU da parte dei processi in esecuzione.
 - Essere eseguito sempre in modalità privilegiata (detta **kernel mode**).

La gestione dei processi

- Un **programma** è un insieme di istruzioni da eseguire (entità statica).
- Un **processo** è un'istanza di un programma in esecuzione (entità dinamica).
- Un processo è costituito dal programma e dal **contesto di esecuzione** (program counter, registri, memoria, etc.).
- In un S.O. multiprogrammato (o multitasking o multiprocesso) possono esistere più processi in esecuzione
- I processi possono essere eseguiti in **kernel mode** (ad es. i servizi forniti dal sistema operativo) o **user mode** (ad es. i programmi applicativi)

La gestione dei processi (2)

- Compiti del s.o. per la gestione dei processi
 - Creazione/terminazione dei processi
 - Sospensione/ripristino dei processi
 - Sincronizzazione/comunicazione tra processi
 - Gestione di situazioni di stallo (blocco critico o deadlock)
 - Esempio di diagramma di stato dei processi



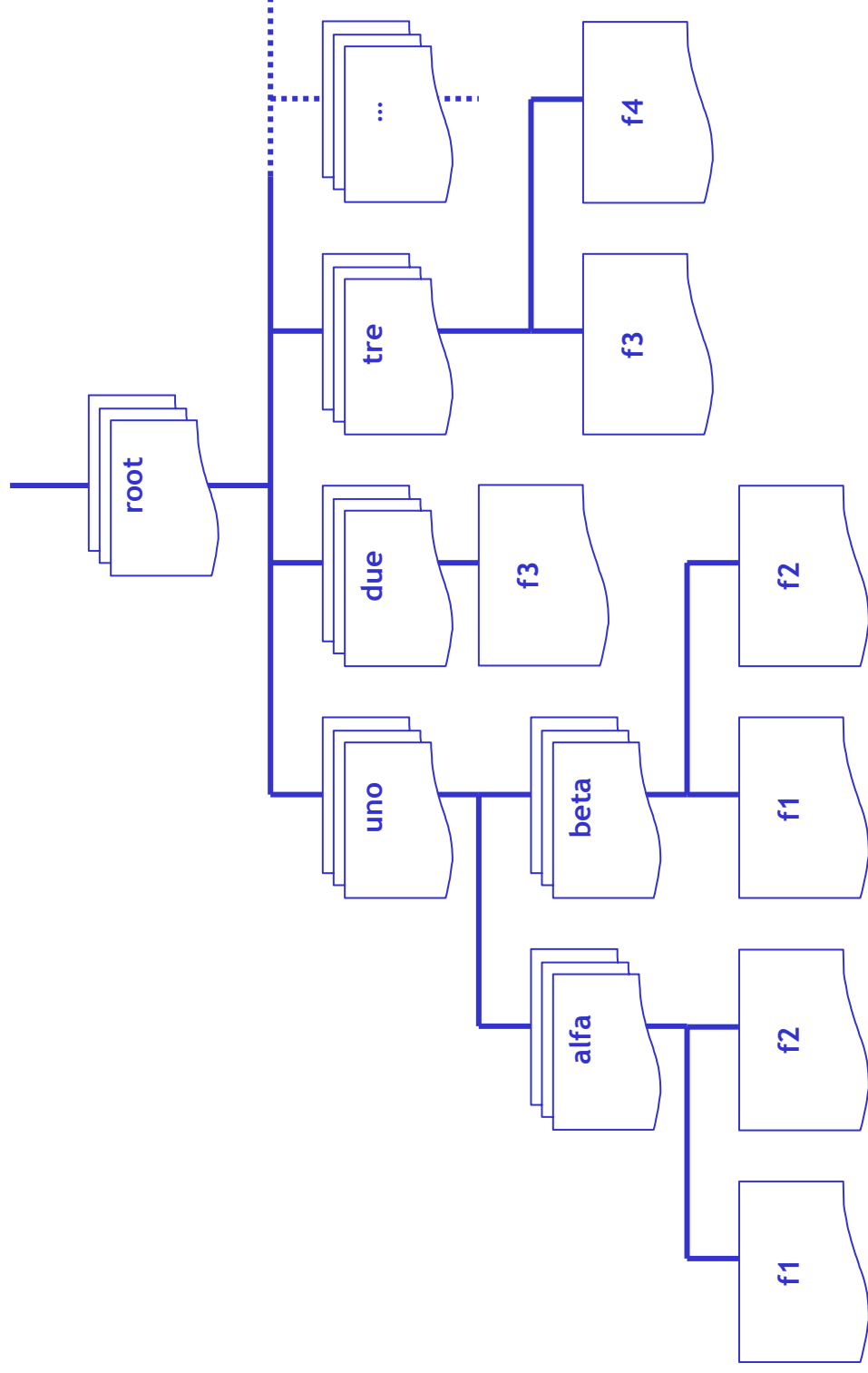
La gestione della memoria

- Il gestore della memoria centrale
 - Risolve le relative esigenze dei vari processi in esecuzione.
 - Consente ai programmi di lavorare in un proprio spazio di **indirizzamento virtuale** e di ignorare quindi le effettive zone di memoria fisica occupata.
 - Protegge **programmi** e **relativi dati** caricati nella memoria di lavoro.
 - Fornisce alle macchine virtuali di livello superiore la possibilità di lavorare come se esse avessero a disposizione una **memoria dedicata**, di capacità anche maggiore di quella fisicamente disponibile (memoria virtuale).

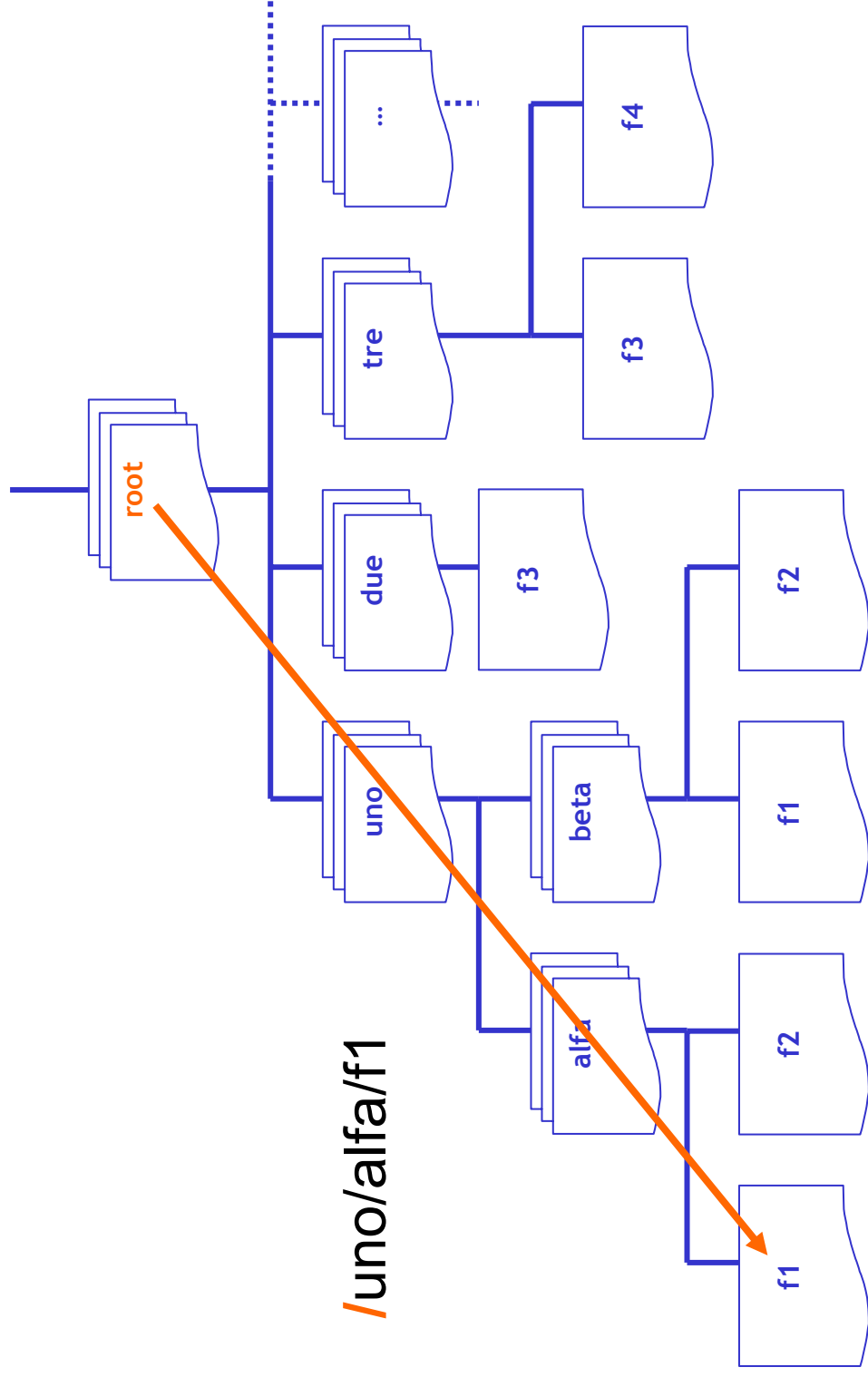
Il file system

- Permette di organizzare i dati contenuti nella memoria di massa in forma strutturata.
- I dati sono raccolti in strutture logiche dette **files**, identificate da un nome (**filename**).
- I files sono organizzati in più contenitori logici (**cartelle** o **directory**) secondo una struttura ad albero.
- Ciascun file è individuato mediante il suo **percorso (path) assoluto**, ottenuto giustapponendo i nomi dei nodi che si incontrano dalla radice al file (separati da un carattere apposito, in Linux **/**).
Esempio: `/uno/alfa/f1`

Il file system (2)



Il file system (2)



/uno/alfa/f1

La gestione del file system

- Il gestore del file system fornisce i servizi di base per
 - La **creazione/cancellazione** di file e cartelle.
 - La **manipolazione** di file e cartelle esistenti.
 - La **copia** e lo **spostamento** di dati su supporti diversi.
 - L'associazione tra file e dispositivi di memorizzazione secondaria (memorie di massa).
 - La gestione di **collegamenti** (**link o alias**) tra file e cartelle. Un collegamento è un riferimento a un oggetto (file o cartella) presente nel file system.

La gestione delle periferiche

- Il gestore delle periferiche
 - Fornisce una visione del sistema in cui i processi possono operare mediante periferiche astratte.
 - Maschera le caratteristiche fisiche delle periferiche e le specifiche operazioni di ingresso/uscita.
 - Mette a disposizione di ogni processo delle periferiche virtuali.
 - In alcuni S.O. come Linux, il gestore fa in modo che file, cartelle e periferiche I/O vengano presentati all'utente in modo uniforme.

L'interprete di comandi

- E' un modulo del S.O. direttamente accessibile dall'utente.
- Ha la funzione di interpretare i comandi che gli giungono e di attivare i programmi corrispondenti.
- Svolge operazioni di lettura della memoria centrale dei programmi da eseguire.
- Alloca la memoria necessaria e vi carica programmi e dati.
- Crea e attiva il processo relativo ad un programma.

Laboratorio di Informatica di Base

Laurea in Informatica

Docente: *Massimo Merro*
profs.sci.univr.it/~merro

Lucidi a cura di

Andrea Colombari, Carlo Drioli, Andrea Fusiello e Barbara Oliboni

Lezione 2

Introduzione a Linux



Testo di riferimento:
M. Bertacca, e A. Guidi
“Introduzione a Linux”
McGrawHill

Il S.O. GNU/Linux: introduzione

- GNU (GNU's Not UNIX) Linux è un sistema operativo libero di tipo Unix, distribuito con licenza GNU GPL (General Public License)
- Il kernel Linux nasce dalla collaborazione a distanza, grazie a Internet, di numerosi sviluppatori volontari
- E' un sistema **multiutente** e **multiprocesso** (**multitasking**)
- Adotta un proprio file system per la memorizzazione dei file (ext2fs) ma e' compatibile con i principali file system in uso (es. MS-DOS, Fat32)
- E' dotato di potenti interfacce a caratteri (**shell** di comando) ma anche di ambienti desktop evoluti come **KDE** e **GNOME**

Breve storia di Unix

- 1969 Nasce Unix presso i **Bell Labs** di AT&T. I padri sono **Ken Thomson** e **Dennis Ritchie** (quello del C).
- 1975 Unix V6 esce dai laboratori.
- 1977 Thomson in sabbatico a UC Berkley con **Bill Joy** e Chuck Haley sviluppa **BSD** (Berkley Software Development) derivato da Unix V6.
- 1982 Bill Joy fonda Sun e nasce SunOS da 4.1BSD.
- 1983 AT&T rilascia Unix **System V**.
- 1983 Rilascio di **4.2BSD**.
- Tutte le successiva variazioni di Unix fanno riferimento a questi due ceppi: System V e BSD.

Breve storia di Unix (continua)

- Nel 1984 **Richard Stallman** fonda il progetto **GNU** con lo scopo di creare una versione non commerciale (*free software*) di Unix.
- Nel 1987 A.S. Tanenbaum pubblica **MINIX**, una versione didattica di Unix (non free).
- Nel 1991 lo studente **Linus Torvalds** scrive una versione non commerciale di MINIX, che diventerà il **Linux** kernel (nucleo) che fondendosi con GNU darà origine al s.o. **GNU/Linux**.
- Nel 1999 Apple rilascia **Mac Os X**, basato sulla implementazione di BSD in NEXTSTEP, il s.o. sviluppato da **Steve Jobs** (co-fondatore di Apple) per NeXT.

Steve Jobs

QuickTime™ e un
decompressore TIFF (Non compresso)
sono necessari per visualizzare quest'immagine.

GNU: un po' di storia...

- **GNU** è un acronimo “ricorsivo” e significa **GNU's Not UNIX**. Proprio perché nasce con l'idea di sviluppare un S.O. stile UNIX ma libero, che permettesse di sviluppare liberamente favorendo così la collaborazione tra programmatori.
- Il Progetto GNU (1983, **Richard Stallman**), si basa sulla gestione dei diritti d'autore secondo la definizione di **software libero** (contrapposta a software proprietario).
- Fulcro del Progetto GNU è la licenza GNU General Public License (**GNU GPL**), che sancisce e protegge le libertà fondamentali che, secondo Stallman, permettono l'uso e lo sviluppo collettivo e naturale del software.
- Nel 1984 inizia lo sviluppo del Sistema GNU. In realtà il kernel di tale sistema Hurd è tuttora in lavorazione. Ma nel 1991 **Linus Torvalds** scrisse il kernel Linux e lo distribuì sotto licenza GNU GPL. I sistemi GNU con kernel Linux vengono ufficialmente chiamati **GNU/Linux**.

Il SO Linux: accesso alla macchina

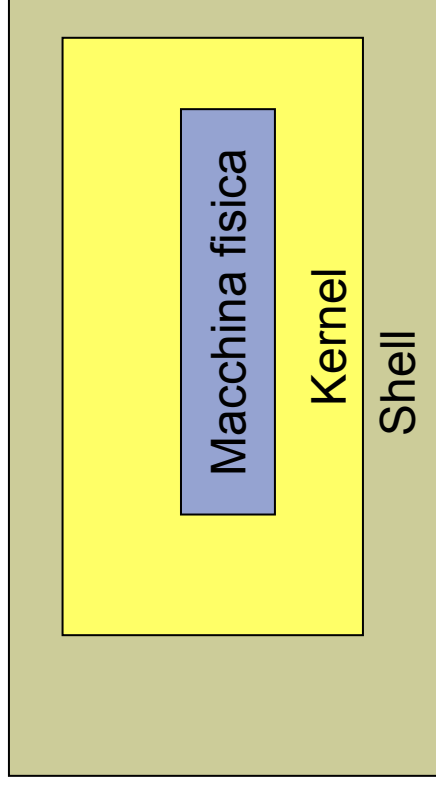
- Per accedere ad una sessione Linux è necessario disporre di un nome utente riconosciuto dal sistema (**login**) e di una parola d'ordine (**password**)

```
login: mialogin
password: *****
```

- Dopo l'autenticazione, l'utente accede alla propria cartella di lavoro (**home directory**, ad es. **/home/mialogin**) e può interagire con il S.O. mediante un interprete di comandi (**shell**)
- Il prompt (**\$**) avvisa l'utente che l'interprete è pronto ad accettare comandi

La Shell e i comandi

- La shell è l'interfaccia tra l'utente e il sistema operativo.
- Legge i comandi dell'utente, li interpreta e richiede al sistema l'esecuzione delle operazioni richieste dal comando.



Comandi

- Per terminare la sessione di lavoro si possono usare i comandi **logout** o **exit**. Il sistema tornerà nello stato di richiesta di login e password

```
$ logout  
oppure  
$ exit
```

- Per “listare” il files presenti nella cartella di lavoro impartire il comando

```
$ ls
```

- Per creare un file vuoto, impartire il comando

```
$ touch nome_file
```

Comandi

- I comandi hanno opzioni che ne modificano il comportamento, p.es.

```
$ ls -l
```

visualizza informazioni più dettagliate sui files.

- In generale la sintassi di un comando Linux è:

```
comando [opzioni] [argomenti]
```

Caratteri jolly o metacaratteri

- ★ Sostituisce un insieme di zero o più caratteri qualsiasi

Esempio:

```
$ ls c*
```

- ? Sostituisce un carattere qualsiasi

Esempio:

```
$ ls co??o.txt
```

- [] Permettono di specificare una lista e/o un intervallo di caratteri possibili

Esempio:

```
$ ls [a-c]*.txt
```

Nomi dei file

- I nomi “ ” “ .. ” e “ ~ ” sono riservati (vedremo per cosa).
- Tutti gli altri sono validi, ma è buona norma evitare i metacaratteri.
- Di solito si usano i caratteri alfanumerici ed il trattino basso (*underscore*).
- I file che iniziano con “ . ” non vengono visualizzati da ls (usare opzione -a)
- È buona norma che il nome del file termini con un suffisso costituito da un punto e alcuni caratteri che indicano il tipo di file. Es: prova.txt, lenna.png, mio.c

Il manuale dei comandi

- Il **manuale** dei comandi descrive l'utilizzo e le caratteristiche di ogni comando. Le pagine del manuale si invocano con

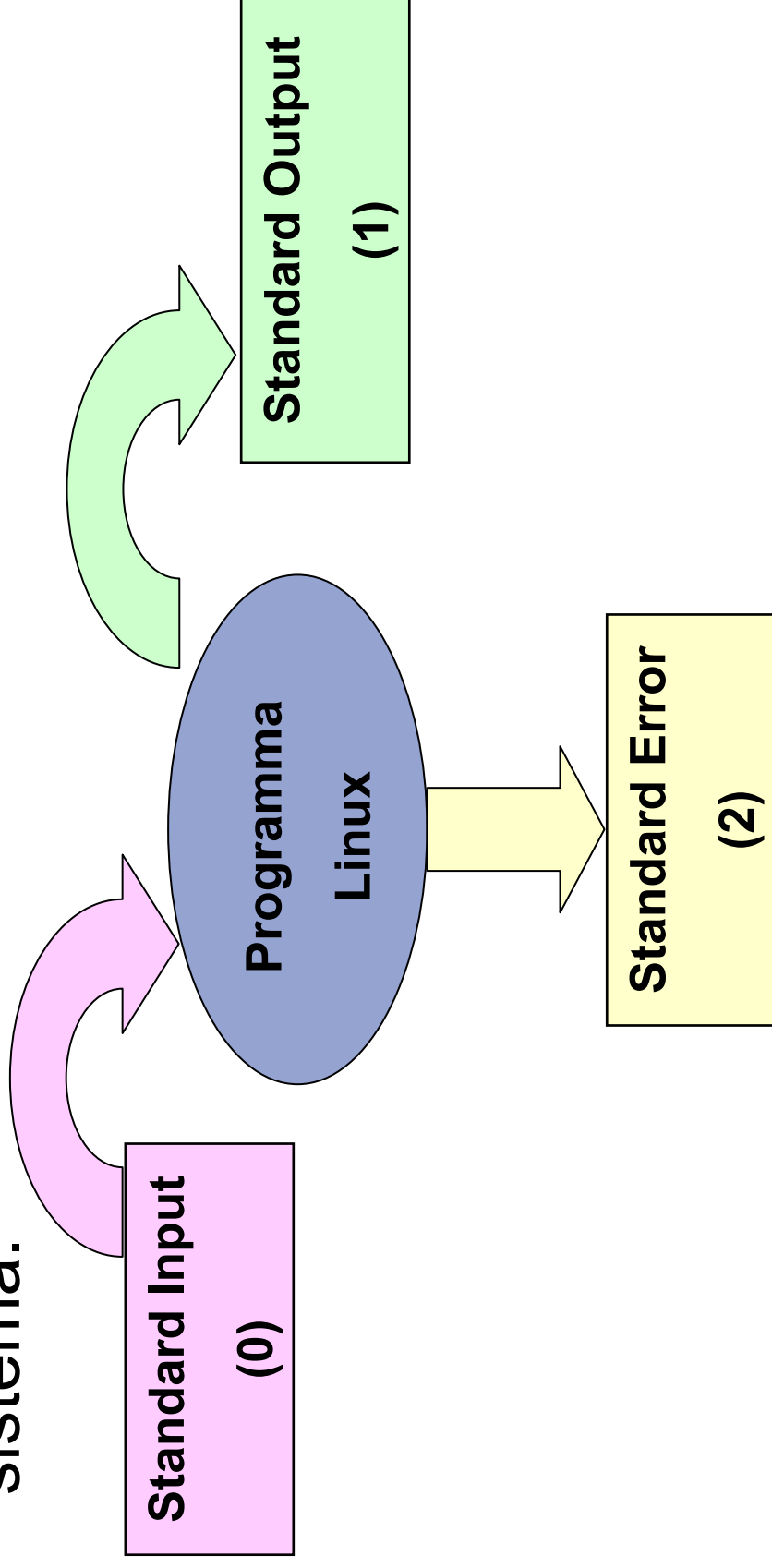
`man [comando]`

e hanno tutte la seguente struttura comune:

NAME: riporta il nome del comando e una breve descrizione delle sue funzioni
SYNOPSIS: descrive la sintassi del comando
DESCRIPTION: descrive lo scopo e il funzionamento del comando
OPTIONS: riporta il funzionamento di tutte le opzioni
ENVIRONMENT: descrive eventuali variabili d'ambiente che interagiscono con il comando
AUTHOR: note sull'autore del comando
COPYRIGHT: note su copyright
BUGS: eventuali errori o malfunzionamenti noti
SEE ALSO: eventuali altre pagine del manuale a cui fare riferimento

Redirezione (1)

- Un comando Linux quando viene eseguito ha sempre tre flussi preventivamente aperti dal sistema.



Redirezione (2)

- I tre flussi sempre aperti di un comando si possono *redirigere* da o verso un file.
 - Un comando può scrivere le informazioni che produce su un file piuttosto che sul video (standard output).

```
$ ls -l >lista
$ ls -l >>lista
$ echo 3 + 4 >conto
```

- Un comando può leggere le informazioni di cui necessita da un file piuttosto che da tastiera (standard input).

```
$ bc < conto
7
```

- Un comando può scrivere i messaggi di errore su un file piuttosto che sul video (standard error).

```
$ ls -l /qkxq 2>lista
```

Convogliamento (piping)

- La shell possiede un meccanismo che permette di convogliare direttamente lo standard output di un comando sullo standard input di un'altro comando:
- Risorsa che permette questo tipo di comunicazione: **pipe** (condotto).

```
$ ls -l /bin | more
```

- La **pipe** permette solo uno scambio di dati unidirezionale.

Il ruolo della shell

- Una precisazione: è la shell che effettua l'espansione dei metacaratteri (globbing), la redirectione, il convogliamento, mantiene la storia dei comandi ed altre funzionalità che vedremo.
- I **comandi** invece sono forniti dal S.O.
- Infatti, dentro Linux, si possono avere shell diverse:
 - Bash (default)
 - C-shell (/bin/csh)
 -
- I comandi sono sempre gli stessi, ma le funzionalità elencate sopra cambiano nella forma e nella sostanza.