Informatica A e B

Antonio Lieto

Parte III

Software e Sistema Operativo

(Come usiamo un computer?)

Software

- Il computer è una macchina programmabile
- l'hardware da solo non è sufficiente a far funzionare l'elaboratore -> è necessario introdurre del software ovvero…
- ...un insieme di programmi che permettono di trasformare un insieme di circuiti elettronici in un oggetto in grado di svolgere:
  - operazioni di natura diversa
  - per diversi tipi di utenti

Software

- Il software (= i programmi) è costituito da istruzioni che dicono alla macchina hardware cosa deve fare.
- Una programmazione diretta della macchina hardware da parte degli utenti? Difficile. Qual è il problema?

• l'utente dovrebbe conoscere l'organizzazione fisica dell'elaboratore e il suo linguaggio macchina (improbabile!)
• ogni programma dovrebbe essere scritto utilizzando delle sequenze di bit (0-1) (inumano!)
• l'hardware cambia da macchina a macchina! Un programma scritto per la macchina A non funzionerebbe se eseguito sulla macchina B -> ogni piccola differenza hardware comporterebbe una riscrittura del programma (non portabilità!)

Alcune note programmatrici…

Ada Lovelace

Grace Hopper

Margaret Hamilton

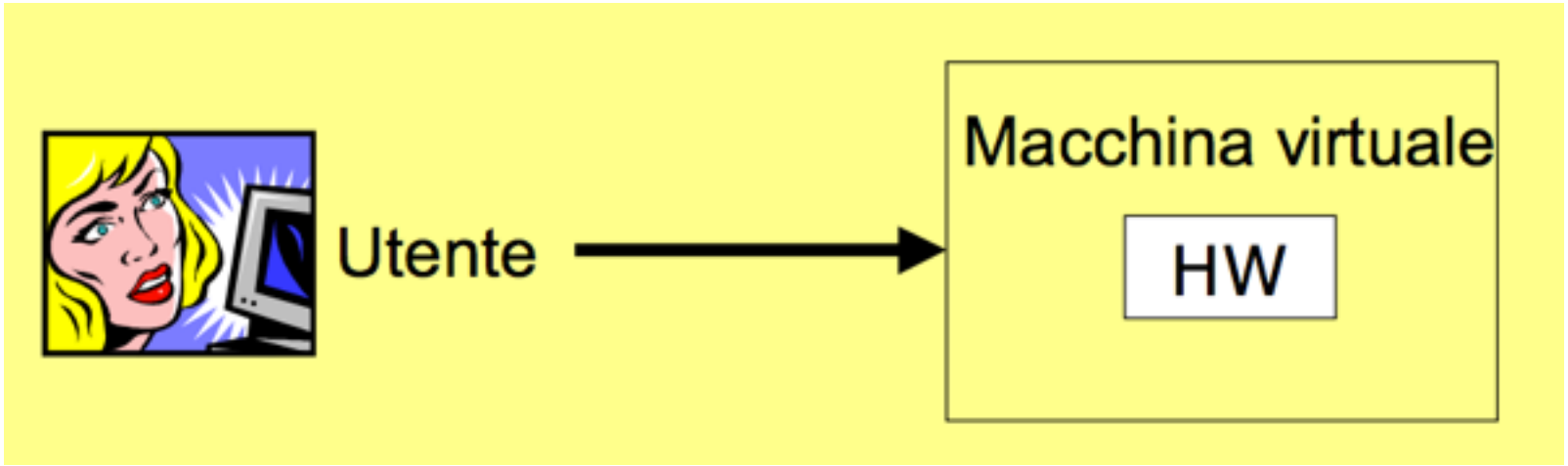
many more…

Software

- Occorre fornire all'utente un modo per:
  - astrarre dalle caratteristiche fisiche della macchina
  - avere un linguaggio semplice di interazione con la macchina
  - riuscire a interagire più o meno allo stesso modo con macchine con caratteristiche hardware un po' diverse
  - avere un insieme di programmi applicativi per svolgere compiti diversi

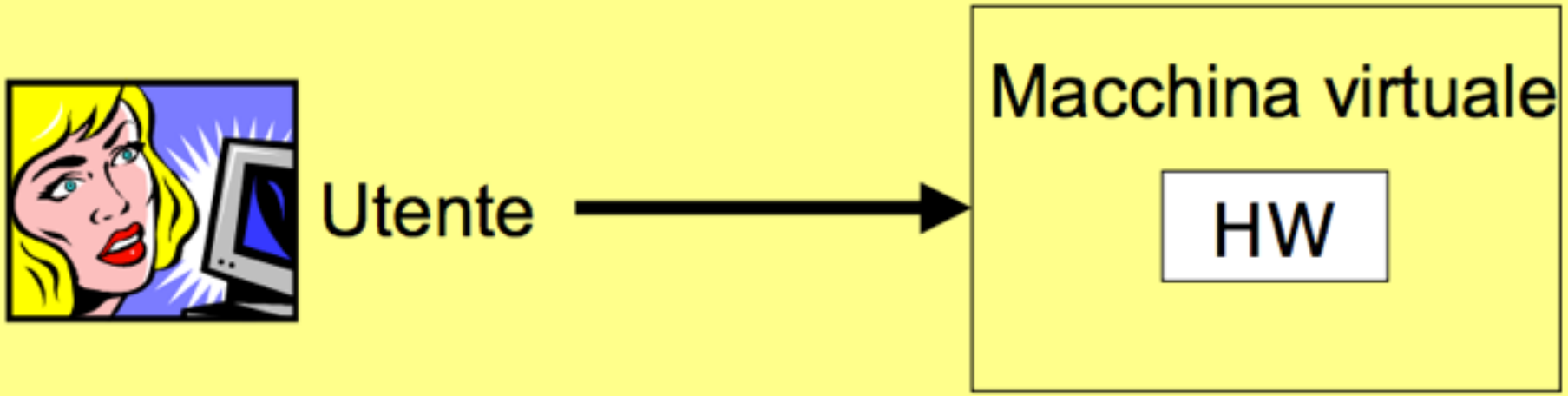
Macchina virtuale

- Nei moderni sistemi di elaborazione questi obiettivi vengono raggiunti grazie alla definizione di una macchina virtuale che viene realizzata al di sopra della macchina hardware reale
- Virtuale in quanto essa non esiste fisicamente ma viene realizzata mediante software: il software di base o di sistema, che fornisce all'utente una visione sul calcolatore astratta, con cui interagire



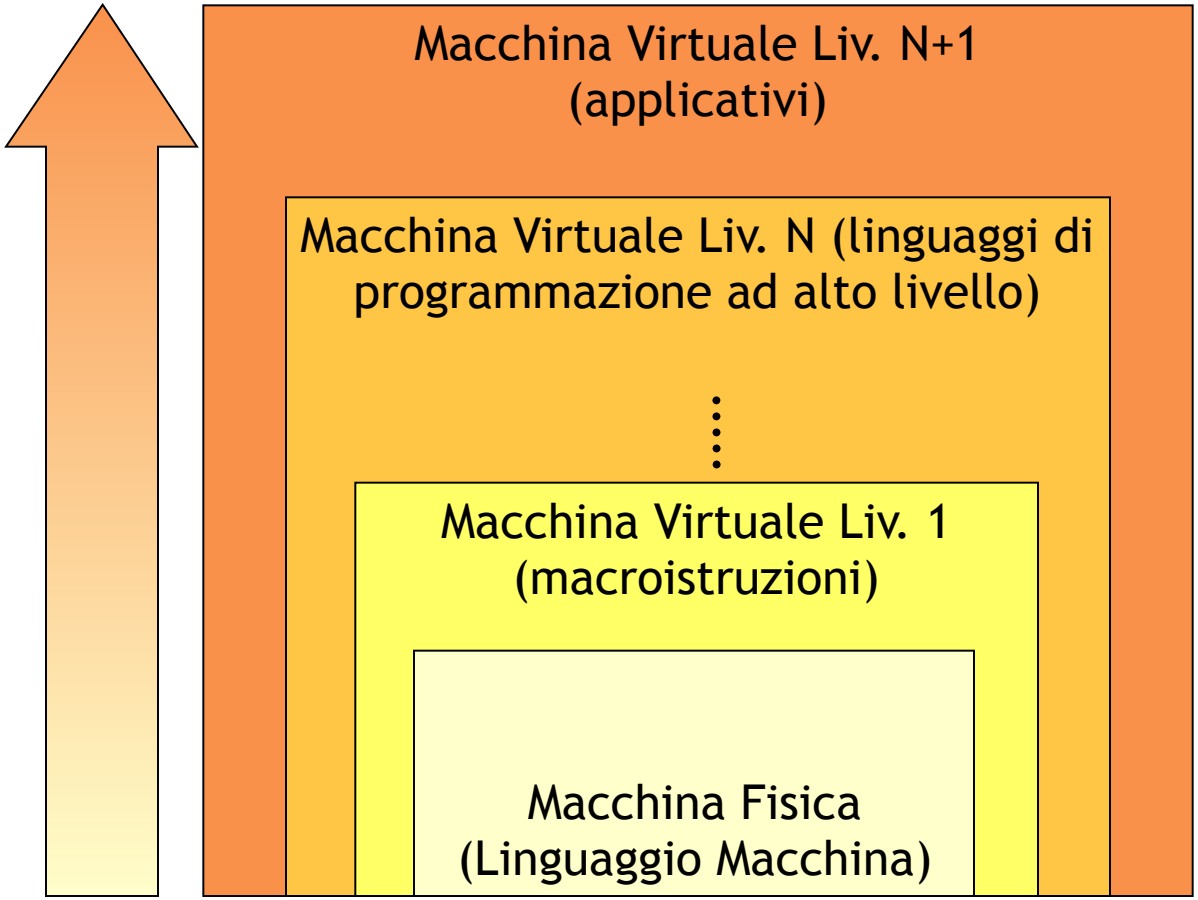
Macchina virtuale

- L'utente interagisce con la macchina virtuale grazie ad un opportuno linguaggio di comandi
- La macchina virtuale si preoccupa della traduzione di ogni comando impartito dall'utente nella sequenza di comandi che realizzano la stessa funzione e sono riconosciuti dalla macchina fisica sottostante

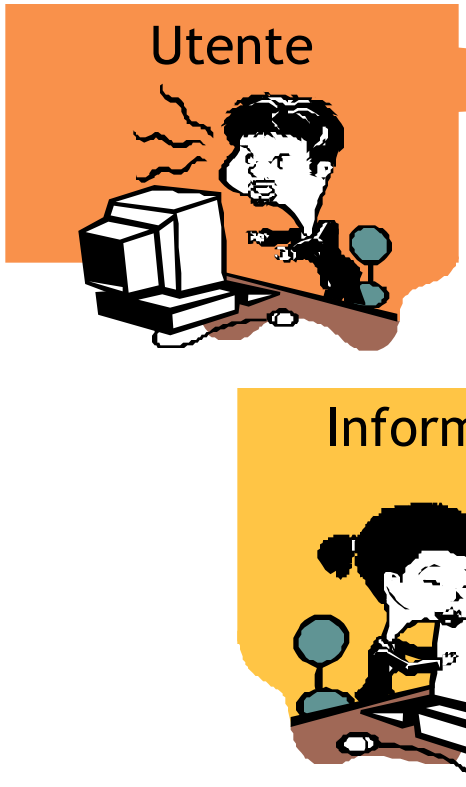


Astrazione

- Maggiore **astrazione** dalle caratteristiche fisiche della macchina.
- Maggiore **facilità** d'uso.
- Maggiore **rapidità** nell'istruire la macchina in compiti complessi.



Astrazione


Utente
Informatico
Macchina Virtuale Liv. N+1
(sistema operativo e applicativi)
Macchina Virtuale Liv. N (linguaggi di programmazione ad alto livello)
Macchina Virtuale Liv. 1
(macroistruzioni)
Macchina Fisica
(Linguaggio Macchina)

Software di base

- Gli strumenti software che permettono all'utente (e ai programmi applicativi) di gestire le risorse fisiche e interagire con l'elaboratore in modo semplice sono parte della macchina virtuale e ci riferiamo ad essi come software di base;
- Denota un insieme di programmi che, a livello macroscopico, offrono due classi di funzioni
  - funzioni proprie del sistema operativo
  - funzioni di traduzione tra linguaggi diversi
- Sistemi operativi: la componente software principale, presente in qualsiasi computer

Il ruolo del Sistema Operativo

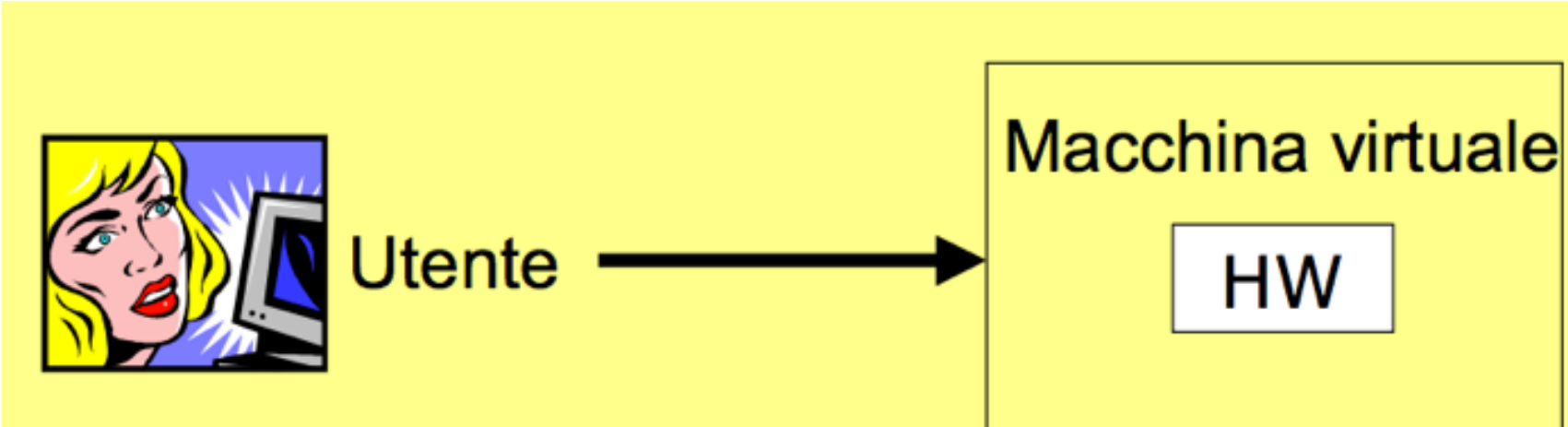
- E' costituito da programmi per la gestione delle operazioni più elementari di un computer che contribuiscono in modo determinante a creare la macchina virtuale
- Ha due obiettivi principali:
  1. Gestione efficiente delle componenti fisiche dell'elaboratore: CPU, memoria, periferiche) in modo da sfruttarne al meglio le potenzialità
  2. Creazione di un ambiente virtuale per facilitare l'interazione uomo-macchina, ovvero creazione dell'ambiente di lavoro in cui l'utente interagisce con la macchina
- Consente l'esecuzione del software applicativo sul computer

Il ruolo del Sistema Operativo

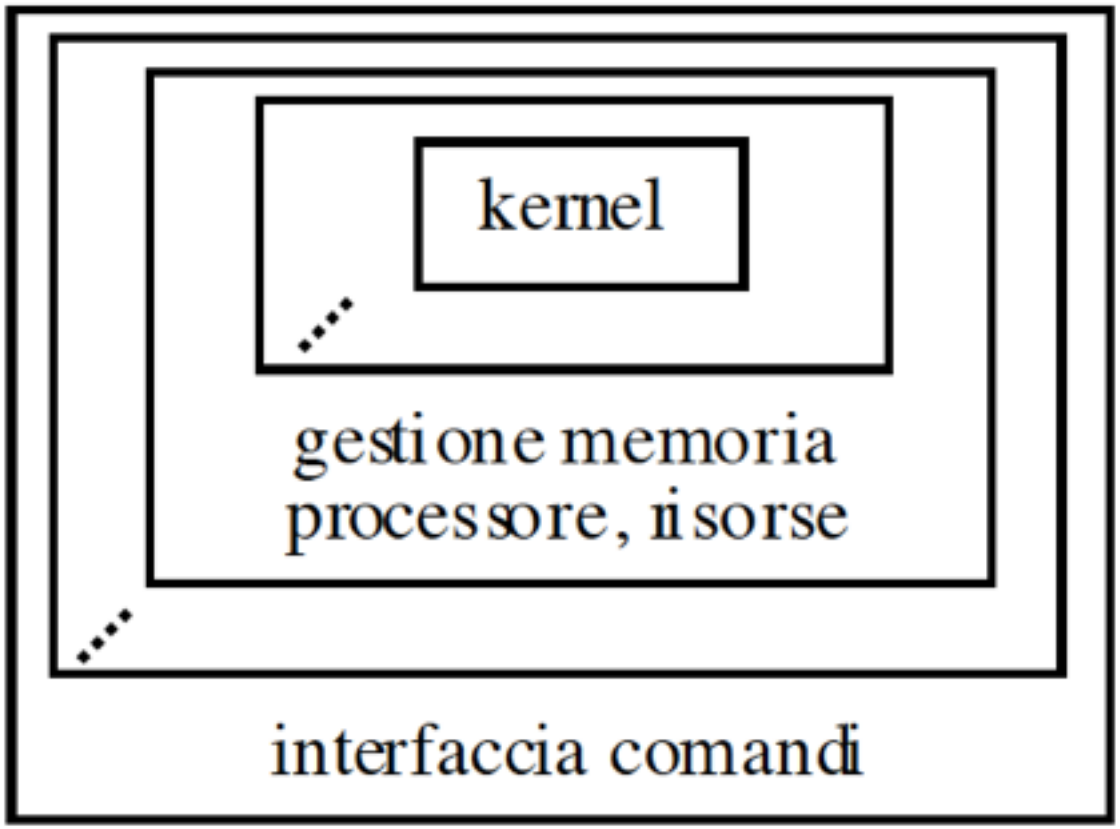
- **Automatizza** la gestione di molti compiti contribuendo ad alzare il livello di astrazione.
- È fondamentale nell'ultimo **livello di astrazione**.
  - Permette all'utente di usare un computer senza conoscere in dettaglio i suoi meccanismi e senza saperlo programmare!
    - L'utente gestisce le attività del computer attraverso il sistema operativo.
    - Esegue programmi già scritti per lui dai programmatori (applicativi).

Sistema operativo

- Ob 1. Gestione efficiente delle componenti fisiche dell'elaboratore: CPU, memoria, periferiche) in modo da sfruttarne al meglio le potenzialità
- Il sistema operativo realizza una macchina virtuale con cui l'utente interagisce, che può essere più potente di quella fisica
- L'effetto di questa gestione efficiente è che l'utente ha l'impressione di interagire con una macchina con più potenza di calcolo e più memoria di quella che realmente ha



Struttura S.O. (visione inversa del livello di astrazione)



Struttura a cipolla

Struttura teorica del S.O.

Esempio: sistemi multi-utente

- Ho un sistema multi-utente (una macchina utilizzabile da più utenti contemporaneamente)
- E' realizzato con una sola macchina hardware potente collegata e tanti piccoli terminali collegati ad essa che sfruttano le sue risorse (memoria, potenza di calcolo, stampante)

- Il SO nasconde all'utente la presenza di altri utenti, creando per lui una sorta di macchina virtuale in cui lui ha l'impressione di lavorare da solo e di avere a disposizione un sistema dedicato solo alle proprie elaborazioni

Sistema operativo

- Ob 2: Creazione di un ambiente virtuale per facilitare l'interazione uomo-macchina ovvero creazione dell'ambiente di lavoro in cui l'utente interagisce con la macchina
- Il sistema operativo ha il compito di creare un'interfaccia intuitiva verso l'utente che fornisca un linguaggio di comandi semplice per l'interazione con la macchina

Interfaccia utente

- Consente al computer di interpretare le richieste dell'utente all'interno delle varie applicazioni tramite vari comandi
- Esistono vari tipi di interfacce che variano rispetto al modo in cui l'utente può comunicare con la macchina -> vari modi di permettere all'utente di dare comandi
- Once upon a time ... solo interfacce a comandi (no mouse): l'utente doveva digitare istruzioni di tipo testuale per dare un comando in input al computer:
  >> copy a:\pippo.doc c:
- Interfacce grafiche (GUI): i comandi sono impartiti mediante l'interazione via il mouse. Il clic del mouse su un'icona viene tradotto in una opportuna sequenza di istruzioni che il calcolatore esegue per soddisfare la richiesta dell'utente

Esempio

Realizzazione di una scrivania elettronica

GUI

Graphical user interface

From Wikipedia, the free encyclopedia

"GUI" redirects here. For other uses, see Gui (disambiguation).

In computer science, a **graphical user interface** or **GUI**, pronounced /ɡuːi/ ("gooey")[1] is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs),[2][3][4] which require commands to be typed on the keyboard.

The actions in a GUI are usually performed through direct manipulation of the graphical elements.[5] In addition to computers, GUIs can be found in hand-held devices such as MP3 players, portable media players, gaming devices, smartphones and smaller household, office and industrial equipment. The term "GUI" tends not to be applied to other low-resolution types of interfaces with display resolutions, such as video games (where HUD[6] is preferred), or not restricted to flat screens, like volumetric displays[7] because the term is restricted to the scope of two-dimensional display screens able to describe generic information, in the tradition of the computer science research at the PARC (Palo Alto Research Center).

A Unix-based X Window System desktop

Contents [hide]

GUI

- Graphical User Interface: A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages.
- On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.
- Graphical user interfaces, such as Microsoft Windows and the one used by the Apple Macintosh, feature the following basic components:
  - pointing device (puntatore)
  - icone (icons)
  - desktop (scrivania)
  - finestre (windows)
  - menu

GUI

- pointing device: A device, such as a mouse or
- trackball, that enables you to select objects on the display screen.
- icons: Small pictures that represent commands, files, or windows. By moving the pointer to the icon and pressing a mouse button, you can execute a command or convert the icon into a window. You can also move the icons around the display screen as if they were real objects on your desk.
- desktop: The area on the display screen where icons are grouped is often referred to as the desktop because the icons are intended to represent real objects on a real desktop.

GUI

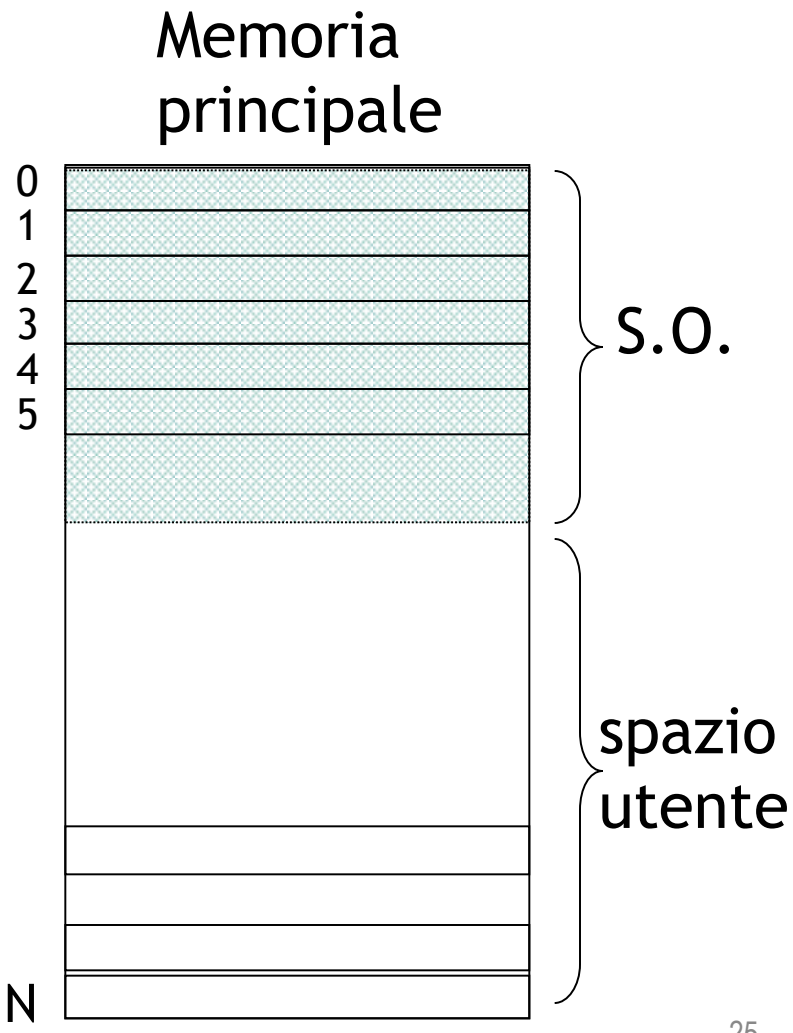
- windows: You can divide the screen into different areas. In each window, you can run a different program or display a different file. You can move windows around the display screen, and change their shape and size at will.
- menus: Most graphical user interfaces let you execute commands by selecting a choice from a menu.

Cosa fa il sistema operativo?

1. Bootstrapt: configurazione e accensione della macchina
2. Permette di avere in esecuzione più programmi contemporaneamente:
   - Gestione del processore → come "distribuire" il suo tempo fra programmi diversi?
   - Gestione della memoria principale → come "distribuire" il suo spazio fra programmi diversi?
3. Gestisce la memoria secondaria.
4. Permette agli utenti di interagire con il computer:
   - Gestione del sistema operativo stesso:
     - Installare (e disinstallare) programmi .
     - Configurare il computer e il sistema operativo.
   - Attività vera e propria:
     - Mandare in esecuzione un programma con cui creare nuovi documenti o elaborare documenti esistenti.
     - Eliminare e rinominare documenti creati e riorganizzarne la disposizione.
     - Utilizzare le periferiche.

Avvio dell'elaboratore

- Il sistema operativo viene mandato in esecuzione al momento dell'accensione del computer.
- Questa fase prende il nome di ***bootstrap***.
- In questa fase una parte del sistema operativo viene caricata nella memoria principale.
- Una parte del sistema operativo deve essere sempre mantenuta in memoria principale e deve essere sempre pronta per l'esecuzione.



Cosa fa il sistema operativo?

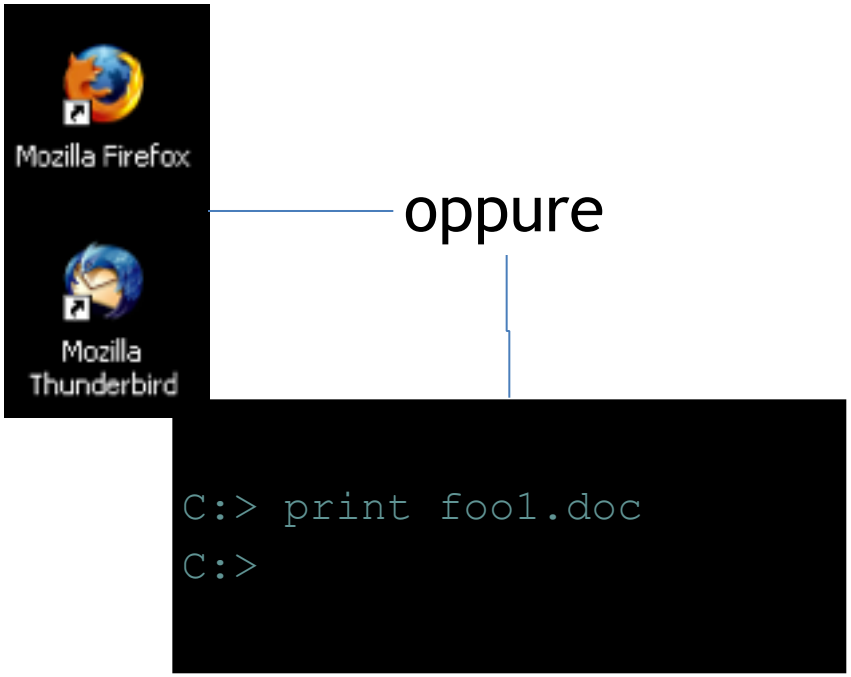
1. Bootstrapt: configurazione e accensione della macchina
2. Permette di avere in esecuzione più programmi contemporaneamente:
   - Gestione del processore → come "distribuire" il suo tempo fra programmi diversi?
   - Gestione della memoria principale → come "distribuire" il suo spazio fra programmi diversi?
3. Gestisce la memoria secondaria.
4. Permette agli utenti di interagire con il computer:
   - Gestione del sistema operativo stesso:
     - Installare (e disinstallare) programmi .
     - Configurare il computer e il sistema operativo.
   - Attività vera e propria:
     - Mandare in esecuzione un programma con cui creare nuovi documenti o elaborare documenti esistenti.
     - Eliminare e rinominare documenti creati e riorganizzarne la disposizione.
     - Utilizzare le periferiche.

Cosa fa il sistema operativo?

1. Bootstrapt
2. Permette di avere in esecuzione più programmi contemporaneamente:
   - Gestione del processore → come "distribuire" il suo tempo fra programmi diversi?
   - Gestione della memoria principale → come "distribuire" il suo spazio fra programmi diversi?
3. Gestisce la memoria secondaria.
4. Permette agli utenti di interagire con il computer:
   - Gestione del sistema operativo stesso:
     - Installare (e disinstallare) programmi .
     - Configurare il computer e il sistema operativo.
   - Attività vera e propria:
     - Mandare in esecuzione un programma con cui creare nuovi documenti o elaborare documenti esistenti.
     - Eliminare e rinominare documenti creati e riorganizzarne la disposizione.
     - Utilizzare le periferiche.

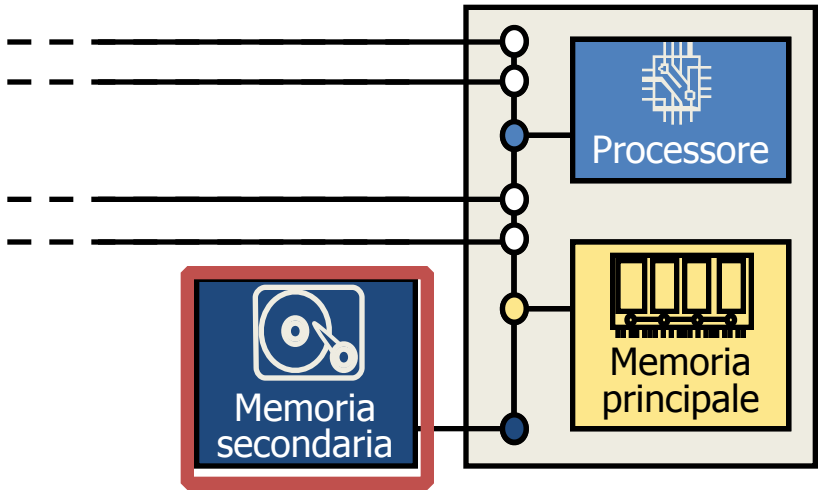
Esecuzione dei programmi

- Quando si fa doppio clic sull'icona di un programma (oppure si scrive un comando)
  …



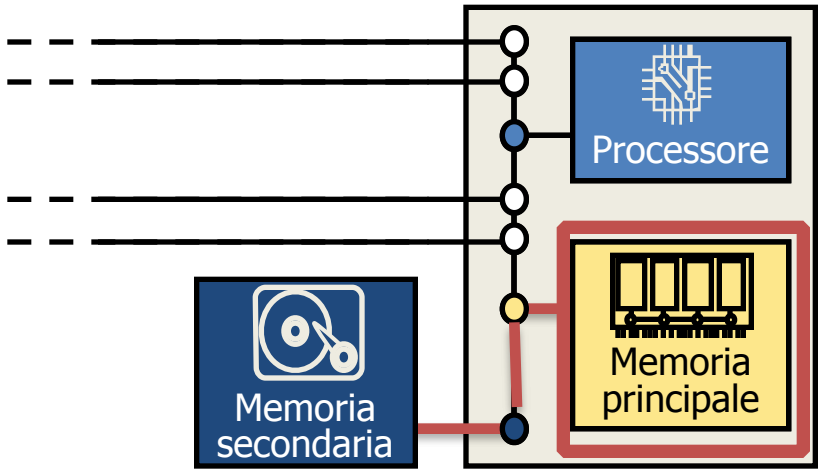
Esecuzione dei programmi

- Quando si fa doppio clic sull'icona di un programma (oppure si scrive un comando) …
- … il sistema operativo:
  - Cerca il programma corrispondente sulla memoria secondaria
  - …



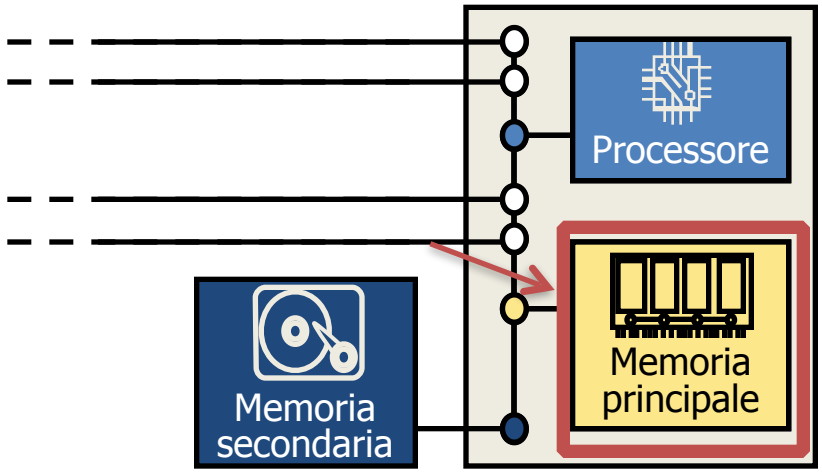
Esecuzione dei programmi

- Quando si fa doppio clic sull'icona di un programma (oppure si scrive un comando) …
- … il sistema operativo:
  - Cerca il programma corrispondente sulla memoria secondaria
  - Copia il programma in memoria principale
  - …



Esecuzione dei programmi

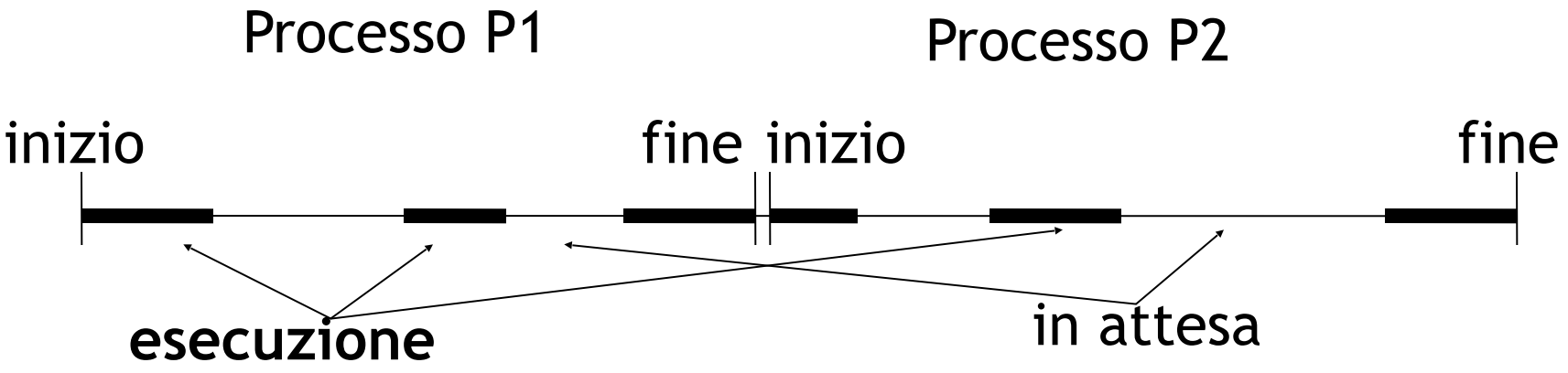
- Quando si fa doppio clic sull'icona di un programma (oppure si scrive un comando) …
- … il sistema operativo:
  - Cerca il programma corrispondente sulla memoria secondaria.
  - Copia il programma in memoria principale.
  - Imposta il registro Program Counter (PC) con l'indirizzo in memoria principale della prima istruzione del programma.



Sistemi mono-utente, mono-programmati

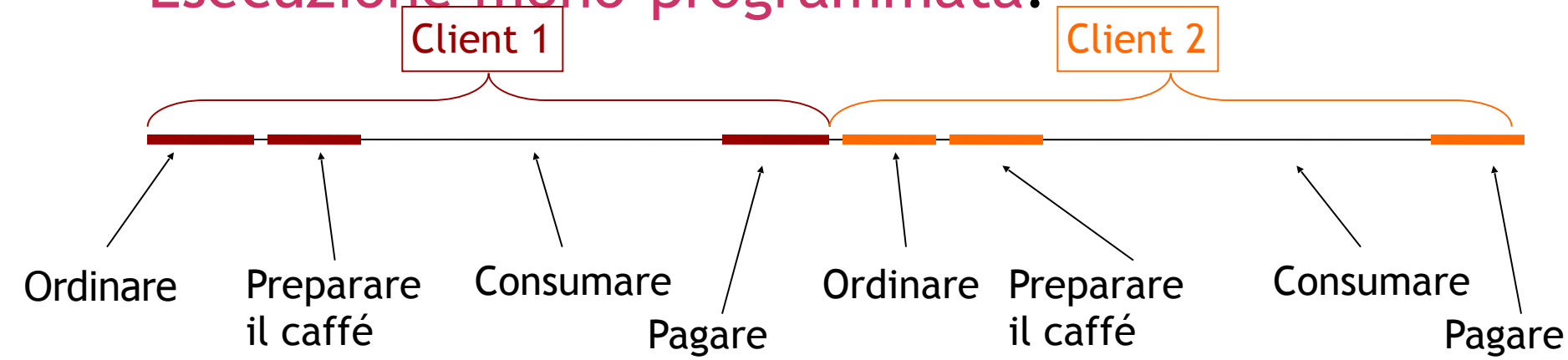
- Un solo utente può eseguire un solo programma alla volta.
  - È forzato a "sequenzializzare" i programmi.
  - Il programma viene lanciato, eseguito e quindi terminato.
- Così il processore ***non*** viene sfruttato al meglio: si spreca molto tempo.
  - Il processore è molto più veloce dei supporti di memoria secondaria e delle altre periferiche.
  - Passa (potenzialmente) la maggior parte del suo tempo in attesa.
  - Durante l'attesa si dice che il processore è un uno stato inattivo (idle).
  - Quindi: necessità di più processi in parallelo
- Terminologia: un ***processo*** è un programma in esecuzione.

Esecuzione sequenziale



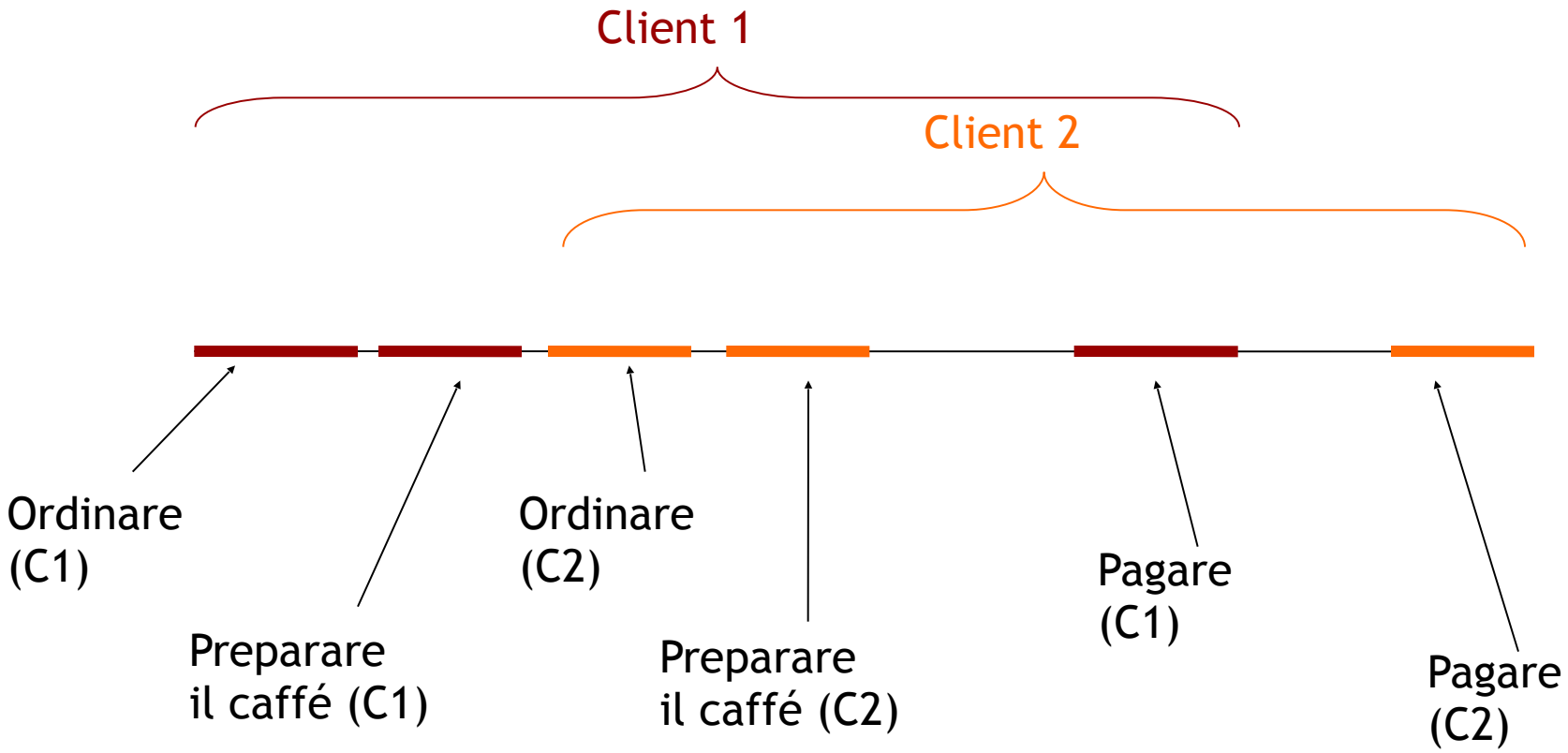
Esecuzione sequenziale

- Supponiamo che il nostro sistema sia un bar in cui il barista serve diversi clienti.
- Il barista è corrispondente del processore, i clienti sono l'equivalente dei processi da eseguire.
- Esecuzione mono-programmata:



Soluzione

- In realtà:

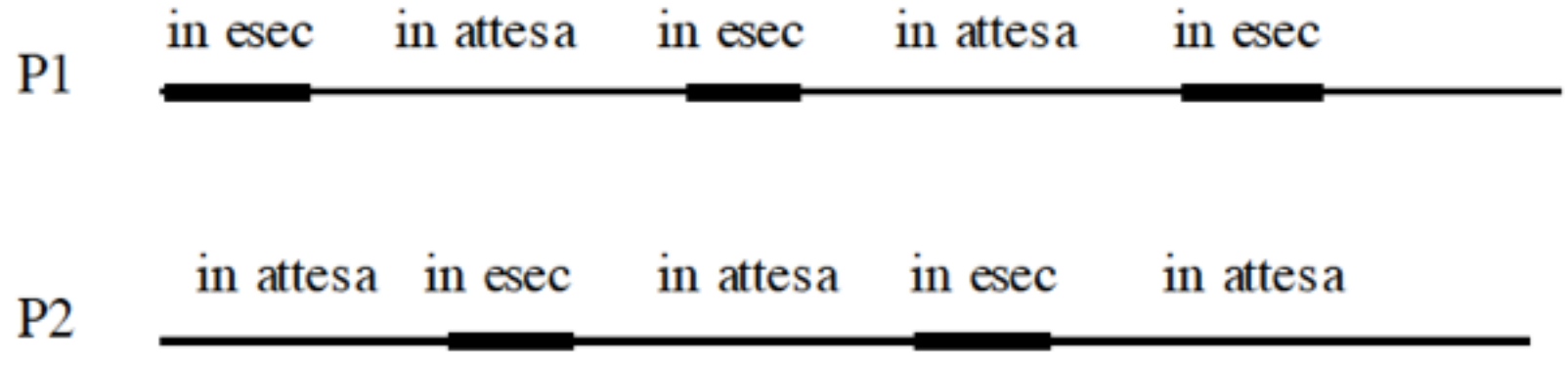


Soluzione: sistemi multiprogrammati

- Quando il processore è nello stato di idle lo si può sfruttare per eseguire (parte di) un altro processo.
- Quando un processo si ferma (per esempio in attesa di un dato dall'utente) il processore può passare ad eseguire le istruzione di un altro processo.
- Il sistema operativo si occupa di gestire l'**alternanza** tra i processi in esecuzione.

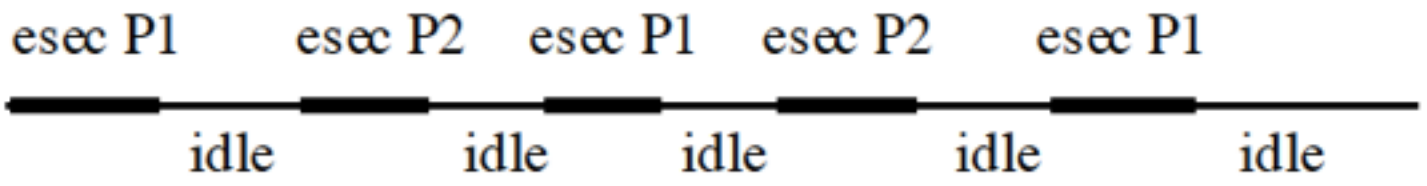
Sistemi multiprogrammati

Dal punto di vista dei processi:



(a)

Dal punto di vista del processore



(b)

Sistemi multiprogrammati

- Più programmi sembrano essere eseguiti "contemporaneamente".
- In realtà in esecuzione c'è sempre ***un solo*** processo.
  - Ma, se l'alternanza è molto frequente, si ha un'idea di simultaneità.

Sistemi multiprogrammati

- Un processo può trovarsi in tre diversi stati: in ***esecuzione***, in ***attesa***, ***pronto***

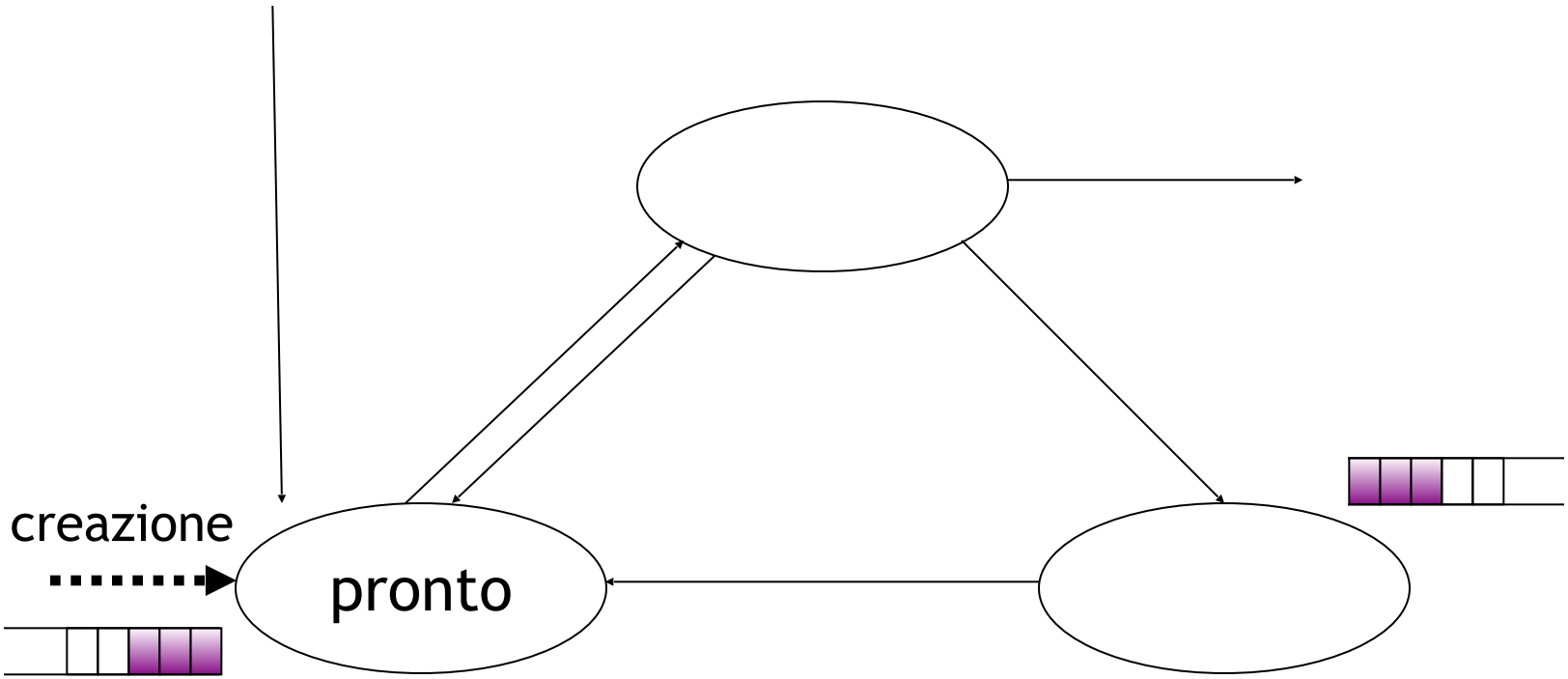
Quando sta utilizzando il processore

Quando è potenzialmente in condizione di poter utilizzare il processore che è occupato da un altro processo

Quando è in attesa del verificarsi di un evento esterno

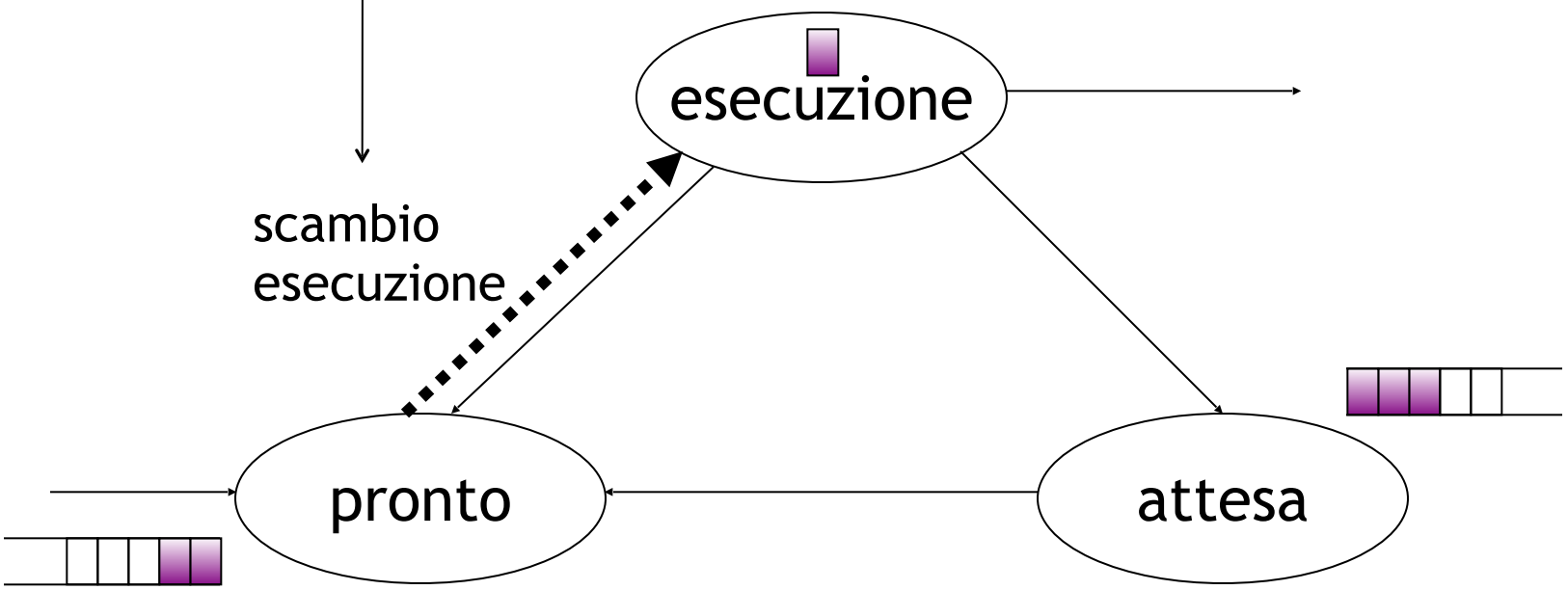
Sistemi multiprogrammati

Quando un processo viene creato viene messo nello stato di **pronto**: in tale stato rimane fino a quando non arriverà il suo turno.



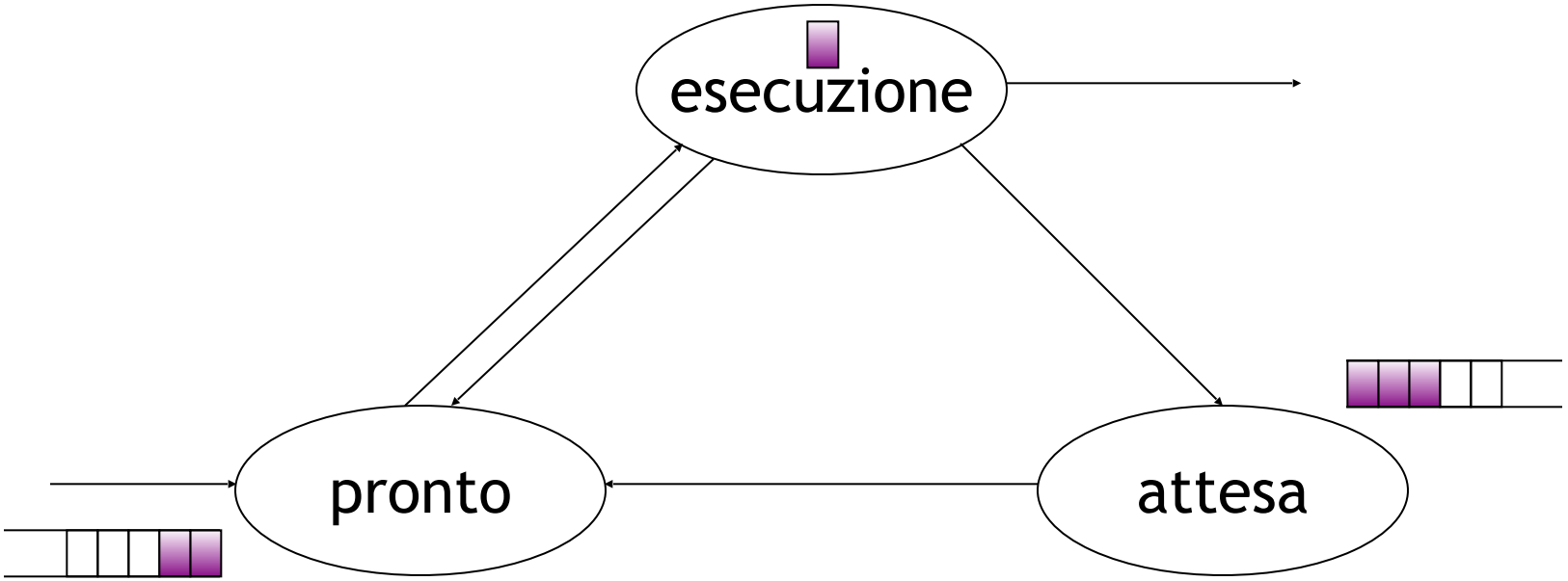
Sistemi multiprogrammati

Quando il processore si libera, il primo processo **pronto** viene mandato in **esecuzione.**



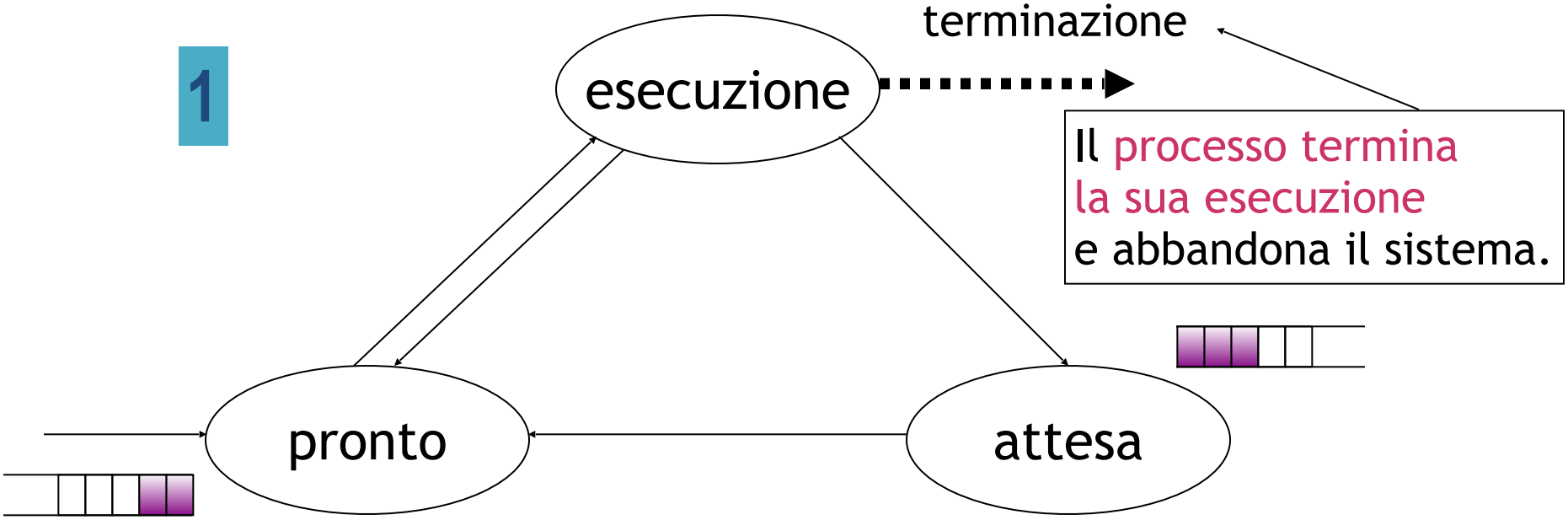
Sistemi multiprogrammati

Un processo può abbandonare lo stato di **esecuzione** per tre diverse ragioni.



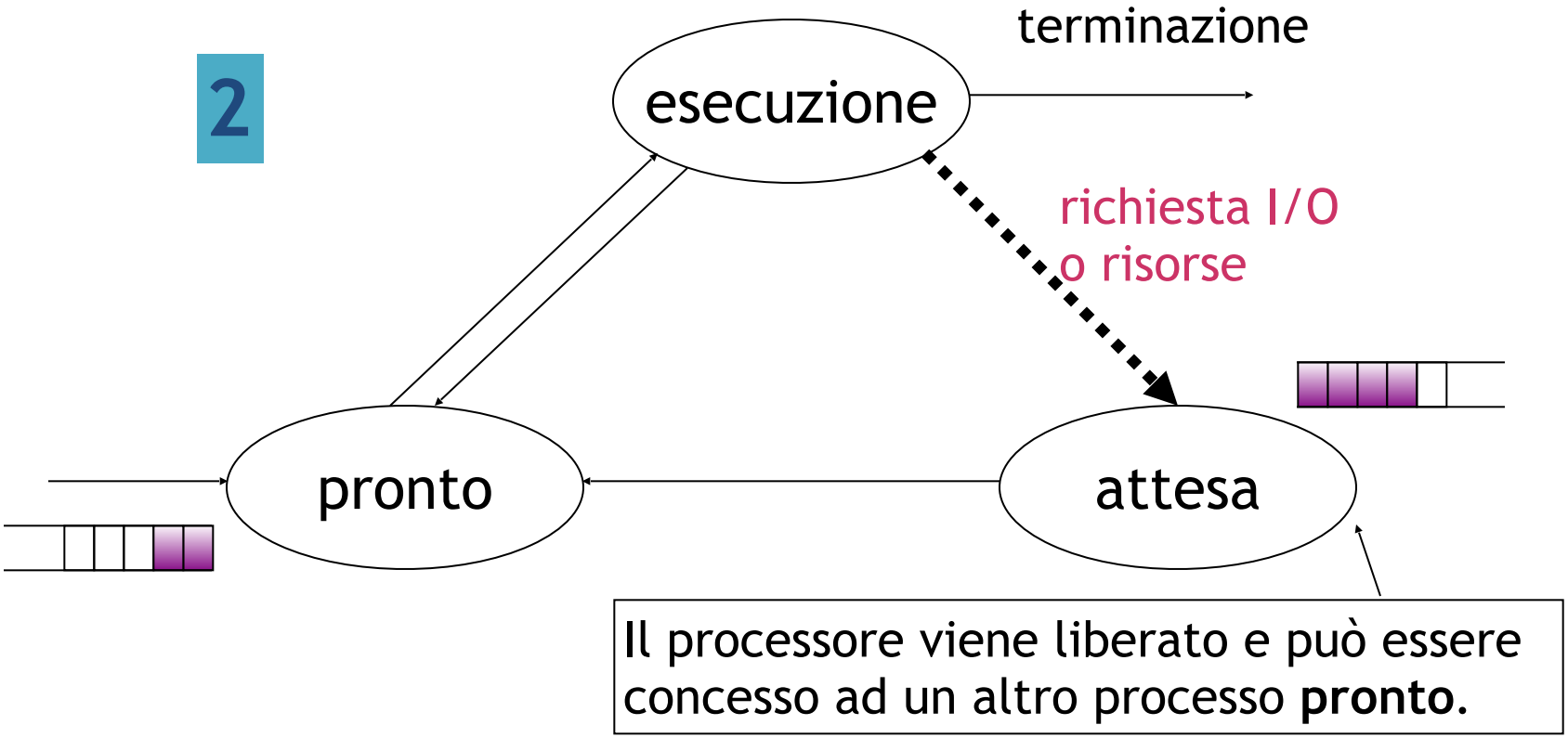
Sistemi multiprogrammati

Un processo può abbandonare lo stato di **esecuzione** per tre diverse ragioni.



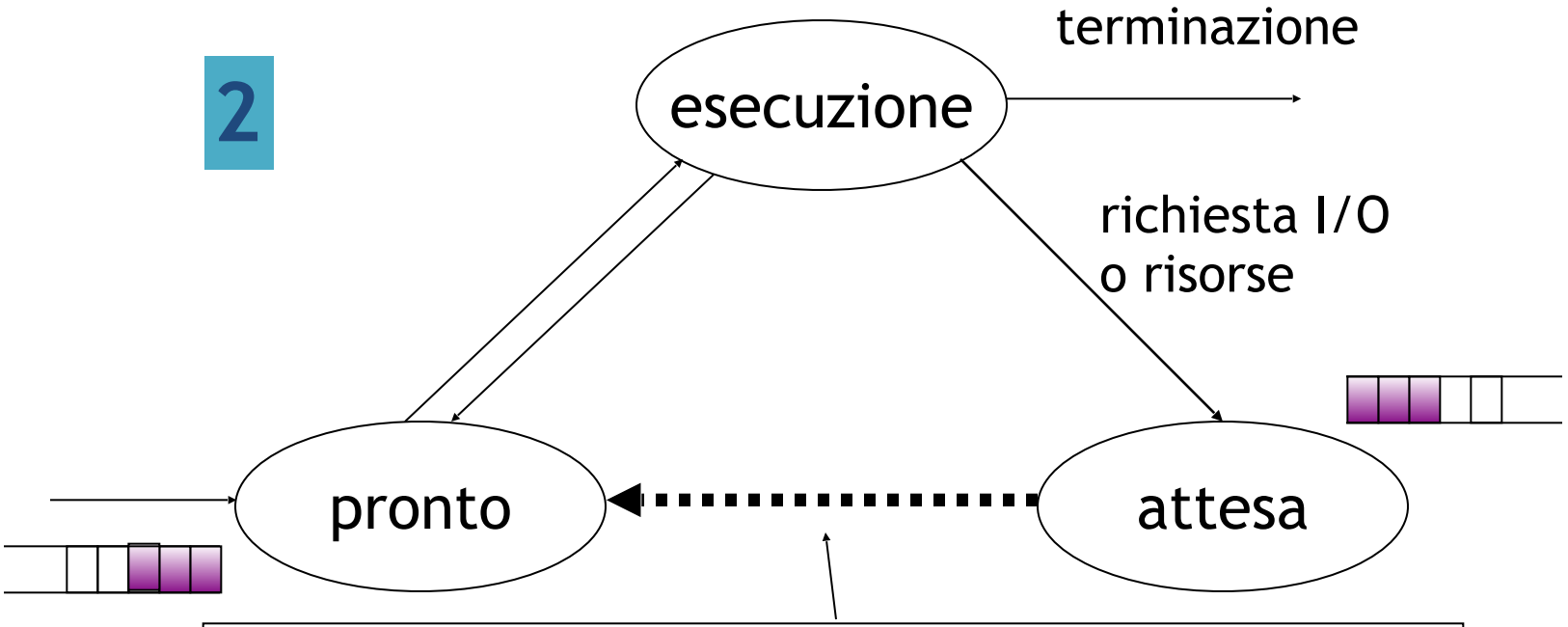
Sistemi multiprogrammati

Un processo può abbandonare lo stato di **esecuzione** per tre diverse ragioni.



Sistemi multiprogrammati

Un processo può abbandonare lo stato di **esecuzione** per tre diverse ragioni.

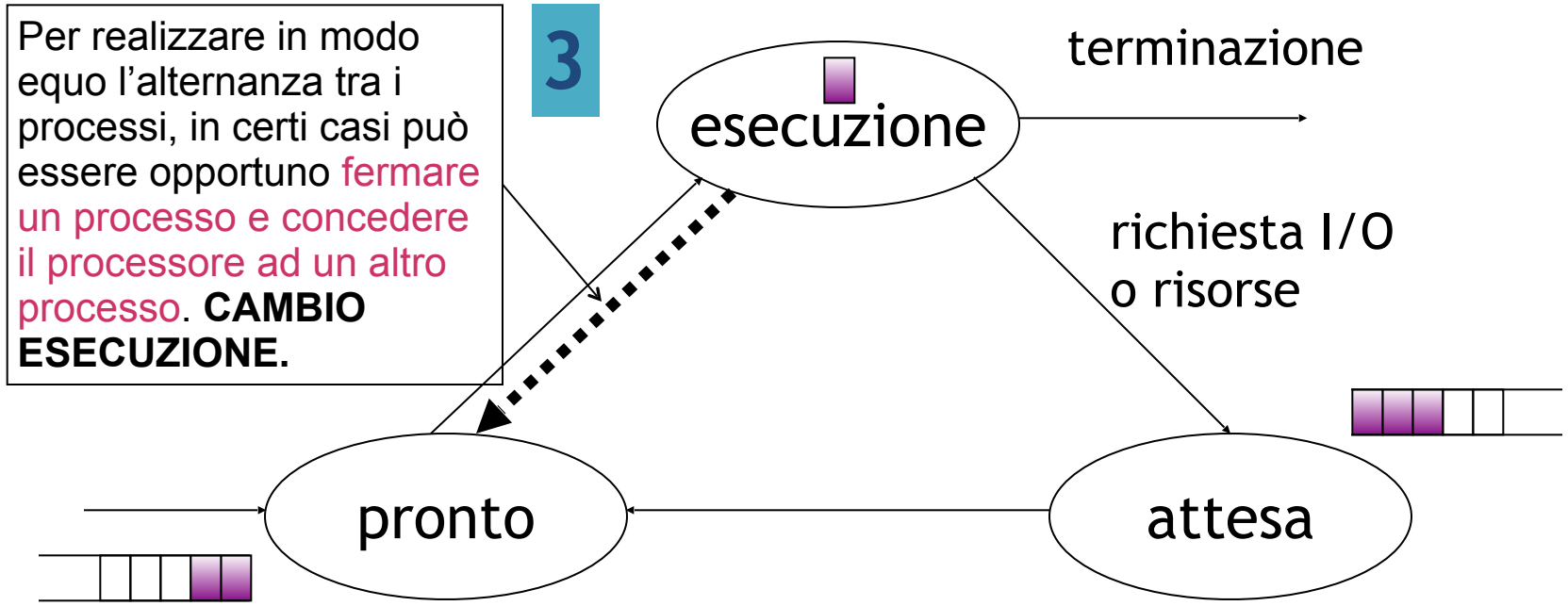


Una volta che l'evento atteso si è verificato, il processo tornerà ad essere **pronto** e dovrà aspettare il suo turno.

Sistemi multiprogrammati

Un processo può abbandonare lo stato di **esecuzione** per tre diverse ragioni.

Per realizzare in modo equo l'alternanza tra i processi, in certi casi può essere opportuno fermare un processo e concedere il processore ad un altro processo. **CAMBIO ESECUZIONE.**



Sistemi multiprogrammati

- In quali casi è opportuno fermare un processo e concedere il processore ad un altro processo?
  - Se un processo non si ferma mai in attesa di input/ output o di una risorsa.
  - Se più utenti vogliono usare il computer.
- In questi casi è necessario far sì che il processore sia distribuito più equamente tra i processi dello stesso utente e di utenti diversi.
- Si parla di *scheduling* del processore.

Politiche di scheduling del processore (1)

Due parametri di riferimento :

- Massimizzare il grado di utilizzazione del processore, ossia fare in modo che il processore rimanga attivo per maggior
tempo possibile;

- Minimizzare il tempo di attesa dei processi o, nel caso di processi interattivi, minimizzare il tempo di risposta agli utenti.

Politiche di scheduling del processore (2)

- Massimizzare il numero di programmi che vengono eseguiti nell'unità di tempo, cioè il **throughput** del sistema, in modo tale che il processore svolga il massimo lavoro possibile

- Minimizzare il tempo di esecuzione dei processi, detto anche tempo di **turnaround**, cioè il tempo che intercorre tra l'istante in cui un processo viene creato e quello in cui esso termina

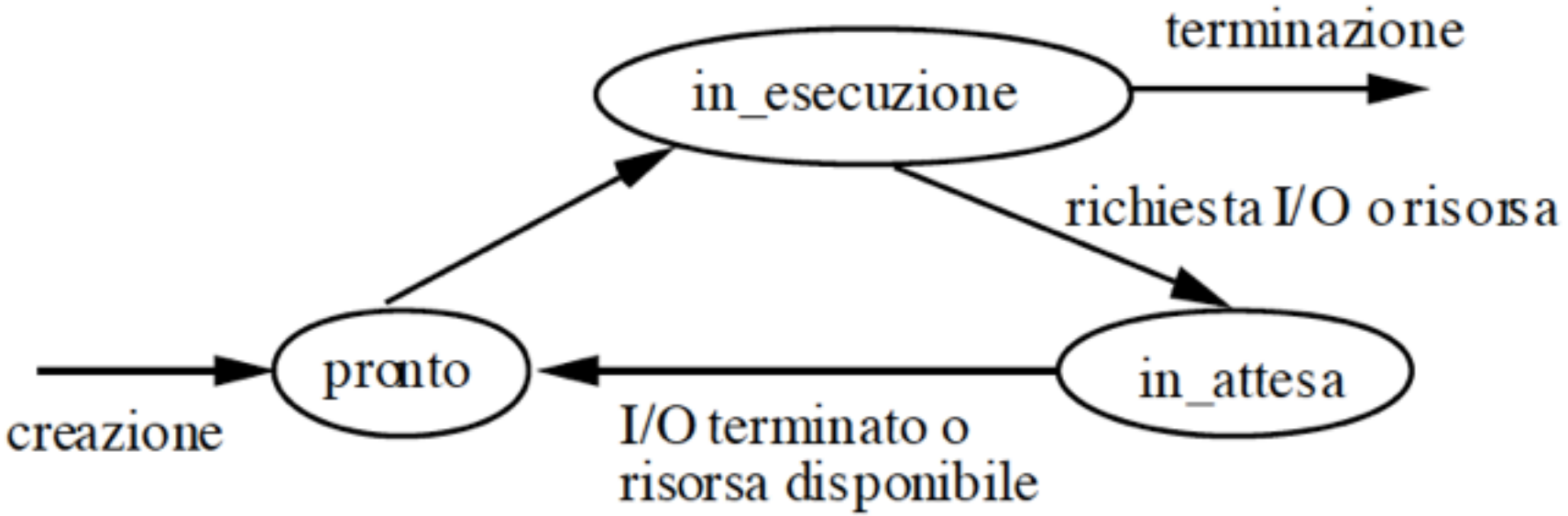
Preemption

- Politiche non-preemptive, in cui il processo in esecuzione può essere sostituito solo se si ferma volontariamente, magari in attesa di un'operazione di I/O;

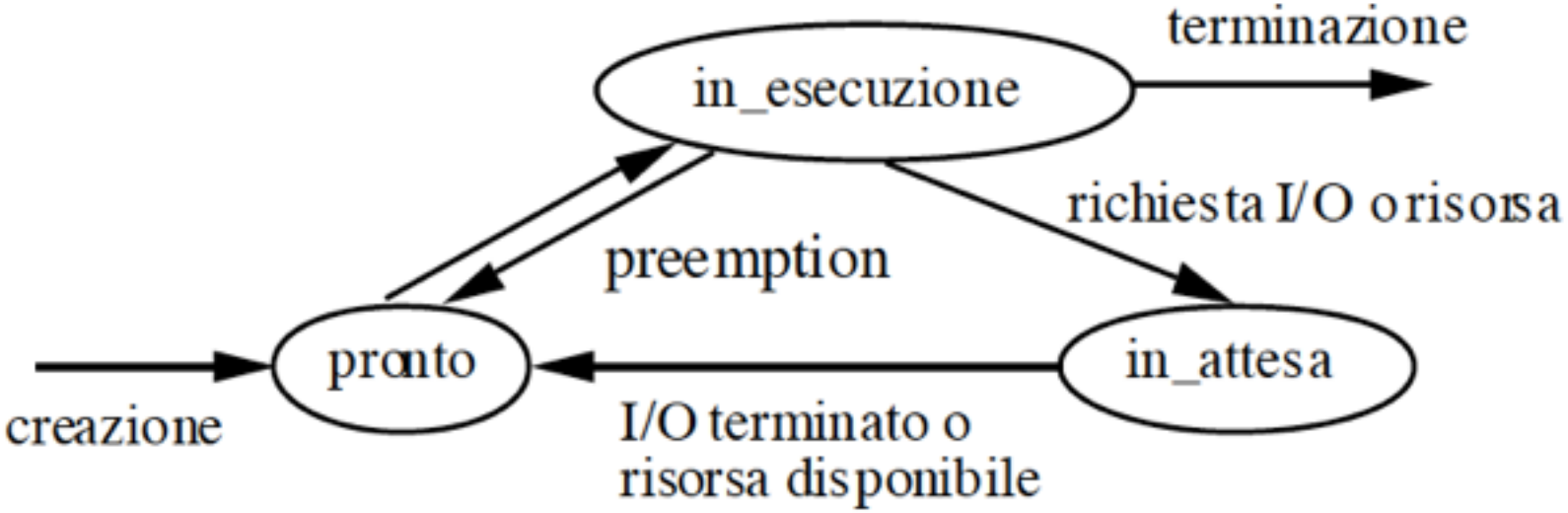
- Politiche preemptive, in cui il sistema operativo può decidere di bloccare un processo per mandarne in esecuzione un altro.

(preemptive = **with prior right**)

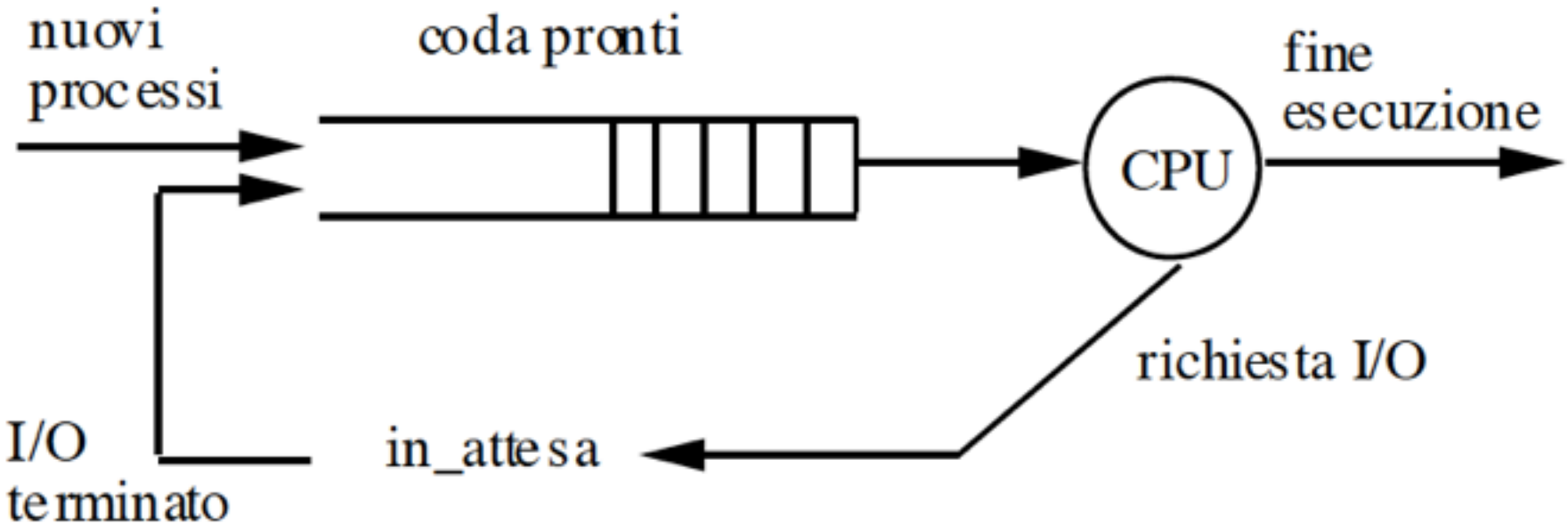
*Gli stati **non-preemptive***



*Gli stati nel **preemptive***



Politica di scheduling FCFS (FIFO)



First Come First Served (FCFS) o First In First Out (FIFO)

Si tratta di una politica **non-preemptive** in cui i processi sono eseguiti nell'ordine in cui vengono sottomessi al sistema.
La coda dei processi pronti viene gestita selezionando il prossimo processo da mandare in esecuzione dall'inizio della coda, e inserendo in fondo alla coda i processi che diventano pronti.

Quindi, ogni volta che il processore è libero, viene selezionato e mandato in esecuzione il primo processo della coda.

Shortest Job First (SJF)

È una politica non-preemptive. Lo schema è simile a quello della politica FCFS ma, invece di selezionare il primo processo della coda dei pronti, viene selezionato quello che richiede meno tempo per terminare o per fermarsi sulla prossima operazione di attesa.

Shortest Job First (SJF)

- Si può dimostrare che eseguendo i processi nell'ordine crescente di tempo, si ottiene il minimo tempo medio di turnaround;
  - questo dipende dal fatto che, se si eseguono prima i processi più corti, il tempo medio di attesa dei processi nella coda dei pronti è minimo.
- Purtroppo, non si conosce a priori il tempo necessario ad un processo per terminare e questo parametro può solo essere stimato (es., dimensioni e numero istruzioni)

Shortest Remaining Time First (SRTF)

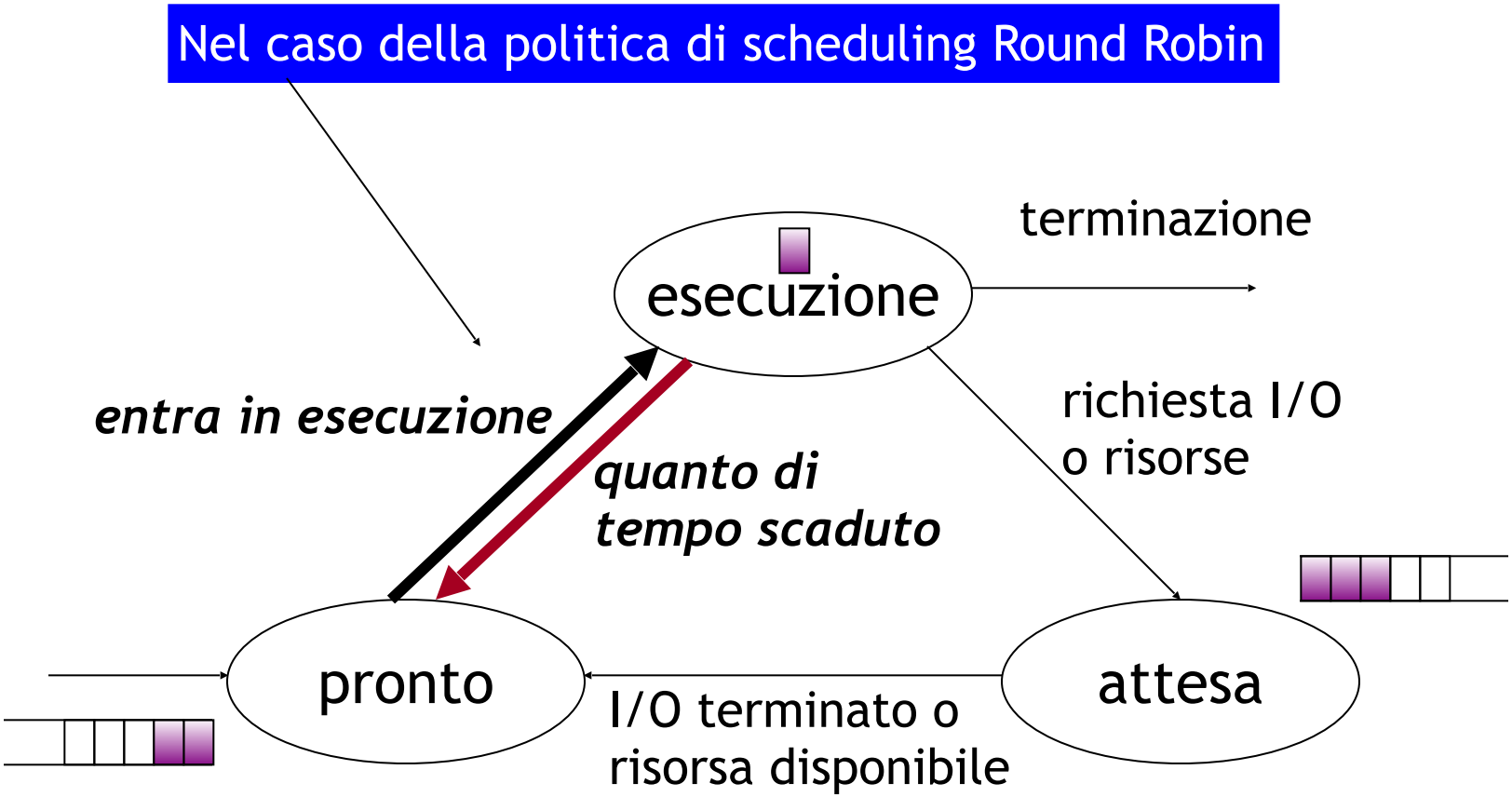
Si tratta di una politica preemptive che costituisce una variante della politica SJF: si manda in esecuzione il processo più breve, ma in ogni istante esso può essere interrotto se, nel frattempo, è diventato pronto un processo che richiede meno tempo per terminare.

Il confronto deve essere fatto tra il tempo di esecuzione del nuovo processo e quello che manca al processo in esecuzione per terminare.

Esempio di scheduling: Round Robin

- Ad ogni processo viene assegnato un *quanto* di tempo del processore (***time slice***).
- Terminato il quanto di tempo, il processo viene sospeso e rimesso nella coda dei processi pronti (al fondo della coda).
- Il processore viene assegnata ad un altro processo pronto.
- Un processo può usare *meno* del quanto che gli spetta se deve eseguire operazioni di I/O oppure ha terminato la sua computazione.

Esempio di scheduling: Round Robin



Round Robin

Un aspetto critico della politica Round Robin è la scelta della durata del quanto di tempo.

- Se è troppo breve, si ha uno scambio frequente, e quindi una contemporaneità di esecuzione più marcata, ma si hanno anche frequenti context switch con la conseguente perdita di tempo.

- Se il quanto di tempo è troppo lungo, si rischia di cadere nella politica FCFS perché nessun processo viene mai interrotto.

- La politica Round Robin (o alcune sue varianti più sofisticate) è quella adottata nei sistemi operativi interattivi

Fair e unfair

Distinzione tra politiche eque (fair) e politiche non eque (unfair).

Una politica si dice equa se garantisce che ogni processo prima o poi verrà selezionato e andrà in esecuzione; le politiche FCFS e Round Robin sono esempi di politiche eque.

Nel caso di politiche non eque, invece, c'è il rischio che un processo pronto sia costretto ad aspettare in eterno perché non arriva mai il suo turno per l'esecuzione.

Sistemi multi-utente, multi-programmati

- Più utenti possono usare allo stesso tempo il computer:
  - … perché il processore viene assegnato periodicamente ai processi dei vari utenti (per esempio ogni 10 o 100 millisecondi).
- All'aumentare del numero di processi e del numero di utenti le ***prestazioni*** del sistema possono ***degradare***.

Gestione dei processi

- Per gestire un insieme di processi "*contemporaneamente*" attivi il sistema operativo mantiene la ***tabella di processi***.
  - Per ogni processo vi è un ***descrittore*** nel quale sono memorizzate informazioni come:
    - L'identificatore del processo.
    - L'identificatore dell'utente proprietario.
    - Lo stato del processo.
    - Ecc.
- Queste informazioni servono per realizzare l'operazione di ***cambio di contesto*** (context switching)
  - Quando un processo rilascia il processore, le informazioni sul suo stato vengono memorizzate nel suo descrittore all'interno della tabella dei processi.
  - In questo modo, quando tornerà nuovamente in esecuzione, il processo potrà ***ripartire dal punto in cui era stato interrotto***.

Cosa fa il sistema operativo?

1. Permette di avere in esecuzione più programmi contemporaneamente:
   - Gestione del processore → come "distribuire" il suo tempo fra programmi diversi?
   - **Gestione della memoria principale → come "distribuire" il suo spazio fra programmi diversi?**
2. Gestisce la memoria secondaria.
3. Permette agli utenti di interagire con il computer:
   - Gestione del sistema operativo stesso:
     - Installare (e disinstallare) programmi .
     - Configurare il computer e il sistema operativo.
   - Attività vera e propria:
     - Mandare in esecuzione un programma con cui creare nuovi documenti o elaborare documenti esistenti.
     - Eliminare e rinominare documenti creati e riorganizzarne la disposizione.
     - Utilizzare le periferiche.

Parte III

Software e Sistema Operativo

(Come usiamo un computer?)

Sul libro...

- Console, Ribaudo, Avalle, Carmagnola, Cena: *Introduzione all'Informatica*.
- **Capitolo 4: tutto**
- **Capitolo 5:**
  - **Introduzione → tutta**
  - **5.1 → tutto**
  - **5.2 → tutto**
  - **5.3 → fino al 5.3.1 escluso ma:**
    - Va saputa la parte su Round Robin
    - **5.4 → tutto ma:**
      - … di Partizioni Fisse, Partizioni Variabili, Segmentazione, Paginazione, Swapping e Demand Paging è sufficiente sapere su che principio si basano e quali sono le differenze (ciò che è svolto sui lucidi).
      - Va invece saputa bene la parte iniziale di 5.4.1 e 5.4.2 e la differenza fra Allocazione Contigua e Allocazione Non Contigua.
  - **5.5 → tutto**
  - **5.6 → no**
  - **5.7 → tutto**

Gestione della memoria

- Due problemi fondamentali:
  - **Multiprogrammazione:**
    - Più programmi in memoria contemporaneamente.
    - Come lasciare a ciascuno il suo spazio senza che si modifichino i dati a vicenda per errore?
  - Programmi e/o dati di **grandi dimensioni**:
    - Alcuni programmi potrebbero essere troppo grandi per essere contenuti in RAM.
    - Oppure i dati che essi elaborano potrebbero essere troppo grandi.
    - Esiste un modo che permetta a programmi in queste condizioni di funzionare ugualmente?

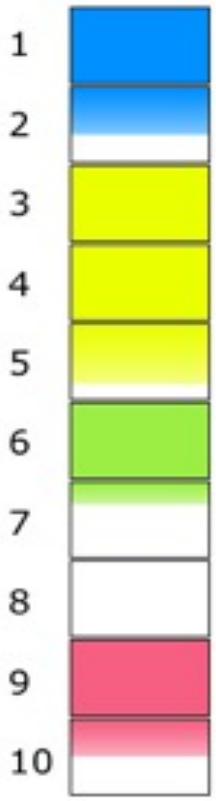
Partizioni multiple con allocazione contigua

- Per tenere più programmi in memoria contemporaneamente:
  - Si divide la memoria in sezioni chiamate **partizioni**.
  - In ogni partizione: un programma con i suoi dati.
  - **Allocazione contigua:** ciascuna partizione deve contenere interamente un programma con i suoi dati
  - Una partizione consiste di alcuni "blocchi" *consecutivi* della memoria principale (i blocchi in questo contesto sono diverso di un blocco della memoria secondario).
- Una partizione è individuata da:
  - Un indirizzo di inizio partizione (**base**).
  - Un indirizzo di fine partizione (**limite**).

Partizioni fisse

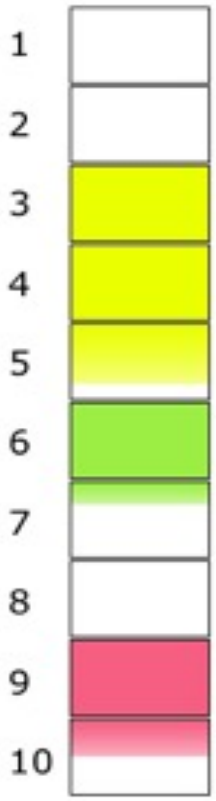
- Partizioni **fisse**:
  - Dimensione delle partizioni decisa a priori una volta per tutte.
  - Problema: **spreco di spazio** di memoria.
    - All'**interno** di ciascuna partizione (perché il processo probabilmente sarà un po' più piccolo della partizione) → **Frammentazione interna**.
    - **Complessivo** (perché potrei avere tante partizioni libere tutte troppo piccole per contenere il mio processo, che però potrebbe stare in memoria se la partizione fosse unica) → **Frammentazione esterna**.

Frammentazione interna



- Il processo P_1 occupa il blocco 1 e parte del blocco 2.
- Il processo P_2 occupa i blocchi 3 e 4 e parte del blocco 5.
- Il processo P_3 occupa il blocco 6 e parte del blocco 7.
- Il processo P_4 occupa il blocco 9 e parte del blocco 10.
- C'è frammentazione interna nei blocchi 2,5,7 e 10.

Frammentazione esterna



- Il processo P_1 termina.
- Il ("nuovo") processo P_5 occupa 3 blocchi e deve essere caricato in memoria. Ci sarebbero 3 blocchi liberi (1, 2 e 8) ma non sono contigui: frammentazione esterna.

Partizioni variabili

- Partizioni **variabili**:
  - Dimensione delle partizioni decisa al momento del caricamento del processo.
    - Ovviamente più difficile da gestire della precedente, ma più flessibile.
    - La **frammentazione esterna** esiste comunque.

Allocazione non contigua

- Le Partizioni sono in generale **poco flessibili**.
  - Se non azzecco la dimensione giusta per la partizione il programma potrebbe non riuscire a girare!
  - La memoria potrebbe avere abbastanza spazio per un programma, ma non contiguamente.
    - Tanti spazi liberi ciascuno di dimensioni troppo piccole.
- Sistema **più flessibile**: **allocazione non contigua**.
  - I programmi (e i relativi dati) vengono **spezzettati**.
  - Si deve in pratica memorizzare per ciascun pezzo di programma dove esso viene a trovarsi in memoria.

2 Tecniche: Segmentazione vs. Paginazione

- **Segmentazione**:
  - Il processo viene suddiviso in base ai contenuti.
    - Posso separare la parte contenente le istruzioni da quella contenente i dati.
    - Possono esserci analoghi criteri per suddividere ulteriormente, ad es. tipi di dati o pezzi di programma che vengono usati in momenti diversi.
  - Problema: gestione di segmenti di dimensioni diverse piuttosto complicata.
- **Paginazione**:
  - Il processo viene diviso in **pagine** di uguale dimensione (solitamente tra 1 KB e 4 KB).
  - La memoria viene divisa in **frame** *[inglese, significa cornice]* (simili alle partizioni) della stessa dimensione.
    - Ad ogni pagina è assegnato un *frame*.
    - Una zona particolare della memoria (tabella delle pagine) contiene le associazioni fra pagine e *frame*.
  - **Vantaggio: tutti gli oggetti da gestire sono dalla stessa dimensione.**

Memoria virtuale

- Permette di avere in esecuzione un insieme di processi la cui dimensione complessiva supera la capacità della memoria.
- Due tecniche principali:
  - **Swapping**
  - **Paginazione (a richiesta) (saltare, non la vediamo)**
- Entrambe basate sull'idea di usare la memoria secondaria come deposito temporaneo per ciò che non sta in RAM.
  - Ovviamente questo causa dei rallentamenti nell'esecuzione.

Swapping

- I processi sono in esecuzione uno alla volta:
  - Quando un processo non è in esecuzione posso **toglierlo temporaneamente** dalla RAM e metterlo in memoria secondaria.
  - Quando ritorna in esecuzione lo riporto in RAM spostando eventualmente un altro processo.

Trashing

- Se la memoria è troppo piccola per l'insieme di programmi caricato il processore va in **trashing** (thrashing in inglese):
  - Passa più tempo a spostare i processi dalla memoria secondaria alla RAM (e vice versa) che a eseguire i programmi.
  - Il computer rallenta tantissimo e non è più utilizzabile.
  - Si capisce che il computer è in trashing perché sebbene sembra non stia facendo nulla il disco è continuamente utilizzato.
  - Unica soluzione: chiudere uno o più programmi.

Cosa fa il sistema operativo?

1. Permette di avere in esecuzione più programmi contemporaneamente:
   - Gestione del processore → come "distribuire" il suo tempo fra programmi diversi?
   - Gestione della memoria principale → come "distribuire" il suo spazio fra programmi diversi?
2. **Gestisce la memoria secondaria**.
3. Permette agli utenti di interagire con il computer:
   - Gestione del sistema operativo stesso:
     - Installare (e disinstallare) programmi.
     - Configurare il computer e il sistema operativo.
   - Attività vera e propria:
     - Mandare in esecuzione un programma con cui creare nuovi documenti o elaborare documenti esistenti.
     - Eliminare e rinominare documenti creati e riorganizzarne la disposizione.
     - Utilizzare le periferiche.

Gestione della memoria secondaria

- La memoria secondaria serve per memorizzare programmi e dati in modo permanente.
  - Il concetto centrale è quello di **file** (= archivio, dossier).
- **File** → insieme di informazioni omogenee che vengono memorizzate insieme e a cui viene associato un nome che permetta di reperirle.
  - File di programma → contengono istruzioni che possono venire eseguite dal processore.
  - File di dati → detti anche documenti, contengono i dati dell'utente.
- **File system** → parte del sistema operativo che si occupa di gestire e strutturare le informazioni memorizzate sulla memoria secondaria.

File system

- Il sistema operativo si occupa di registrare la posizione di tutti i file all'interno della memoria secondaria
- Inoltre deve fornire una visione astratta dei file su disco in modo che l'utente abbia la possibilità di:
  - identificare ogni file con un nome astraendo dalla sua posizione nella memoria
  - avere un insieme di operazioni per lavorare sui file
  - strutturare i file, organizzandoli in sottoinsiemi secondo le loro caratteristiche, per avere una visione "ordinata" delle informazioni sul disco
  - effettuare l'accesso alle informazioni mediante operazioni ad alto livello, che non tengono conto del tipo di memorizzazione. Esempio: si deve accedere allo stesso modo ad un file memorizzato sul disco rigido oppure su un CD-ROM

Dal punto di vista dell'utente

- **File di programma:**
  - Un programma è solitamente fatto da più file.
    - Uno di questi è chiamato "principale" o "eseguibile".
  - **Installazione** (copiatura "ordinata" dei file che costituiscono il programma, registrazione del nuovo programma presso il sistema operativo).
  - **Disinstallazione** (cancellazione dei file e de-registrazione).
  - **Esecuzione** (caricamento in memoria del file principale del programma e creazione del processo corrispondente).
- **Documenti:**
  - A seconda del tipo di documento (immagine, testo, etc.) esso potrà essere **creato** e **elaborato** con programmi specifici.
  - Il *file system* permette inoltre di intervenire sulla **organizzazione** dei documenti.
    - In realtà permette di intervenire anche sulla organizzazione dei file di programma, ma conviene lasciarli stare!

Organizzazione dei file

- I file devono essere organizzati in memoria secondaria perché sia facile reperirli!
  - Immaginiamo ciascun file come un documento cartaceo e la memoria secondaria come un armadio: non è una buona idea impilare tutti i documenti nell'armadio in modo casuale!
- Il primo passo è associare a ciascun file una **denominazione** che permetta di identificarlo univocamente.
- Le denominazioni dei file hanno solitamente questa forma:

NOME.EXT

  - Il **nome** è scelto da chi crea il file, e dovrebbe permettergli di ricordare con facilità cosa esso contenga.
  - L'**estensione** viene solitamente scelta dal programma con cui viene creato il file, e identifica la **tipologia** di informazioni in esso contenute.

Denominazione dei file

- Chi **crea** il file ne sceglie il nome.
  - Alcune regole:
    - Scegliere nomi che abbiano senso (no “a”, “mio”, “documento”...).
    - Non usare nei nomi caratteri speciali (no segni di punteggiatura, simboli, etc.).
  - Successivamente è possibile modificarlo.
    - Non si devono modificare i nomi dei file di programma ma solo dei documenti!
- L’estensione identifica il **tipo** di file, ed è solitamente di tre lettere.
  - “.lib”, “.dll”, “.exe” sono estensioni tipiche dei file di programma.
  - Per i documenti, dipende dal tipo di documento.
    - Es. “.txt” → testo, “.bmp” → immagine bitmap, etc.
  - È il programma con cui creiamo i documenti che stabilisce l’estensione, solitamente è meglio non modificarla.
    - Modificare l’estensione non cambia il tipo di documento.

Organizzazione dei file

- Se i file sono molti, dargli un nome non è sufficiente a reperirli con facilità (ed è difficile non usare due volte lo stesso nome!).
  - In un armadio useremmo delle cartellette o dei raccoglitori per dividere i documenti in modo logico...
- Anche nel file system esiste il concetto di **cartella** (in inglese, **folder** o **directory**):
  - Una cartella è file speciale che funziona come un contenitore.
  - Una cartella può contenere dei file, e anche delle altre cartelle (chiamate in questo caso **sotto-cartelle**).
    - A differenza del caso dell'armadio, si possono mettere cartelle dentro altre cartelle sino a che si vuole!
  - Due file in due cartelle diverse possono anche avere lo stesso nome: a distinguerli è la **cartella di appartenenza**.

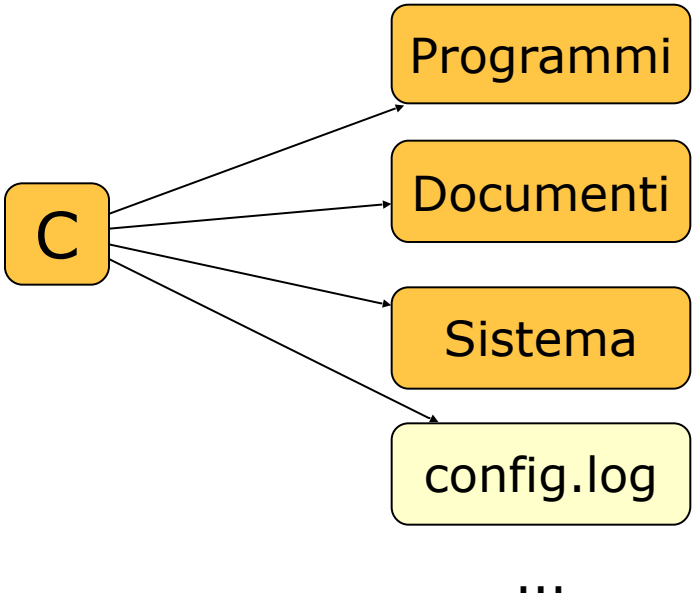
- Questo porta alla **organizzazione gerarchica** dei file.

Organizzazione gerarchica

- La cima della gerarchia è l'unità di memoria secondaria:
  - Windows usa le lettere dell'alfabeto per identificare le diverse unità: es. C → hard disk, D → cd o dvd, E → penna usb,...

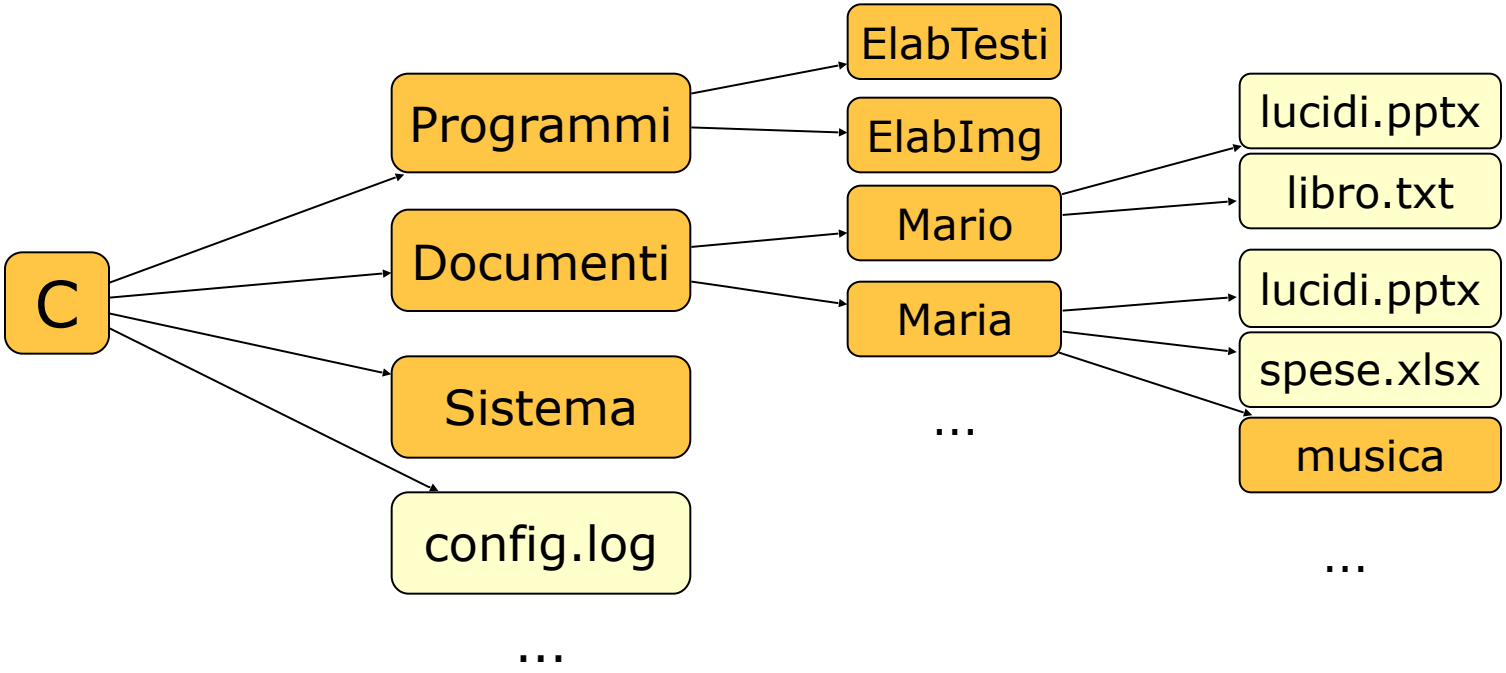
Organizzazione gerarchica

- La cima della gerarchia è l'unità di memoria secondaria:
  - Windows usa le lettere dell'alfabeto per identificare le diverse unità: es. C → hard disk, D → cd o dvd, E → penna usb,...
- All'interno dell'unità si trovano file e cartelle...



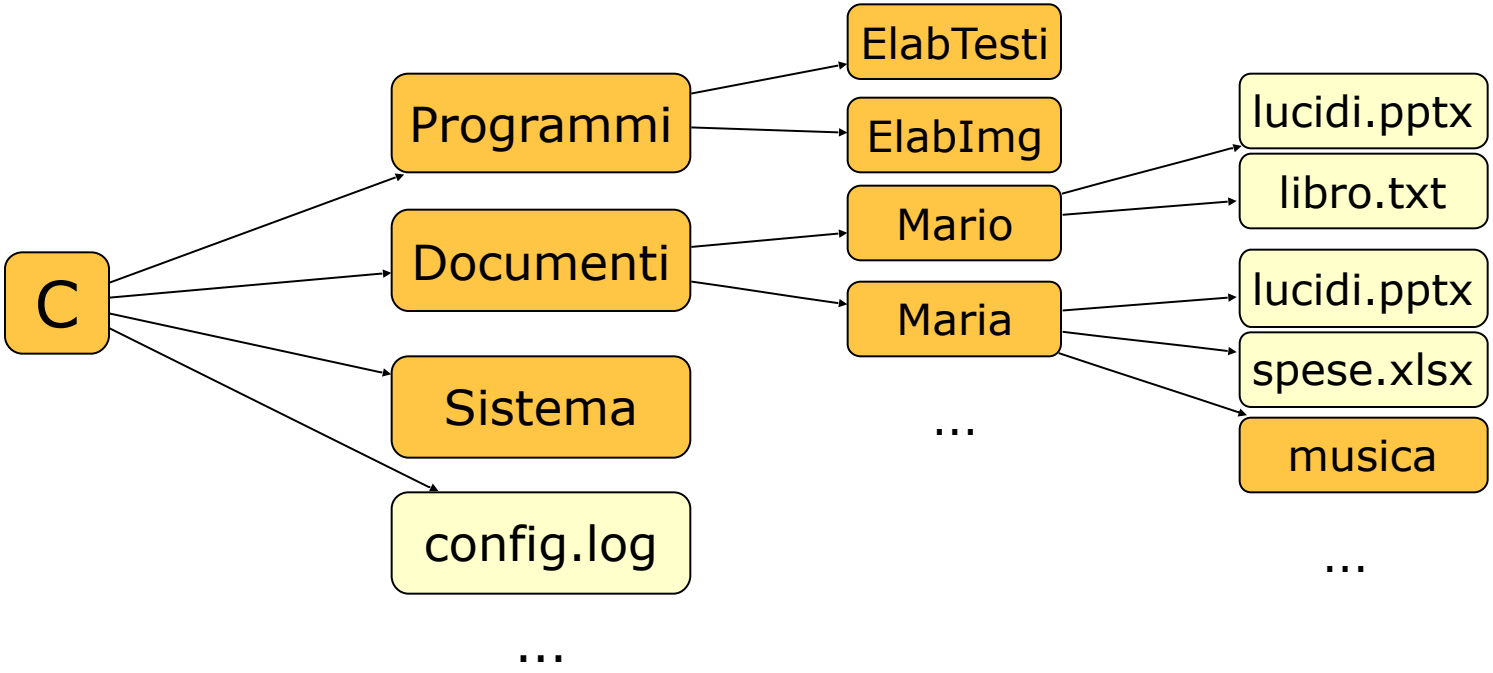
Organizzazione gerarchica

- La cima della gerarchia è l'unità di memoria secondaria
  - Windows usa le lettere dell'alfabeto per identificare le diverse unità: es. C → hard disk, D → cd o dvd, E → penna usb,...
- All'interno dell'unità si trovano file e cartelle...
- ...all'interno dei quali si trovano altri file e cartelle, etc.

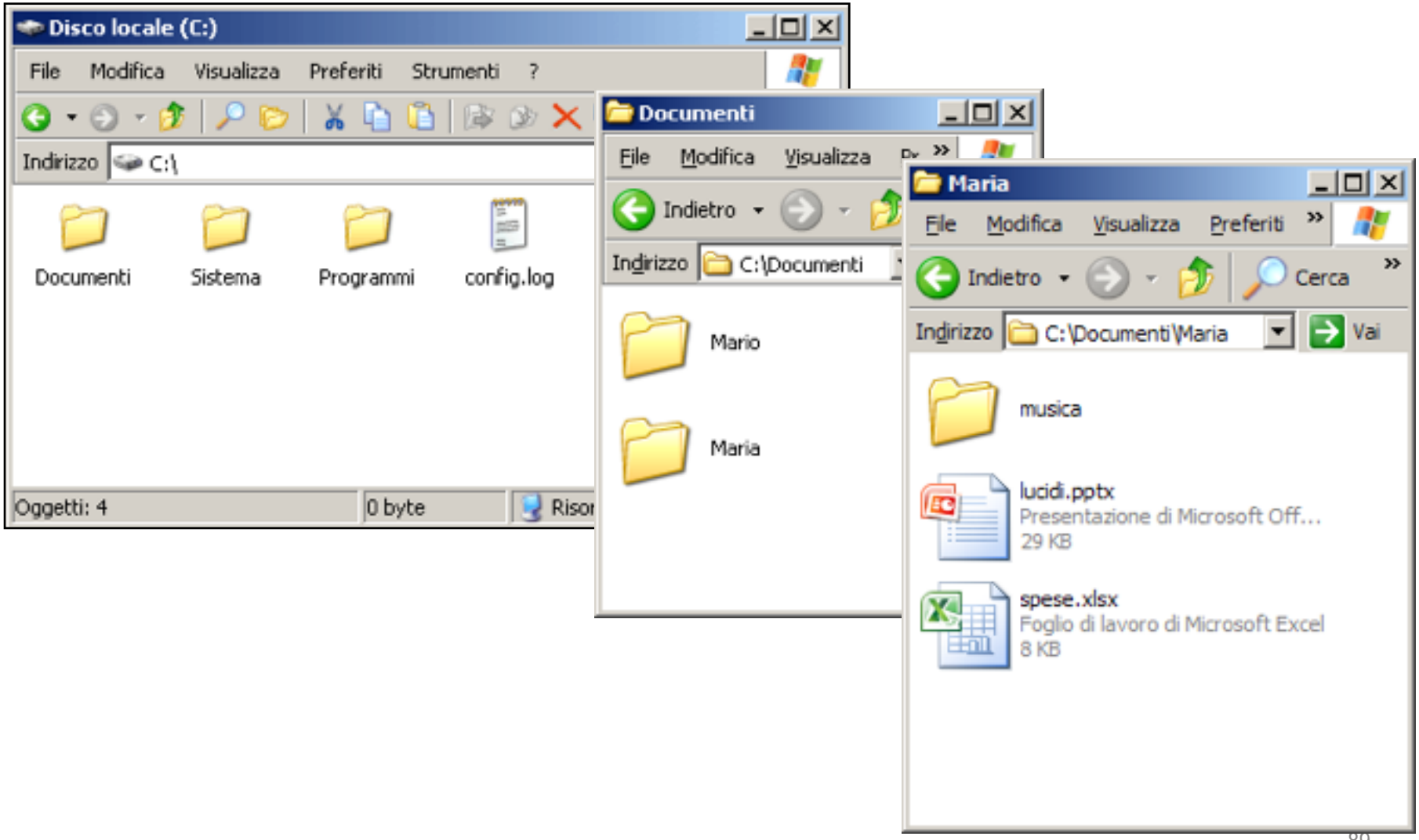


Organizzazione gerarchica

ALBERO DEL FILE SYSTEM

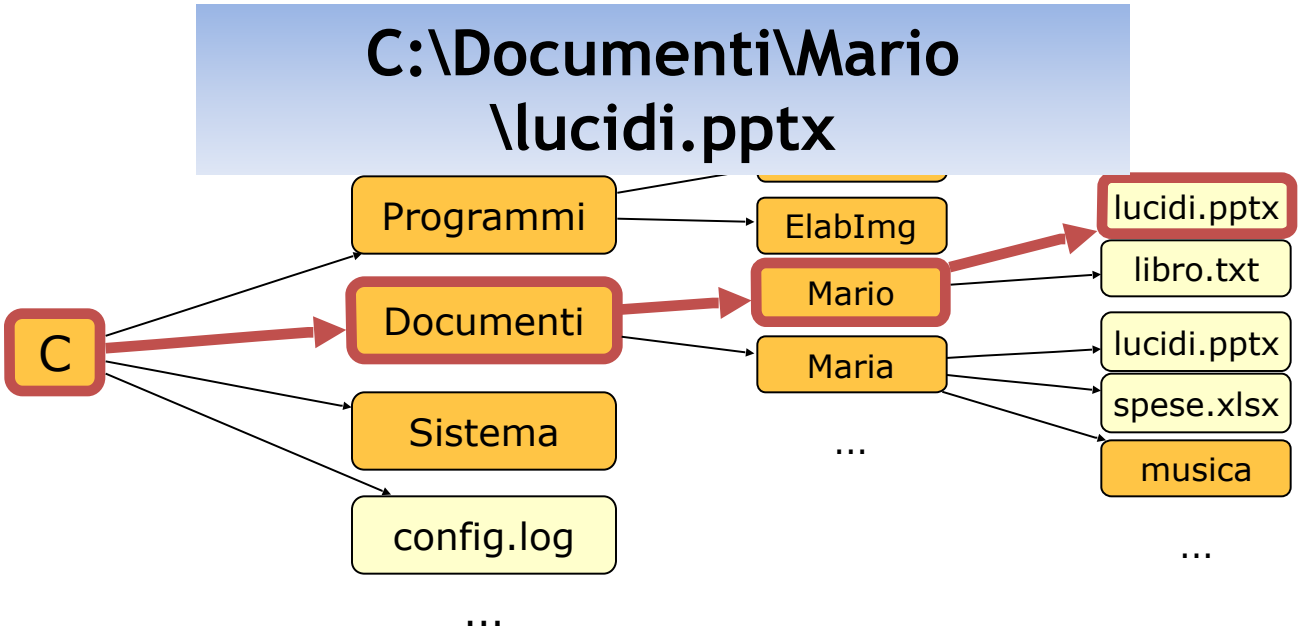


Visualizzazione



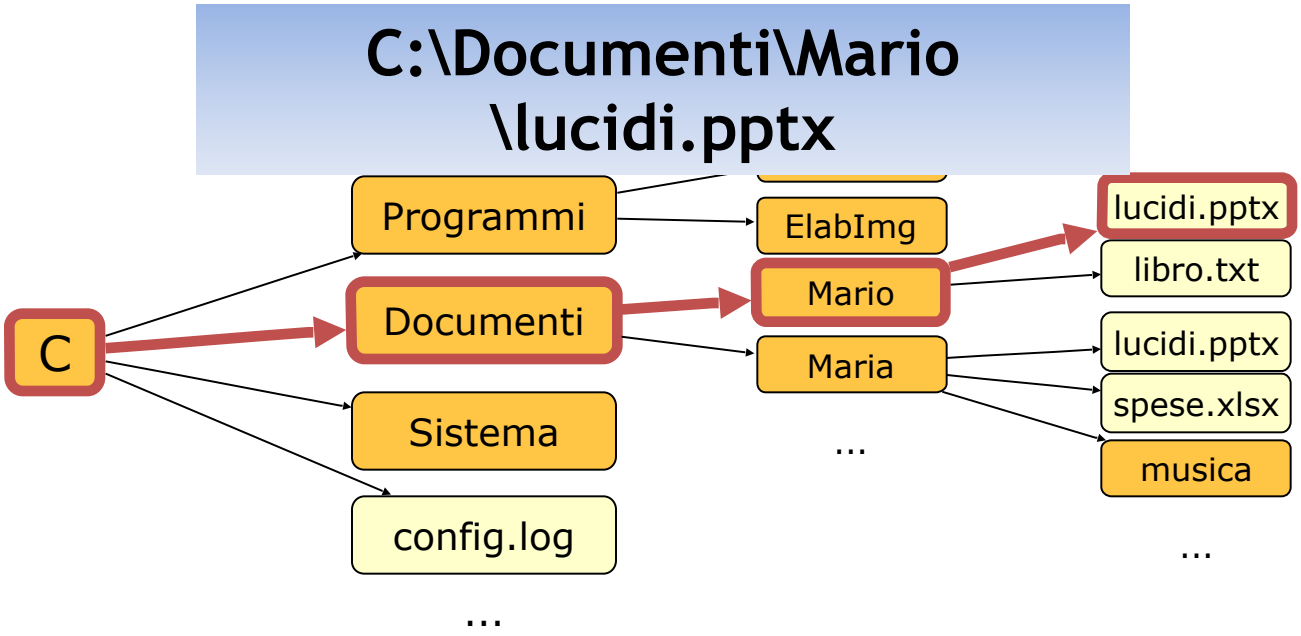
Pathname

- **Pathname** = nome del percorso.
  - Indica il percorso da seguire per raggiungere un file in memoria secondaria
    - Specifica l'unità (es.: "C").
    - la sequenza di cartelle che bisogna aprire per trovare il file (es.: "Documenti" quindi "Mario").
    - Il nome del file (es.: "lucidi.pptx").



Pathname

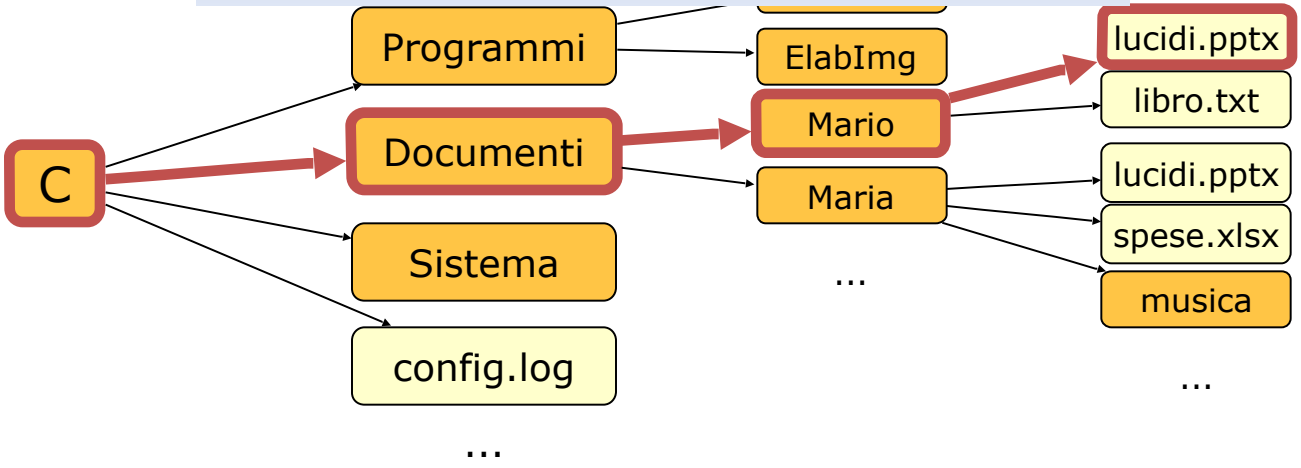
Nome dell'**unità** di memoria secondaria, seguito dai due punti (:).



Pathname

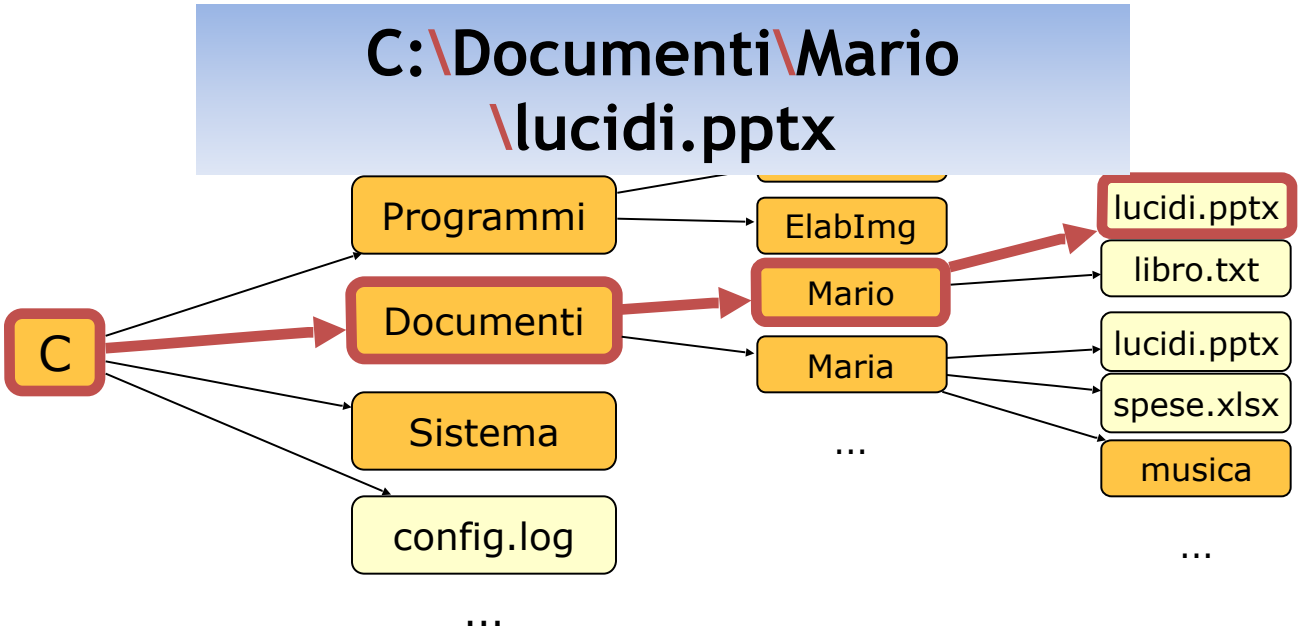
Per finire, nome del **file** vero e proprio.

C:\Documenti\Mario\lucidi.pptx

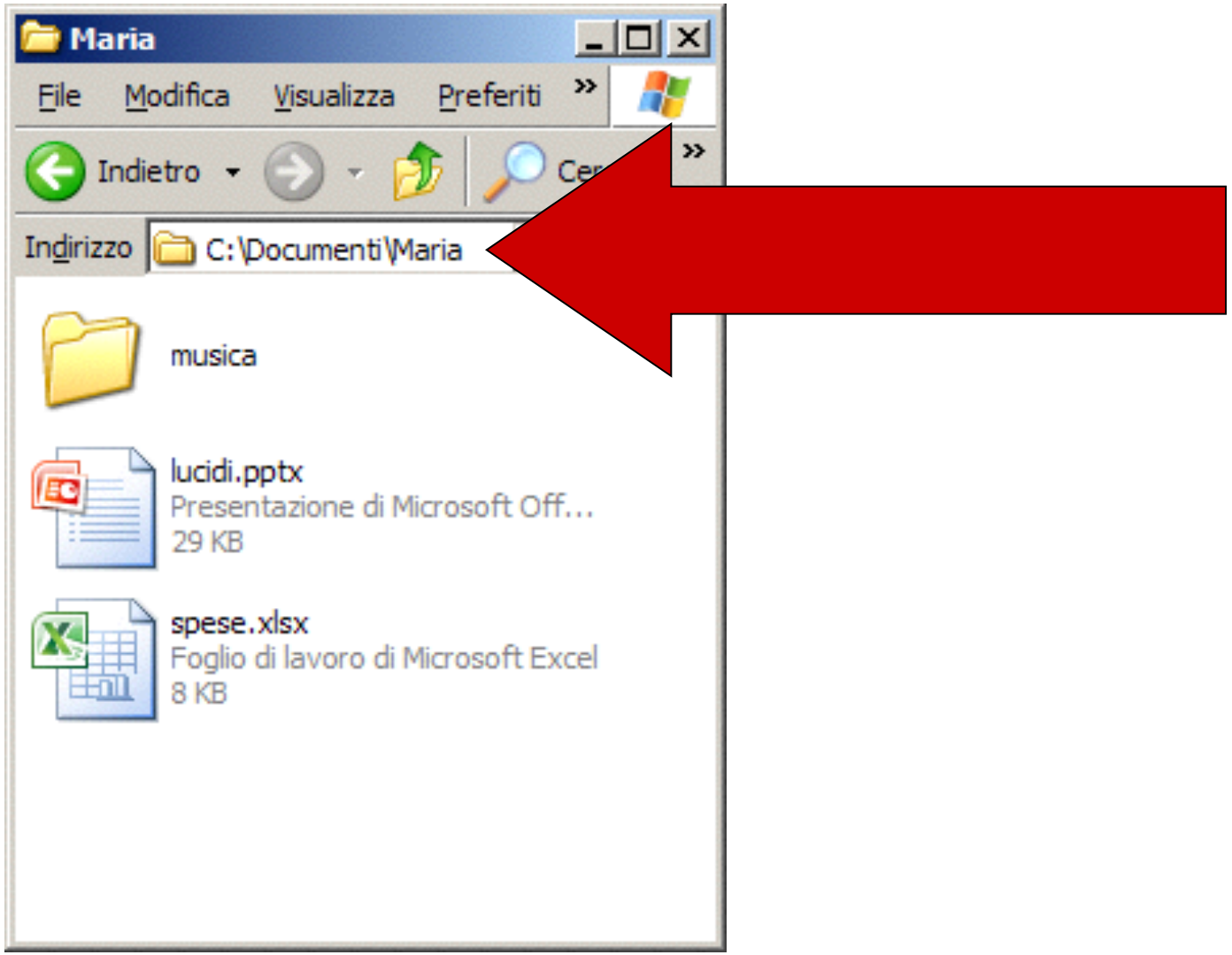


Pathname

I diversi elementi sono separati dal carattere \ chiamato **backslash**.



Formato del pathname



Manipolazione dei file

- Un insieme di operazioni minimale:
  - ***Creazione*** di un file.
  - ***Cancellazione*** di un file.
  - ***Copia*** o ***spostamento*** di un file.
  - ***Visualizzazione*** del contenuto di un file.
  - ***Stampa*** di un file.
  - ***Modifica*** del contenuto di un file.
  - ***Rinomina*** di un file.
  - ***Visualizzazione delle caratteristiche*** di un file.
- Alcune operazioni corrispondono ai programmi applicativi, alcune al sistema operativo.

Operazioni sulle cartelle

- Per organizzare gerarchicamente i file, il sistema operativo deve fornire all'utente un insieme di operazioni per:
  - ***Creare*** una nuova directory.
  - ***Rimuovere*** una directory.
  - ***Rinominare*** una directory.
  - ***Elencare*** il contenuto di una directory.
  - ***Copiare*** o ***spostare*** i file da una directory ad un'altra.
- Le operazioni vengono gestite dal sistema operativo.

Organizzazione fisica dei file

- L'organizzazione del file system vista fino ad ora è di tipo puramente **logico**.
  - Cioè fa vedere come il file system "fa vedere" i propri contenuti all'utente per permettergli di manipolarli.
- Organizzazione **fisica**:
  - I file (e le cartelle) occupano dei **blocchi** su disco.
- Il sistema operativo deve:
  - Sapere quali sono i file memorizzati sul disco
  - Sapere dove si trovano (in quali blocchi) per poterli reperire
  - Ottimizzare lo spazio su disco (cioè evitare di sprecare spazio inutilmente).

Device directory

- Per sapere quali sono i file memorizzati sul disco e dove si trovano …?
- **Device directory** → Una tabella memorizzata in una porzione prefissata del disco che contiene un elenco di **file descriptor**.
  - I file descriptor sono **numerati**.
  - Ciascun file o cartella ha il suo file descriptor che contiene le info su di esso.
    - Per ciascun **file** nome, dimensione, data di creazione e ultima modifica, ... e soprattutto: **dove** si trova su disco.
    - Per ciascuna **cartella** nome, dimensione, data di creazione e ultima modifica, ... e soprattutto: **l'elenco** dei file descriptor (tramite i loro numeri d'ordine) dei file e delle cartelle in essa contenuti.

Device directory

- Esempio: C:\Documenti\Mario\lucidi.pptx

Device directory

- Esempio: **C:**Documenti\Mario\lucidi.pptx
  - Vado nella device directory del disco C; il primo file descriptor è quello della cartella principale.

Device directory

- Esempio: **C:\Documenti**Mario\lucidi.pptx
  - Vado nella device directory del disco C; il primo file descriptor (0) è quello della cartella principale.
  - Cerco nel file descriptor 0 l'elenco dei contenuti della cartella, trovo che "Documenti" corrisponde al file descriptor 7.

Device directory

- Esempio: **C:\Documenti\Mario**lucidi.pptx
  - Vado nella device directory del disco C; il primo file descriptor (0) è quello della cartella principale.
  - Cerco nel file descriptor 0 l'elenco dei contenuti della cartella, trovo che "Documenti" corrisponde al file descriptor 7.
  - Cerco nel file descriptor 7 (che corrisponde a "Documenti") e trovo che "Mario" corrisponde al file descriptor 22.

Device directory

- Esempio: **C:\Documenti\Mario\lucidi.pptx**
  - Vado nella device directory del disco C; il primo file descriptor (0) è quello della cartella principale.
  - Cerco nel file descriptor 0 l'elenco dei contenuti della cartella, trovo che "Documenti" corrisponde al file descriptor 7.
  - Cerco nel file descriptor 7 (che corrisponde a "Documenti") e trovo che "Mario" corrisponde al file descriptor 22.
  - Cerco nel file descriptor 22 (che corrisponde a "Mario") e trovo che "lucidi.pptx" corrisponde al file descriptor 18.

Device directory

- Esempio: **C:\Documenti\Mario\lucidi.pptx**
  - Vado nella device directory del disco C; il primo file descriptor (0) è quello della cartella principale.
  - Cerco nel file descriptor 0 l'elenco dei contenuti della cartella, trovo che "Documenti" corrisponde al file descriptor 7.
  - Cerco nel file descriptor 7 (che corrisponde a "Documenti") e trovo che "Mario" corrisponde al file descriptor 22.
  - Cerco nel file descriptor 22 (che corrisponde a "Mario") e trovo che "lucidi.pptx" corrisponde al file descriptor 18.
  - Cerco nel file descriptor 18 (che corrisponde a "lucidi.pptx") e, dal momento che si tratta di un file, trovo la sua collocazione fisica su disco (ad esempio blocco (30,10)).
  - (30,10) e' un indirizzo virtuale:

Device directory

- Esempio: **C:\Documenti\Mario\lucidi.pptx**
  - Vado nella device directory del disco C; il primo file descriptor (0) è quello della cartella principale.
  - Cerco nel file descriptor 0 l'elenco dei contenuti della cartella, trovo che "Documenti" corrisponde al file descriptor 7.
  - Cerco nel file descriptor 7 (che corrisponde a "Documenti") e trovo che "Mario" corrisponde al file descriptor 22.
  - Cerco nel file descriptor 22 (che corrisponde a "Mario") e trovo che "lucidi.pptx" corrisponde al file descriptor 18.
  - Cerco nel file descriptor 18 (che corrisponde a "lucidi.pptx") e, dal momento che si tratta di un file, trovo la sua collocazione fisica su disco (ad esempio blocco (30,10)).

Allocazione

- Allocazione contigua
- allocazione sparsa linkata
- allocazione sparsa indexata

l'hard disk e' suddiviso in blocchi (es., 512/4KB)

Ogni blocco puo' essere libero o occupato da una porzione di file. Ogni blocco e' identificato da un numero: il suo indirizzo.

Allocazione contigua

- Il file e' memorizzato in una serie di blocchi contigui l'uno all'altro.

E' facile "recuperare" l'intero file

- l'accesso ad ogni blocco è veloce perché si sa immediatamente dove è.
- Si ha uno spreco di memoria quando una sequenza di blocchi contigui non è sufficiente ad ospitare un file
- Se il file aumenta di dimensioni si deve riallocare il file.

Ottimizzare lo spazio su disco

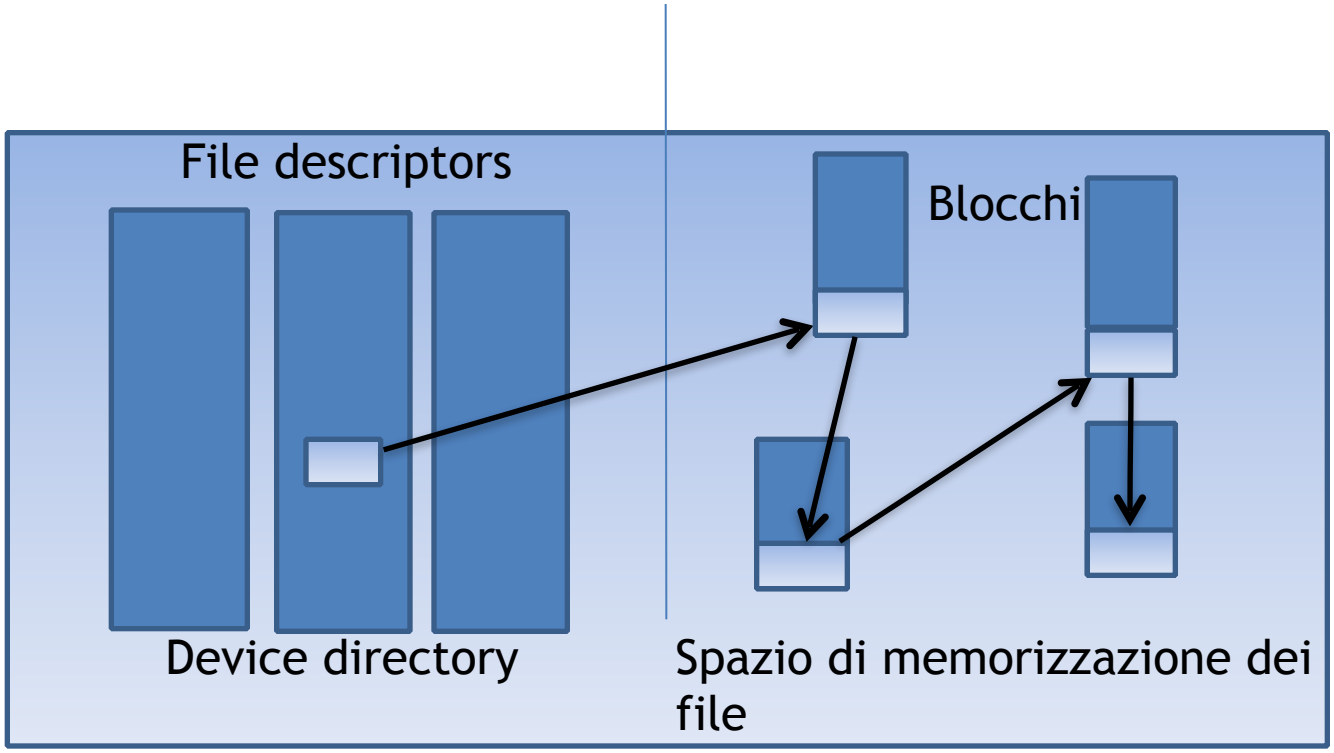
- Allocazione **sparsa**:
  - I blocchi che compongono il file possono trovarsi sparsi per il disco.
  - Come fare a sapere in quali blocchi si trova il file?
- Due metodi principali:
  - Allocazione (sparsa) **linkata**.
  - Allocazione (sparsa) **indexata**.
- Le tecniche più moderne adottano combinazioni di questi due metodi.

Allocazione linkata (2)

- Nel file descriptor viene memorizzato solo l'indirizzo del primo blocco del file sull'hard disk
- L'ultima parte del primo blocco contiene l'indirizzo del secondo blocco del file, e cosi' via per i successivi blocchi
- Si ha un piccolo spreco in ogni blocco per via del puntatore al blocco successivo e l'accesso agli ultimi blocchi del file e' lento.
- Se il file aumenta di dimensioni basta cercare un blocco libero sull'hard disk

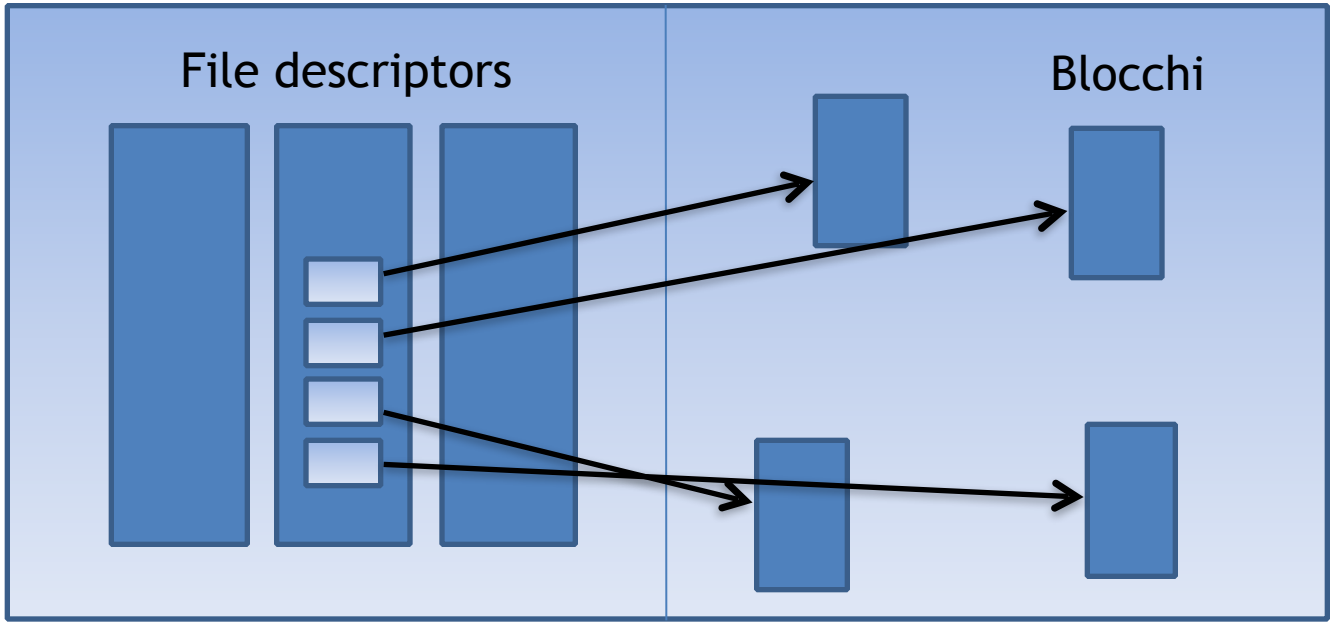
Ottimizzare lo spazio su disco

- Allocazione (sparsa) **linkata**: il file descriptor contiene l'indirizzo del primo blocco. In fondo a ciascun blocco che compone il file c'è l'indirizzo del blocco successivo.
  - L'accesso ai file diventa sequenziale (ossia per accedere al quarto blocco di un file devo prima leggermi i tre precedenti).



Ottimizzare lo spazio su disco

- Allocazione (sparsa) **indexata**: il file descriptor contiene l'elenco di tutti i blocchi.
  - Ad es. per un file di 3 blocchi: (30,10), (27,15), (42,13).



Allocazione indexata (2)

- Nel file descriptor viene riportato l'elenco di tutti i blocchi in cui e' memorizzato il file
- Anche l'accesso all'ultimo blocco del file e' veloce, perche' non si devono attraversare gli altri per trovarlo.

Cosa fa il sistema operativo?

1. Permette di avere in esecuzione più programmi contemporaneamente:
   - Gestione del processore → come "distribuire" il suo tempo fra programmi diversi?
   - Gestione della memoria principale → come "distribuire" il suo spazio fra programmi diversi?
2. Gestisce la memoria secondaria.
3. Permette agli utenti di interagire con il computer:
   - Gestione del sistema operativo stesso:
     - Installare (e disinstallare) programmi.
     - Configurare il computer e il sistema operativo.
   - Attività vera e propria:
     - Mandare in esecuzione un programma con cui creare nuovi documenti o elaborare documenti esistenti.
     - Eliminare e rinominare documenti creati e riorganizzarne la disposizione.
     - Utilizzare le periferiche.

Cosa può fare l'utente grazie al S.O.?

- Gestione del sistema operativo stesso:
  - Installazione di nuovi programmi (e disinstallazione).
  - Configurazione del sistema.
- Attività vera e propria:
  - Mandare in esecuzione un programma (+ tramite il programma, creare nuovi documenti o elaborare documenti esistenti).
  - Manipolare il file system:
    - Eliminare documenti creati.
    - Riorganizzare la disposizione dei documenti.
    - Rinominare documenti esistenti.
  - Accesso alle periferiche.

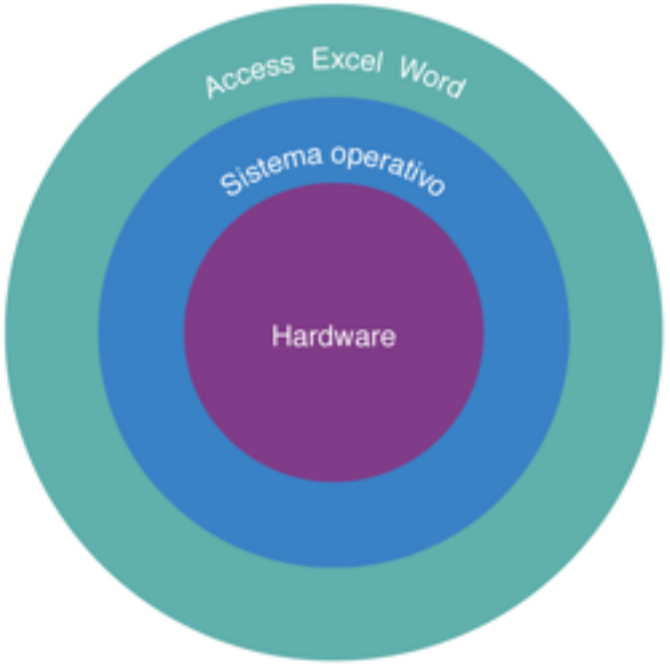
Ricapitolando: il Software

- **Sistema operativo** (software di base):
  - Viene fornito a corredo dell'hardware.
  - Programmi speciali per eseguire operazioni di base che:
    - Determinano in generale il comportamento del computer.
    - Permettono facilità d'uso da parte di un utente che eventualmente non ne conosca la struttura fisica.
  - Consente l'esecuzione del software applicativo.
- **Software applicativo**:
  - Ciascun applicativo permette di elaborare dati di natura diversa.
    - Editare testi, creare fogli elettronici, elaborare immagini.
  - Richiede la presenza del sistema operativo.

Programmi applicativi e sistema operativo

Sistema Operativo:

- **Indispensabile** (senza il sistema operativo il computer non funziona).
- Dal sistema operativo dipendono le funzioni comuni a tutti i programmi.
- Esempi di sistemi operativi su PC: **Windows, Linux, Unix, OS X**.

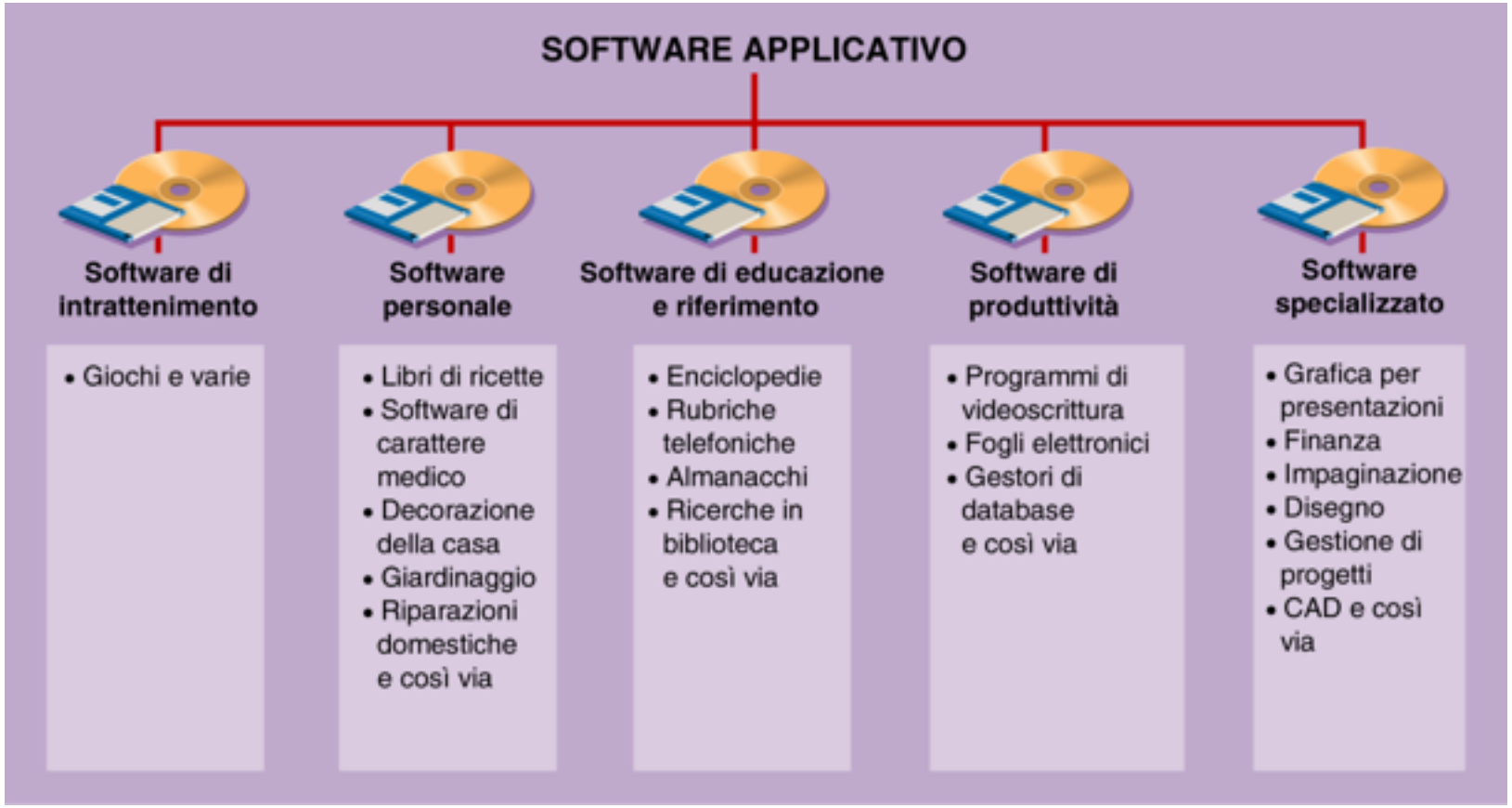


Programmi Applicativi:

- **Opzionali.**
- Necessari per funzioni specifiche (come scrivere, archiviare dati …).
- Compatibili con il software di base.
- Esempi: **Word**, **Excel**, **Access (Microsoft Office); Write, Calc, Impress (Open Office); Internet Explorer, Mozilla Firefox, … .**

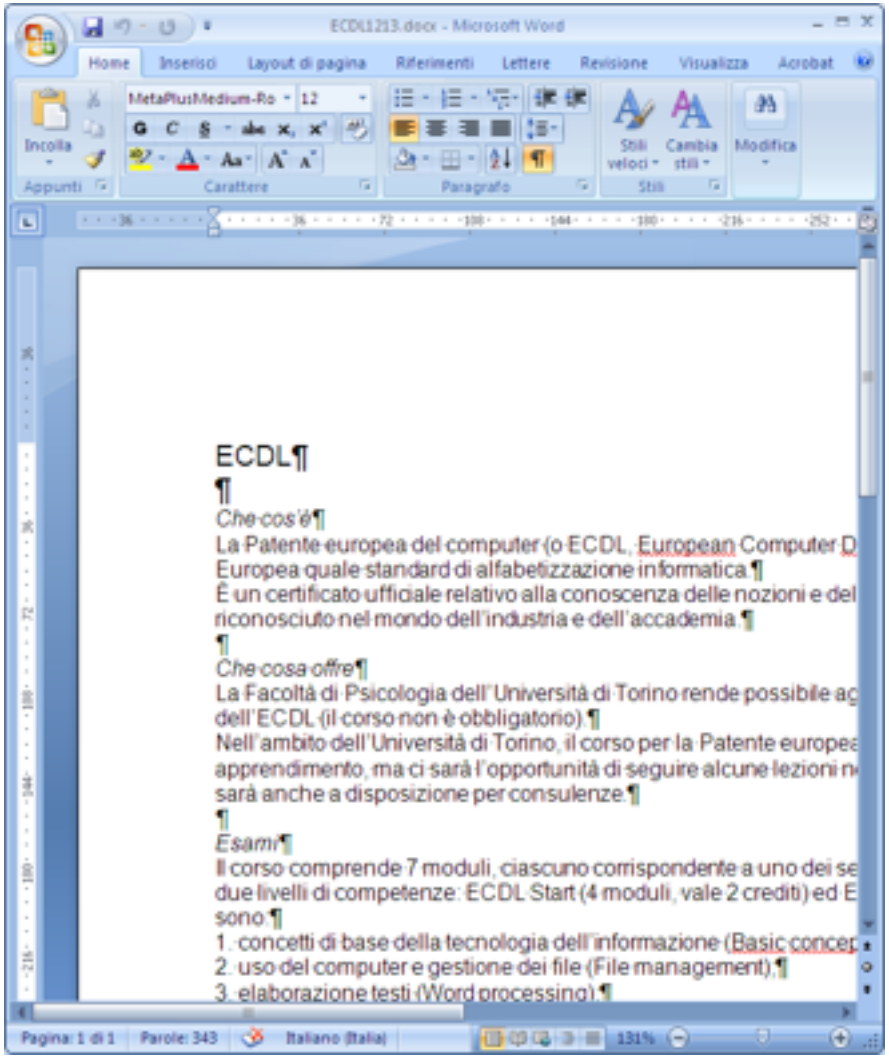
Software applicativo

- Classificazione in base all'uso.



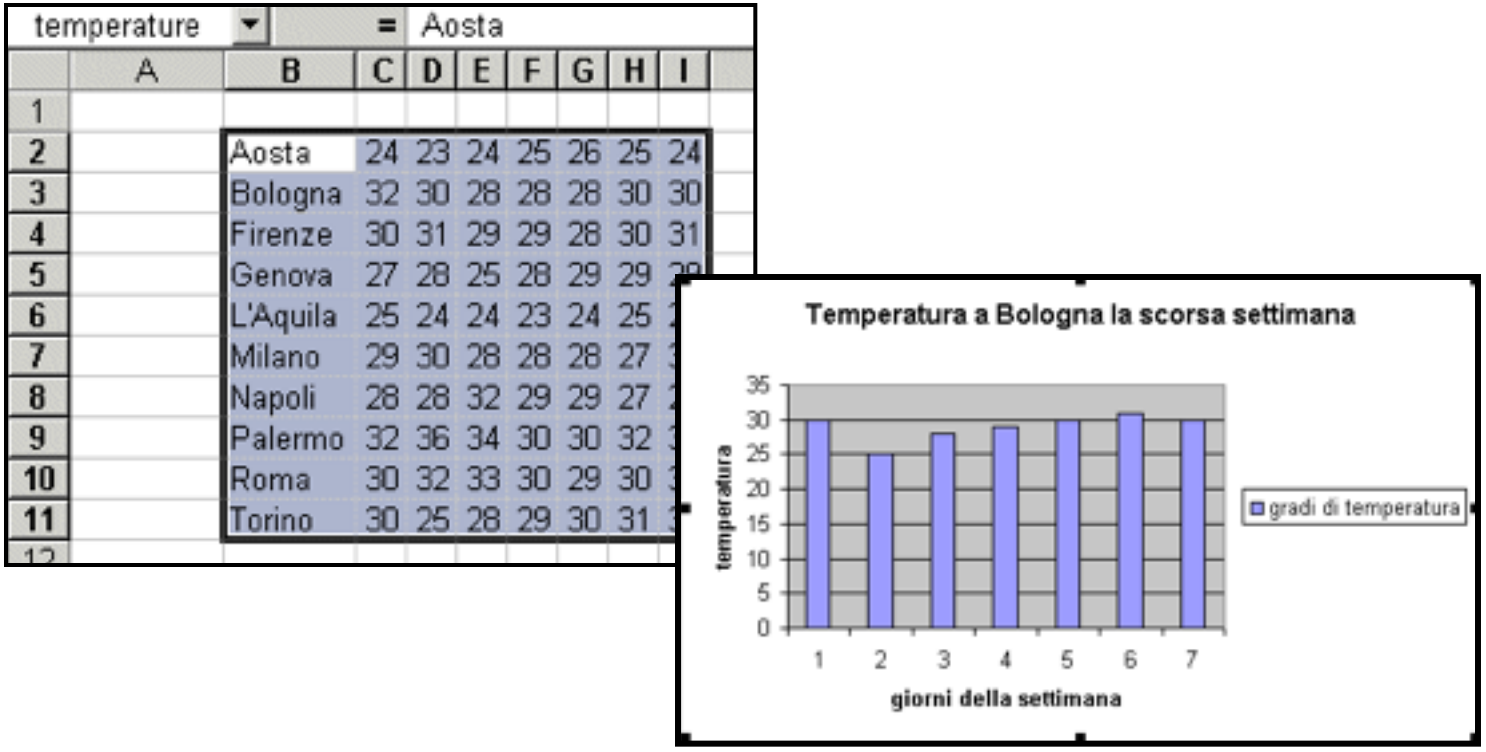
Software di produttività

- Videoscrittura



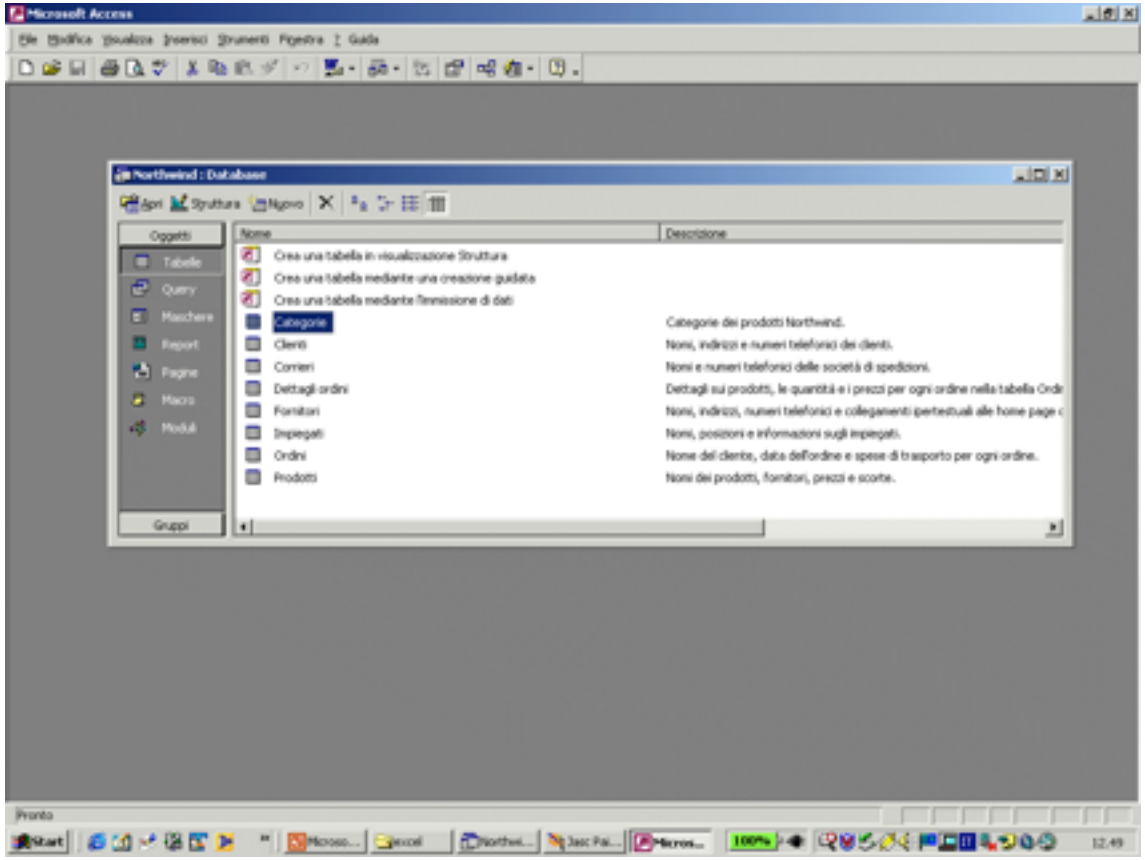
Software di produttività

- Fogli elettronici



Software di produttività

- Database



Copyright vs copyleft

- Classificazione del software in generale e quindi anche degli applicativi:
  1. Copyright vs. senza copyright:
     - **Copyright** → è coperto da diritti d'autore.
       - Una volta venutone in possesso non posso copiarlo, modificarlo, rivenderlo …
         - **Commerciale** → occorre acquistarlo.
         - **Shareware** → gratuito solo per un periodo di prova.
         - **Freeware** → disponibile gratuitamente.
     - **Senza copyright** → software di pubblico dominio non coperto da diritti di autore.
  2. Software proprietario vs. software "open source"
     - **Software proprietario** → il codice sorgente (istruzioni) è tenuto segreto.
     - **Software "open source"** → il codice sorgente è liberamente disponibile.
       - Posso modificarlo come voglio, migliorarlo, adattarlo alle mie esigenze ...

Il manifesto del software libero

- L'espressione "software libero" si riferisce alla libertà dell'utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software.
- Più precisamente, esso si riferisce a quattro tipi di libertà per gli utenti del software:
  - Libertà di eseguire il programma, per qualsiasi scopo .
  - Libertà di studiare come funziona il programma e adattarlo alle proprie necessità. **L'accesso al codice sorgente (open source)** ne è un prerequisito.
  - Libertà di ridistribuire copie in modo da aiutare il prossimo.
  - Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio. **L'accesso al codice sorgente ne è un prerequisito**.

Richard Stallman

Da Wikipedia, l'enciclopedia libera.

Richard Matthew Stallman (New York, 16 marzo 1953) è un programmatore, informatico e attivista statunitense.

È uno dei principali esponenti del movimento del software libero. Nel settembre del 1983 diede avvio al progetto GNU con l'intento di creare un sistema operativo simile a Unix ma composto interamente da software libero[1]; da ciò prese vita il movimento del software libero. Nell'ottobre del 1985 fondò la Free Software Foundation (FSF)[2]. Fu il pioniere del concetto di copyleft ed è il principale autore di molte licenze copyleft compresa la GNU General Public License (GPL), la licenza per software libero attualmente più diffusa.

Richard Stallman nel 2007

Software libero

- Dalla lettura del manifesto emergono tre concetti fondamentali:
  - Accesso al codice sorgente → la traduzione letterale di software “open source” è proprio software a “codice sorgente aperto”.
  - Libertà di copia e ridistribuzione.
  - Libertà di modifica per adattamento alle proprie esigenze o per migliorare il programma in sé.
    - La disponibilità del codice sorgente consente sia la sua libera circolazione, sia processi di modifica, produzione, ridistribuzione, evoluzione e riuso.

Software libero

- Tipici fraintendimenti rispetto al software libero.
  - Il software libero è **spesso gratuito** ma non è detto che lo sia.
    - "Support": esistono programmatori che vengono retribuiti per aggiornare, modificare, adattare programmi.
  - Il software gratuito non è **per forza libero**.
    - Molte aziende divulgano software proprietario gratuitamente per vendere altri prodotti, per attirare nuovi clienti o per porre fuori gioco un concorrente.
    - Si ha gratuitamente l'utilizzo del software, ma non si può modificarlo, ridistribuirlo, etc.

Un esempio: Linux

- http://www.linux.org/
- Linux è una particolare versione di Unix disponibile gratuitamente;
- Linux è un sistema operativo OpenSource: tutti vi possono accedere, tutti possono copiare programmi per modificarli o per eliminare eventuali malfunzionamenti o vulnerabilità;
- Linux è un software sviluppato in comunità: è il risultato di uno sforzo collettivo di migliaia di persone, non è prodotto da alcuna società commerciale e non si avvale di alcun reparto marketing e sviluppo che imponga delle "regole" ai propri utenti;

Linux

Ubuntu

Accesso non effettuato discussioni contributi Registrati Entra

Voce | Discussione | Leggi | Modifica | Modifica wikitesto | Cronologia | Ricerca

Ubuntu

Da Wikipedia, l'enciclopedia libera.

Disambiguazione – Se stai cercando altri significati, vedi ***Ubuntu (disambigua)****.*

(EN)	(IT)
« Ubuntu: Linux for Human Beings »	« Ubuntu: Linux per esseri umani »

(Slogan ufficiale della distribuzione[4])

Ubuntu ([uːˈbʊntuː][5], [ùbúntú] in zulu) è una distribuzione GNU/Linux basata su Debian,[6] nata nel 2004 e focalizzata sulla facilità di utilizzo.[5] Distribuita gratuitamente come software libero con licenza GNU GPL,[7] è orientata all'utilizzo desktop, ma presenta delle varianti per server, tablet, smartphone e dispositivi IoT, ponendo grande attenzione al supporto hardware. È prevista una nuova versione ogni sei mesi.

Il nome *Ubuntu*, termine nguni bantu traducibile come "umanità verso gli altri", fa riferimento ad una filosofia di origine sudafricana che teorizza un legame universale di scambio che unisce l'intera umanità (letteralmente, *"dell'Essere Umano"*).

Lo sviluppo di Ubuntu è sostenuto da Canonical Ltd., un'azienda britannica fondata dall'imprenditore sudafricano Mark Shuttleworth,[8] ed è finanziato dall'offerta di supporto tecnico a pagamento[9] e da accordi commerciali con gli OEM.[10]

Indice [nascondi]

1 Storia
2 Caratteristiche principali
3 Requisiti di sistema
4 Installazione
5 I componenti, o repository, ufficiali
6 Versioni
7 Edizioni

Ubuntu

Screenshot di Ubuntu Desktop 15.04 *Vivid Vervet*

Sviluppatore	Canonical Ltd Ubuntu Foundation
Famiglia SO	GNU/Linux (Linux 4.2.3)
Modello del sorgente	Software libero
Release iniziale	4.10 (Warty Warthog)[1] (20 ottobre 2004)
Release corrente	15.10 (Wily Werewolf)[2] (22 ottobre 2015)
Tipo di kernel	Kernel Linux monolitico

Women in CS

Ada Lovelace

Grace Hopper

Margaret Hamilton

many more…