Università degli Studi di Torino
Scuola di Amministrazione Aziendale

Corso di laurea in
Management dell'informazione e della comunicazione aziendale

Corso di
Informatica generale (parte teorica)

Diego Magro

ARGOMENTI DI QUESTO GRUPPO DI LUCIDI

- **SOFTWARE**
  - generalità
  - macchina virtuale
  - algoritmi e programmi
  - programmi traduttori: compilatori ed interpreti

- **SISTEMI OPERATIVI**

SOFTWARE

SOFTWARE (1)

- **Un moderno elaboratore non si esaurisce nelle sue parti *hardware*: esso ha una duplice natura e, oltre a quella hardware, esso possiede quella *software,* costituita da *programmi***

- **Come detto in precedenza, caratteristica essenziale di un elaboratore è quella di essere programmabile**

- **Si ricordi la distinzione precedentemente introdotta fra software applicativo e software di base**

SOFTWARE (2)

- **L'interazione diretta con la macchina fisica sarebbe problematica sia per**
  - l'*utente finale* (colui/colei che usa la macchina per risolvere dei problemi e, quindi, è principalmente interessato/a alle applicazioni)
  - ➔ fra l'altro, le differenze hardware tra una macchina e l'altra potrebbero comportare differenze nelle modalità di interazione

sia per

- il *programmatore* (colui/colei che scrive i programmi), in particolare per il programmatore di software applicativo: una programmazione diretta della macchina fisica sarebbe molto dispendiosa per il programmatore, costretto a confrontarsi con lunghe sequenze di istruzioni in linguaggio macchina e a tener conto delle peculiarità hardware delle componenti del calcolatore
- ➔ la stretta dipendenza dei programmi applicativi dalle caratteristiche hardware del calcolatore comporterebbe la necessità di scrivere molte versioni di programmi equivalenti (cioè che realizzano le stesse funzionalità), adatte ciascuna ad una specifica tipologia di hardware

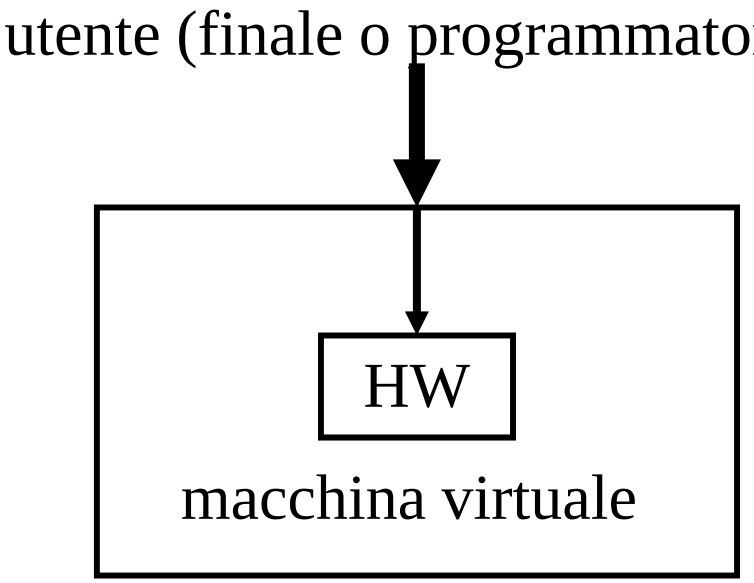
SOFTWARE (3)

- ➔ l'idea è quella di fornire all'utente finale e al programmatore una visione astratta della macchina, più "amichevole" di quella fisica

- questa macchina astratta è detta macchina virtuale, in quanto fisicamente inesistente

- La macchina virtuale è realizzata per mezzo di un livello software, cioè un insieme di programmi, "sopra" la macchina hardware. Questi programmi costituiscono il cosiddetto software di base.

SOFTWARE (4)



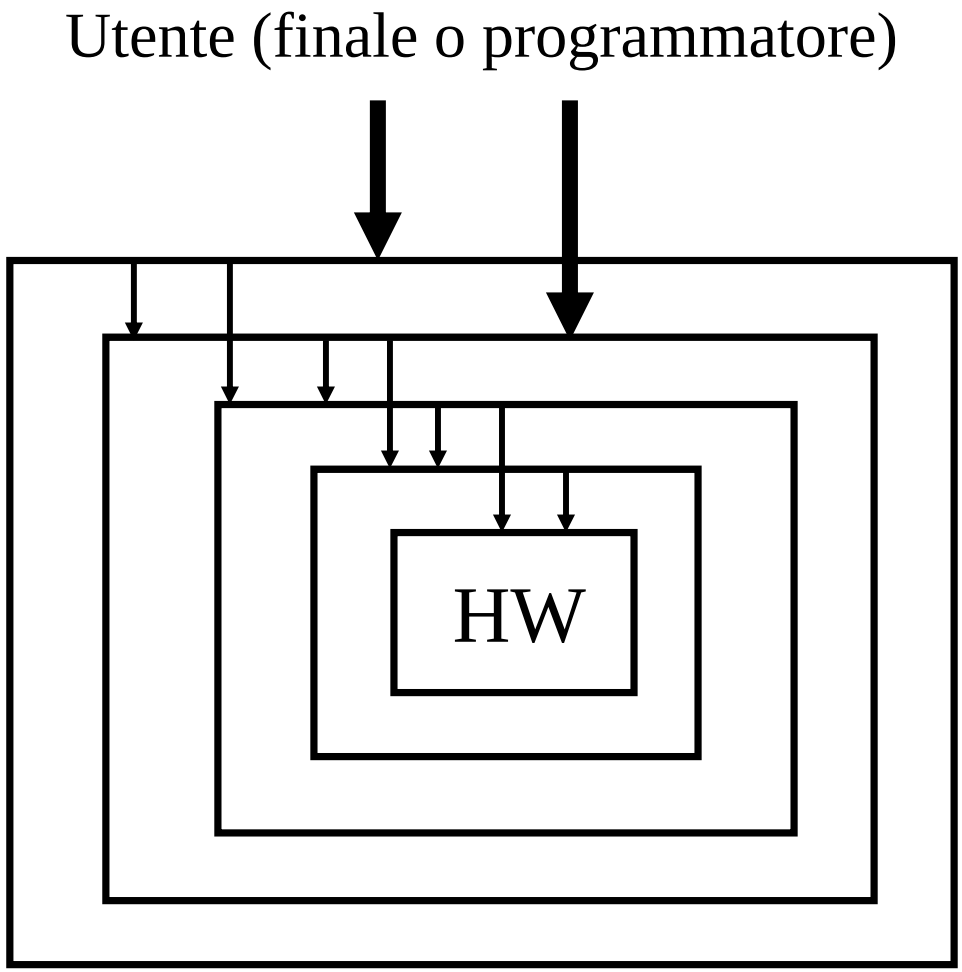
- **La macchina virtuale mette a disposizione dell'utente un nuovo linguaggio (ed un "ambiente"), più adatto ad un essere umano**
- **l'utente interagisce con il calcolatore per mezzo di questo linguaggio**

- **i comandi espressi dall'utente in questo linguaggio sono tradotti dal software di base in comandi eseguibili dalla macchina**
- **l'utente può specificare altri comandi al di fuori di questo linguaggio, ma, in questo caso, deve fornire anche il programma che realizza la nuova funzione**

SOFTWARE (5)

- **In realtà, il moderno software di base è strutturato in una gerarchia di macchine virtuali (struttura a cipolla)**
- **Ogni livello della gerarchia ha a disposizione le funzioni realizzate dai livelli "sottostanti" e fornisce un insieme di funzioni utilizzabili da eventuali livelli "superiori"**
- **Le funzionalità sono sempre più astratte man mano che ci si allontana dalla macchina fisica**
- **In questo contesto, si parla di "basso livello" per riferirsi ai livelli più vicini alla macchina hardware e di "alto livello" per riferirsi ai livelli più lontani dalla macchina hardware e più vicini all'utente ("basso" non ha connotazione negativa e "alto" non ha connotazione positiva...)**

SOFTWARE (6)



Alcuni dei livelli sono forniti direttamente dal software di base, altri possono essere creati dall'utente: programmare un calcolatore significa costruire un nuovo comando virtuale la cui traduzione è il programma stesso.
N.B. Le macchine virtuali non permettono di realizzare funzioni non eseguibili dalla macchina fisica...alla base di tutto, resta la materia!

SOFTWARE (7)

- **Che vantaggi offre la gerarchia di macchine virtuali, rispetto ad un unico strato di software?**

1. **la ridotta distanza fra un livello e l'altro facilita la traduzione di un comando da un livello a quello successivo**

2. **non è necessario fornire traduzioni completamente diverse per macchine hardware diverse: le differenze hardware vengono appianate nei livelli bassi e i livelli superiori non cambiano**

SOFTWARE (8)

- **Grossolanamente, possiamo suddividere le principali funzionalità del <u>software di base</u> in due categorie:**
  - strumenti per l'uso di linguaggi di programmazione ad alto livello:
    - compilatori ed interpreti che effettuano la traduzione di programmi scritti in linguaggi ad alto livello in sequenze di istruzioni in linguaggio macchina
    - strumenti e ambienti per lo sviluppo di programmi
  - funzionalità del sistema operativo

ALGORITMI E PROGRAMMI (1)

- **Algoritmo: impossibile darne una precisa definizione formale. Informalmente, è la specifica di un procedimento per la risoluzione di un qualche tipo di problema.**
- **Un algoritmo, solitamente, specifica la sequenza di passi che deve essere eseguita per risolvere una certa tipologia di problemi, con gli eventuali punti di scelta e le eventuali ripetizioni di sequenze di istruzioni**
- **Etimologicamente, il termine "algoritmo" deriva dal nome del matematico persiano del IX secolo (che si occupò di metodi di calcolo) Muḥammad ibn Mūsā al-Khwārizmī, il cui nome fu latinizzato in *Algoritmi***
- **Non tutti i problemi ammettono delle soluzioni algoritmiche e, in particolare, esistono problemi definibili in maniera precisa e formale, ma per i quali non esiste alcun algoritmo che possa specificarne il procedimento di risoluzione**

ALGORITMI E PROGRAMMI (2)

- **Programma: è la specifica di un algoritmo in un linguaggio "comprensibile" ad un calcolatore**
- **Il processore è in grado di eseguire direttamente solo istruzioni macchina, pertanto, per poter essere eseguito direttamente dall'hardware, un algoritmo deve essere espresso in linguaggio macchina**
- **Tuttavia, il linguaggio macchina è poco adatto agli utenti umani ➔ esistono linguaggi di programmazione "di alto livello" (cioè più vicini all'utente umano) in cui il programmatore può specificare gli algoritmi**
- **I linguaggi di programmazione di alto livello, sono "comprensibili" al calcolatore (nel modo che vedremo in seguito) inteso come macchina virtuale, anche se non alla macchina hardware**

ALGORITMI E PROGRAMMI (3)

Il frammento di programma in linguaggio macchina (anche qui riportato in un formato mnemonico) introdotto in precedenza per illustrare il ciclo di fetch-decode-execute

...

```
LOAD R0 5000
LOAD R1 5001
ADD R0 R1
STORE R0 5002
JUMPNZ R0 2009
LOAD R0 5003
LOAD R1 5004
ADD R0 R1
STORE R0 5002
HALT
```

Un possibile equivalente (nel senso che "fa le stesse cose") frammento di programma in un linguaggio di alto livello

...

```
c = a + b;

if (c == 0) then c = d + e
```

PROGRAMMI TRADUTTORI: COMPILATORI ED INTERPRETI (1)

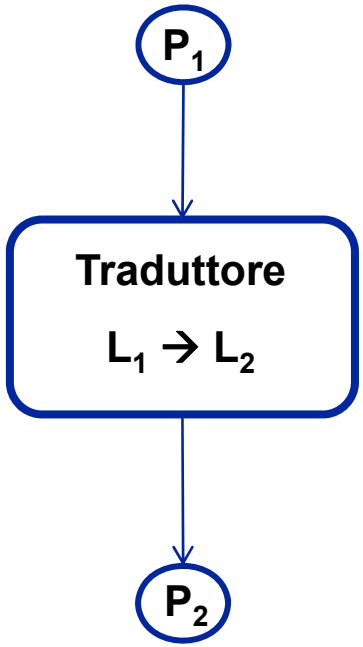
- **Affinché un algoritmo specificato in un programma scritto in un linguaggio di alto livello possa essere eseguito dalla macchina hardware, le istruzioni del programma ad alto livello devono essere tradotte in istruzioni macchina**

- **Questa operazione di traduzione non è eseguita né dall'utente finale, né dal programmatore, ma da degli appositi software (di base), chiamati traduttori**

- **In questo contesto, il programma scritto in linguaggio ad alto livello è detto codice sorgente**

- **In questo modo, anche se la macchina hardware non è in grado di eseguire direttamente programmi espressi in linguaggi di alto livello, la macchina virtuale lo è (per mezzo dei programmi traduttori): in questo senso possiamo dire che i linguaggi di alto livello sono comprensibili al calcolatore (inteso come macchina virtuale)**

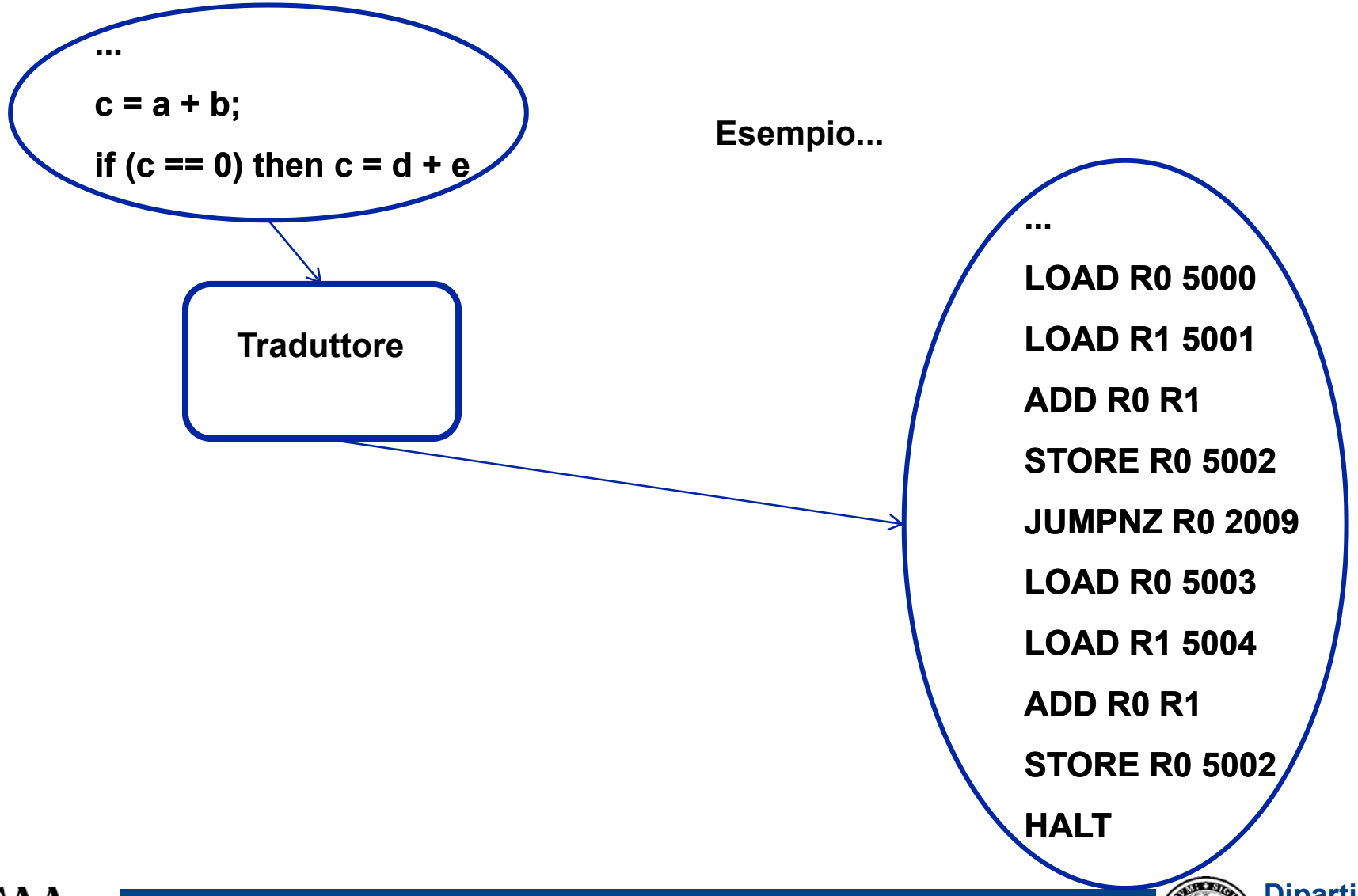
PROGRAMMI TRADUTTORI: COMPILATORI ED INTERPRETI (2)



Un traduttore da linguaggio L_1 a linguaggio L_2, per ogni programma P_1 espresso in L_1 ricevuto in input, produce in output un programma P_2,

espresso in L_2 ed equivalente a P_1 (nel senso che specifica lo stesso algoritmo specificato da P_1)

PROGRAMMI TRADUTTORI: COMPILATORI ED INTERPRETI (3)



PROGRAMMI TRADUTTORI: COMPILATORI ED INTERPRETI (4)

- **Vi sono due categorie di programmi traduttori: i compilatori e gli interpreti**

- **Compilatori: ricevono in input un programma e lo traducono interamente, producendo un file contenente il programma tradotto. In un secondo momento, il programma tradotto (a seconda dei casi) può essere passato in input ad un altro traduttore o (dopo eventuali altri passaggi qui omessi per semplicità) direttamente eseguito dalla macchina hardware (se è espresso in linguaggio macchina)**
- **I compilatori sono l'analogo dei traduttori (umani) di libri, i quali "ricevono in input" un libro e "producono in output" un altro libro (che dice le stesse cose del primo) come risultato della loro traduzione. In un secondo momento, il libro tradotto può essere nuovamente tradotto in un'altra lingua o, più frequentemente, letto da qualcuno**
- **Si noti che la fase di compilazione e quella di esecuzione sono separate ed attuate in sequenza: prima il calcolatore esegue il compilatore che traduce per intero il programma e poi, eventualmente, il calcolatore esegue il programma tradotto**

PROGRAMMI TRADUTTORI: COMPILATORI ED INTERPRETI (5)

- **Interpreti: ricevono in input un programma e ne traducono le istruzioni e le mandano in esecuzione man mano che le incontrano. Non producono alcun file contenente il programma tradotto.**

- **Gli interpreti sono l'analogo dei traduttori simultanei (umani) i quali "ricevono in input" un discorso nel momento in cui questo viene pronunciato e "dicono in output", le stesse cose che sentono (in un linguaggio diverso da da quello che sentono), man mano che le sentono**

- **Si noti che la fase di interpretazione e quella di esecuzione sono interconnesse: il calcolatore esegue l'interprete che traduce e manda in esecuzione le istruzioni del programma sorgente**

SISTEMA OPERATIVO

Generalità (1)

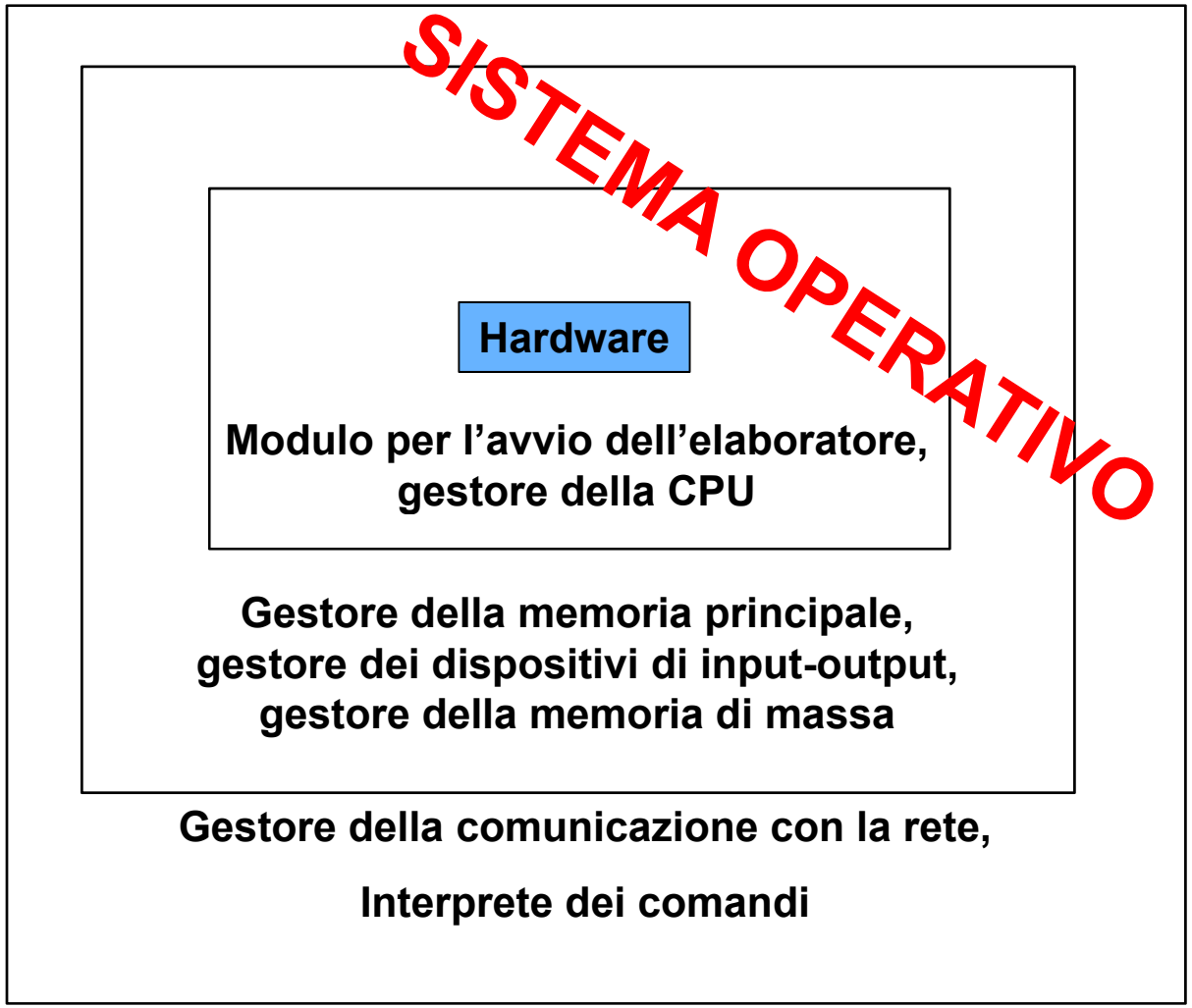
- **Il Sistema Operativo (SO) è il principale rappresentante del software di base**
- **E' costituito da un insieme di programmi che hanno una duplice funzione:**
  1. realizzare un'interfaccia fra l'utente finale e il programmatore di programmi applicativi da un lato e la macchina hardware dall'altro (possiamo anche dire che esso costituisce un'interfaccia fra i programmi applicativi e la macchina hardware). Come detto, esso contribuisce a realizzare la macchina astratta (macchina virtuale) con cui sia gli utenti finali sia i programmi applicativi interagiscono
  2. assicurare un uso corretto ed ottimale dell'insieme delle risorse hardware del calcolatore (CPU, memoria principale, memoria di massa e dispositivi di input-output)

Generalità (2)

- **E' costituito da vari moduli software, generalmente strutturati "a cipolla", che realizzano ciascuno un preciso insieme di funzionalità**
- **I moduli più interni sono quelli più vicini all'hardware, mentre quelli più esterni sono quelli più vicini all'utente**
- **Un modulo è considerato più esterno di un altro se il primo usa le funzionalità e i servizi messi a disposizione dal secondo per realizzare le proprie funzionalità e i propri servizi (questi ultimi messi a disposizione di un eventuale livello ancora più esterno)**

Generalità (3)

...la cipolla...



Principali funzionalità

1. **Avvio dell'elaboratore**
2. **Gestione del processore (e dei processi)**
3. **Gestione della memoria principale (e realizzazione della memoria virtuale)**
4. **Gestione dei dispositivi di input-output**
5. **Gestione della memoria di massa**
6. **Gestione della comunicazione in rete**
7. **Realizzazione dell'interprete dei comandi**

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Avvio dell'elaboratore (1)

- **Nell'interazione con i moderni calcolatori, l'utente finale comunica con il calcolatore attraverso l'interprete dei comandi realizzato dal sistema operativo ➔ l'interazione con il calcolatore presuppone che il sistema operativo sia in esecuzione**

- **Analogamente, i programmi applicativi usano i servizi messi loro a disposizione dal sistema operativo ➔ l'esecuzione di programmi applicativi presuppone che il sistema operativo sia in esecuzione**

- **Il processore è in grado di eseguire istruzioni solo se queste si trovano in memoria principale, ma, all'avvio del calcolatore, la memoria RAM non contiene informazioni, quindi, in particolare, non contiene istruzioni del sistema operativo**

Avvio dell'elaboratore (2)

- **Come è possibile che il calcolatore, all'avvio, inizi ad eseguire programmi del sistema operativo, predisponendo la macchina per l'interazione con l'utente e l'esecuzione dei programmi applicativi?**

- **Esiste nel calcolatore una memoria, detta ROM (Read Only Memory) che contiene le prime istruzioni che il calcolatore esegue al proprio avvio (➔ le prime istruzioni eseguite dalla CPU all'avvio non si trovano nella memoria RAM)**

- **La ROM è una memoria a sola lettura, che contiene le istruzioni cablate nei propri circuiti. Le informazioni nella ROM sono persistenti e non possono essere sovrascritte (a parte per certi tipi di ROM e con apposite procedure), quindi non vanno perdute nel momento in cui il calcolatore non è più alimentato**

Avvio dell'elaboratore (3)

- **Le istruzioni contenute in ROM ed eseguite all'avvio costituiscono il cosiddetto programma di bootstrap**
- **Il programma di bootstrap memorizzato in ROM è parte del firmware del calcolatore**
- **Il programma di bootstrap può essere considerato una parte del sistema operativo sempre residente nei circuiti del calcolatore**
- **La parte restante del sistema operativo si trova memorizzata in un dispositivo di memoria di massa (solitamente un hard disk)[1]**

1 In alcuni semplici calcolatori (es. i PDA, Personal Digital Assistant), il sistema operativo è interamente cablato in ROM

Avvio dell'elaboratore (4)

- **Le operazioni compiute dal programma di bootstrap possono variare da un calcolatore ad un altro, ma, solitamente, il programma di bootstrap:**
  1. esegue alcune operazioni di diagnosi sull'hardware della macchina
  2. inizializza l'hardware predisponendolo al funzionamento
  3. copia da un dispositivo di memoria di massa (solitamente hard disk) in memoria RAM una porzione di sistema operativo
  4. trasferisce l'esecuzione al frammento di sistema operativo copiato in memoria RAM

- **L'esecuzione nel frammento di sistema operativo caricato dal programma di bootstrap produce il caricamento in memoria principale e l'esecuzione delle restanti parti del sistema operativo necessarie alla realizzazione completa della macchina virtuale**

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Gestione del processore (e dei processi) (1)

- **Nell'ambito dei sistemi operativi, riveste fondamentale importanza il concetto di *processo***
- ***Processo*: programma in esecuzione**
- **Programma e processo sono due concetti distinti: un programma è un'entità passiva e statica, un processo è un'entità attiva e dinamica**
- **Es. una ricetta di cucina è un'entità passiva e statica (non fa nulla e non varia nel tempo), mentre la sua esecuzione è un entità dinamica, un processo – appunto – che è eseguita da qualcuno, che richiede delle risorse, che ha una durata temporale e che produce del cibo**

Gestione del processore (e dei processi) (2)

- **Il sistema operativo gestisce l'allocazione della risorsa hardware CPU fra i vari programmi che sono eseguiti (cioè fra i vari processi)**
- **In generale, possiamo dire che fra i principali compiti del sistema operativo vi è la gestione dei processi**
- **L'esecuzione di un programma che è parte del sistema operativo produce un processo si sistema**
- **L'esecuzione di un programma applicativo (programma utente) produce un processo utente**
- **Il sistema operativo gestisce l'uso delle risorse da parte dei processi utente**

Gestione del processore (e dei processi) (3)
[sistemi mono /multi-utente, sistemi mono/multi-programmati]

- **Per capire le problematiche relative alla gestione dei processi nei moderni calcolatori, è utile introdurre i seguenti concetti:**

  - **Sistemi mono-utente**: sono calcolatori che possono essere usati da una sola persona per volta (es i primi calcolatori, i moderni PC, ...)

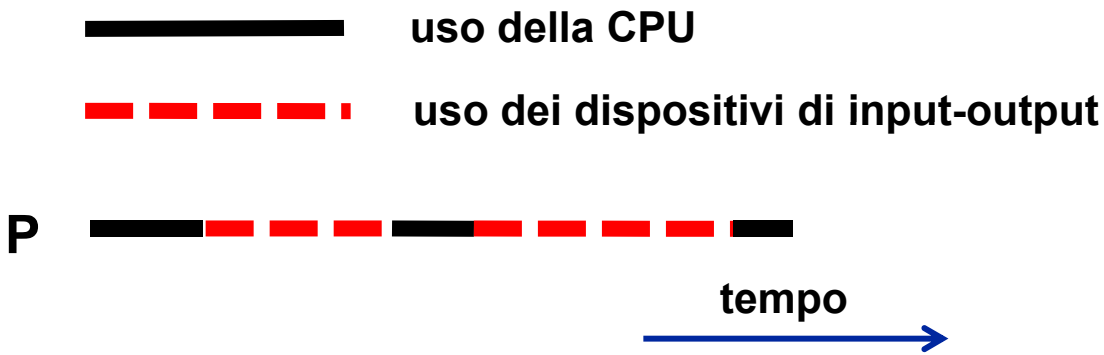
  - **Sistemi multi-utente**: sono calcolatori che possono essere usati da più persone contemporaneamente (es. server aziendali,...)

  - **Sistemi mono-programmati**: sono sistemi in grado di eseguire un solo programma utente alla volta (es. i primi calcolatori)

  - **Sistemi multi-programmati**: sono sistemi in grado di eseguire più programmi utente alla volta (es. i moderni PC, i server aziendali, ...)

Gestione del processore (e dei processi) (4)

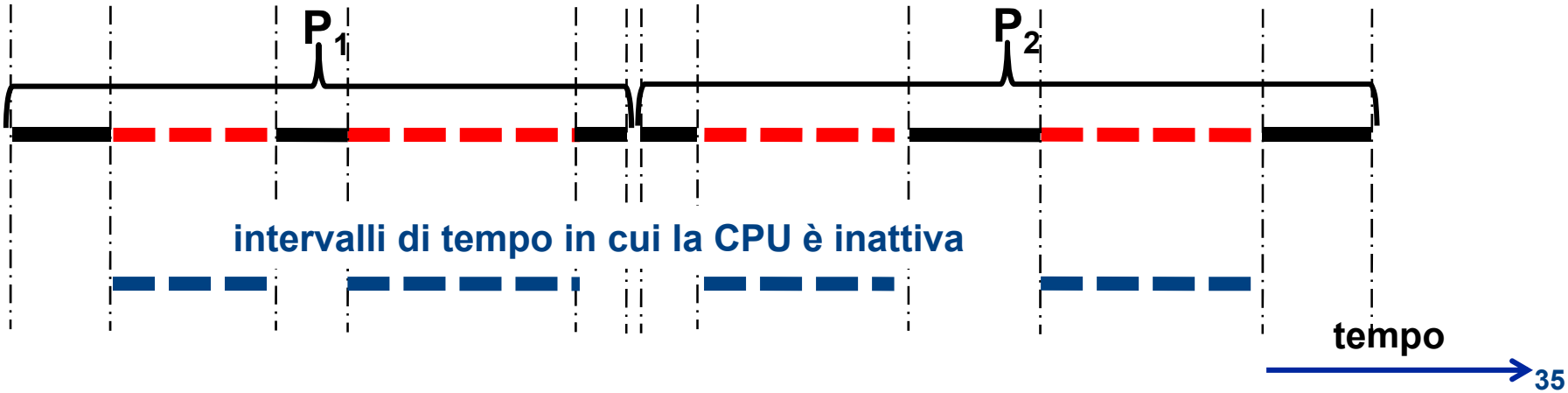
- I moderni calcolatori sono sistemi multi-programmati
- La multi-programmazione consente un uso migliore della CPU (purché essa sia gestita secondo criteri di ottimalità, quali sono quelli che informano l'attività dei sistemi operativi)
- Tutti i processi, da un certo punto di vista, si comportano nello stesso modo: durante la loro esistenza, alternano momenti di uso della CPU a momenti di uso dei dispositivi di input-output



Gestione del processore (e dei processi) (5)

- **Un sistema mono-programmato esegue i processi utente in sequenza: l'esecuzione di un processo può iniziare solo quando termina l'esecuzione del processo eventualmente correntemente in esecuzione**
- **In questo modo, la CPU resta lunghi periodi di tempo inattiva ed è, pertanto, sottoutilizzata**
- **...si ricordi che la CPU è in grado di lavorare molto più rapidamente rispetto ai dispositivi di input-output, quindi lasciarla inattiva in attesa che un operazione di input o di output sia terminata è uno speco enorme!**

- **Es.: esecuzione in sequenza di due processi utente:**



Gestione del processore (e dei processi) (6)

- **La CPU sarebbe sfruttata meglio se potesse eseguire istruzioni del processo P_2 nel momento in cui P_1 è in attesa del completamento di un'operazione di input o di output, e viceversa (questo consentirebbe una riduzione del tempo complessivo, necessario ad eseguire entrambi i processi):**

Gestione del processore (e dei processi) (7)

- **Nei sistemi multi-programmati, più processi utente possono essere contemporaneamente attivi**

- ***Viene realizzato un parallelismo virtuale tra i processi: più processi si alternano nell'uso della stessa CPU; anche se, in ogni istante, una singola CPU può eseguire istruzioni di al più un processo, l'alternanza dei processi dà l'illusione che una stessa CPU esegua contemporaneamente più processi in parallelo***

- **Questo consente sia un miglior sfruttamento della CPU rispetto alla mono-programmazione, sia di realizzare un ambiente interattivo, in cui uno o più utenti possono interagire con più processi contemporaneamente**

Gestione del processore (e dei processi) (8)

- **...es. per un utente di PC, è usuale usare contemporaneamente un browser, un client di posta, un elaboratore di testi, un foglio elettronico, ecc., avendo l'impressione che queste applicazioni siano tutte contemporaneamente in esecuzione e di poter interagire in tempo reale con ognuna di esse, anche se il calcolatore dispone di una sola CPU...**

- **Il prezzo da pagare è una maggior complessità del sistema, ma questa complessità è a carico del *sistema operativo*, che *gestisce l'alternanza dell'uso della CPU da parte dei processi***

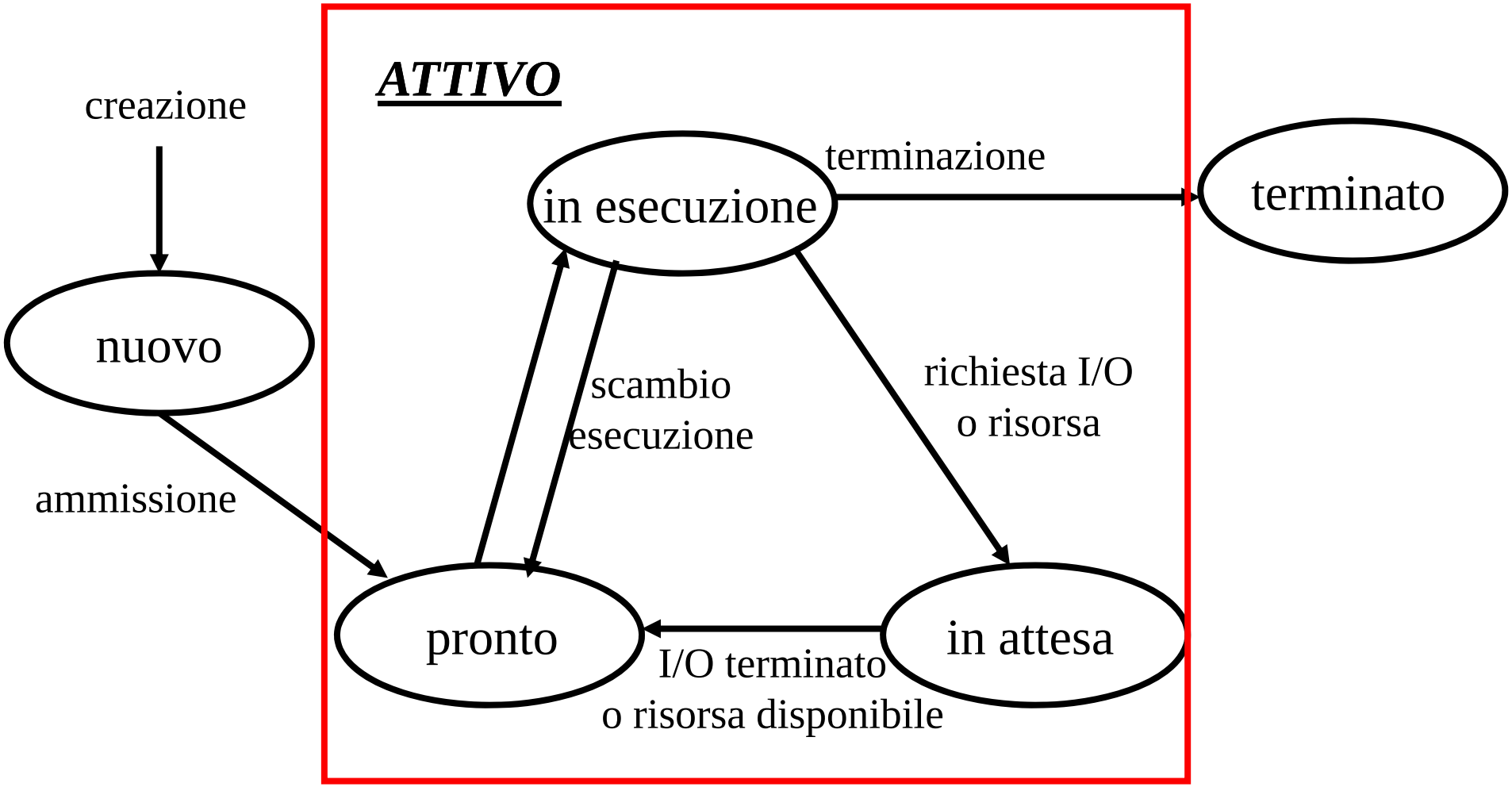
- **Nel seguito, considereremo, per semplicità, solo calcolatori con un singolo processore: i principi introdotti valgono comunque anche nel caso di sistemi multi-processore (anche se tali sistemi presentano specifiche caratteristiche che qui non considereremo)**

Gestione del processore (e dei processi) (9)
[Stati di un processo]

- **Ogni processo, nel corso del proprio ciclo di vita in un calcolatore, assume diversi stati e, in ogni momento, si trova sempre in uno dei seguenti stati[1]:**
  1. Nuovo (o in anticamera): quando viene creato
  2. Pronto: quando è nelle condizioni di poter usare la CPU (cioè dispone di tutte le risorse necessarie alla propria esecuzione, tranne la CPU), occupata da un altro processo
  3. In esecuzione: quanto sta usando la CPU (detto altrimenti: quando la CPU sta eseguendone delle istruzioni)
  4. In attesa: quando è in attesa del verificarsi di un evento esterno, ad esempio, la terminazione di un'operazione di input/output o la possibilità di utilizzare una qualche risorsa in uso da parte di altri processi
  5. Terminato: quando ha terminato la propria esecuzione

1 Elenchiamo qui i principali stati di un processo, riconosciuti in ogni sistema operativo. Alcuni sistemi operativi riconoscono per i processi anche altri stati (ad esempio, lo stato di "scaricato" per quei processi parzialmente eseguiti, ma che sono stati momentaneamente scaricati dalla memoria principale a quella secondaria)

Gestione del processore (e dei processi) (10)
[Stati di un processo e possibili transizioni da uno stato all'altro]



Gestione del processore (e dei processi) (11)
[Stati di un processo e possibili transizioni da uno stato all'altro]

- **I processi che si trovano nello stato *pronto*, *in esecuzione* o *in attesa*, sono detti <u>processi attivi</u>**

- **Si noti che:**
  - un processo *nuovo* non viene subito mandato *in esecuzione*, ma assume prima lo stato di *pronto*
  - solo un processo *pronto* può andare in *esecuzione*
  - quando termina l'*attesa*, un processo, prima di andare *in esecuzione*, deve tornare ad essere *pronto*
  - lo stato di *esecuzione* può essere abbandonato per tre diverse ragioni:
    1. terminazione: il processo termina la propria esecuzione e abbandona il sistema
    2. richiesta di un'operazione di input/output o di una risorsa correntemente non disponibile
    3. cambio di esecuzione: per realizzare in modo equo l'alternanza tra i vari processi, in certi casi il sistema operativo può interrompere l'esecuzione di un processo, rimetterlo nello stato di *pronto* e assegnare il processore ad un altro processo

Gestione del processore (e dei processi) (11)
[Code di processi]

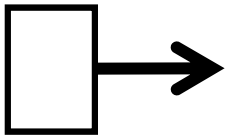
- **Allarghiamo la prospettiva dal singolo processo all'insieme dei processi presenti in un sistema in un dato istante**

- **Tale insieme può essere descritto in termini di code, le quali corrispondono ai possibili stati (il passaggio da una coda all'altra avviene secondo le modalità descritte nel precedente schema)**

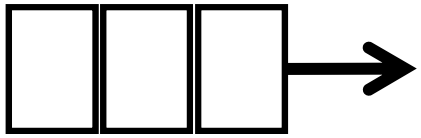
- **Si noti che, in ogni istante, la coda dei processi in esecuzione può contenere al più un elemento**

Gestione del processore (e dei processi) (12)
[Code di processi]

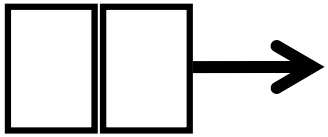
coda dei processi in esecuzione (al più un processo)



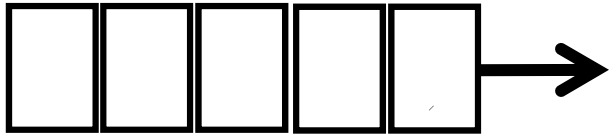
coda dei processi nuovi o in anticamera



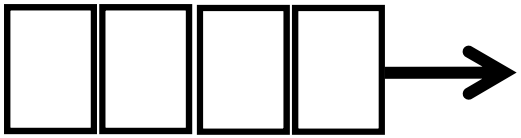
coda dei processi in attesa dell'evento 1



coda dei processi pronti



coda dei processi in attesa dell'evento N



Gestione del processore (e dei processi) (13)
[Schedulatori]

- **Periodicamente, il sistema operativo:**
  1. sceglie un processo nella coda dei processi nuovi e lo ammette fra i processi pronti, cioè lo accoda nella coda dei processi pronti (scheduling o schedulazione a lungo termine). Questo compito è eseguito da un modulo del sistema operativo, detto schedulatore a lungo termine
  2. sceglie un processo nella coda dei processi pronti e lo manda in esecuzione (scheduling o schedulazione a breve termine, detto anche scheduling o schedulazione della CPU). Questo compito è eseguito da un modulo del sistema operativo, detto schedulatore a breve termine
- **In alcuni sistemi operativi, lo schedulatore a lungo termine non è presente (e ogni processo è immediatamente accodato fra i pronti non appena viene creato)**
- **In alcuni sistemi operativi è presente anche uno schedulatore a medio termine, per i processi parzialmente eseguiti e *scaricati***

Gestione del processore (e dei processi) (14)
[Schedulatori e politiche di schedulazione]

- **Gli schedulatori attuano le proprie scelte in base a precisi criteri, detti politiche di schedulazione**
- **Le politiche di schedulazione perseguono cinque principali obiettivi:**
  1. Massimizzare il grado di utilizzazione del processore, ossia fare in modo che il processore rimanga attivo per il maggior tempo possibile
  2. Massimizzare il numero di processi che vengono completati nell'unità di tempo, cioè il *throughput* del sistema
  3. 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
  4. Minimizzare il tempo totale di attesa di ciascun processo nella coda dei processi pronti (...un processo pronto occupa risorse...)
  5. Nel caso di processi interattivi, minimizzare il tempo di risposta agli utenti

Gestione del processore (e dei processi) (15)
[Politiche di schedulazione]

- **Alcuni di questi obiettivi sono in contraddizione fra loro (es: se si aumenta il numero dei processi contemporaneamente attivi, verosimilmente si aumenta anche il grado di utilizzo del processore, ma, altrettanto verosimilmente, si aumenta anche il tempo di attesta dei processi nella coda dei pronti)**

- **➔ le politiche di schedulazione tendono ad ottenere ragionevoli compromessi**

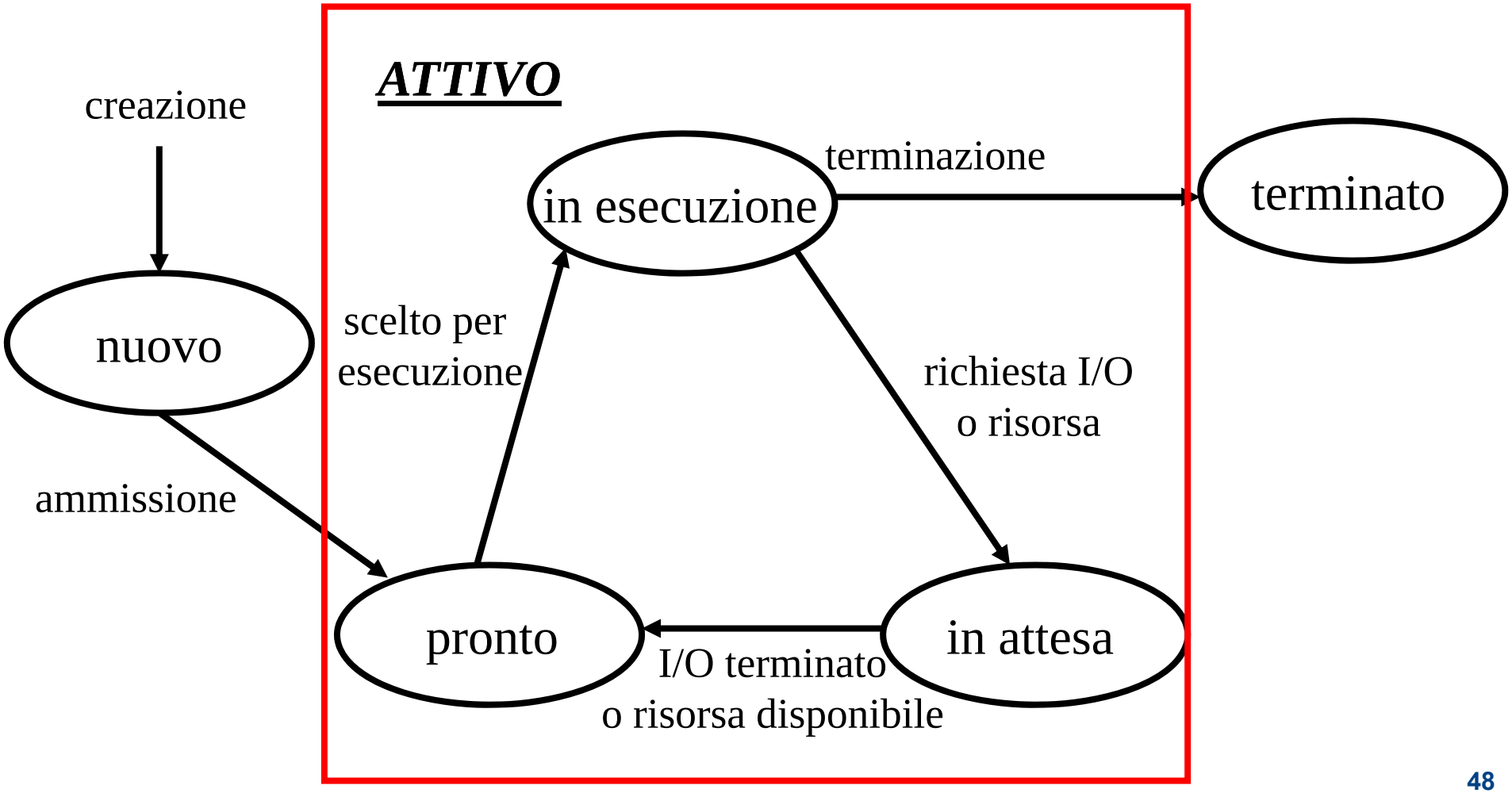
- **Ci limiteremo ad analizzare alcuni esempi delle principali politiche base per la schedulazione a breve termine (o schedulazione della CPU)**

Gestione del processore (e dei processi) (16)
[Politiche di schedulazione della CPU: politiche preemptive/non preemptive, fair/unfair]

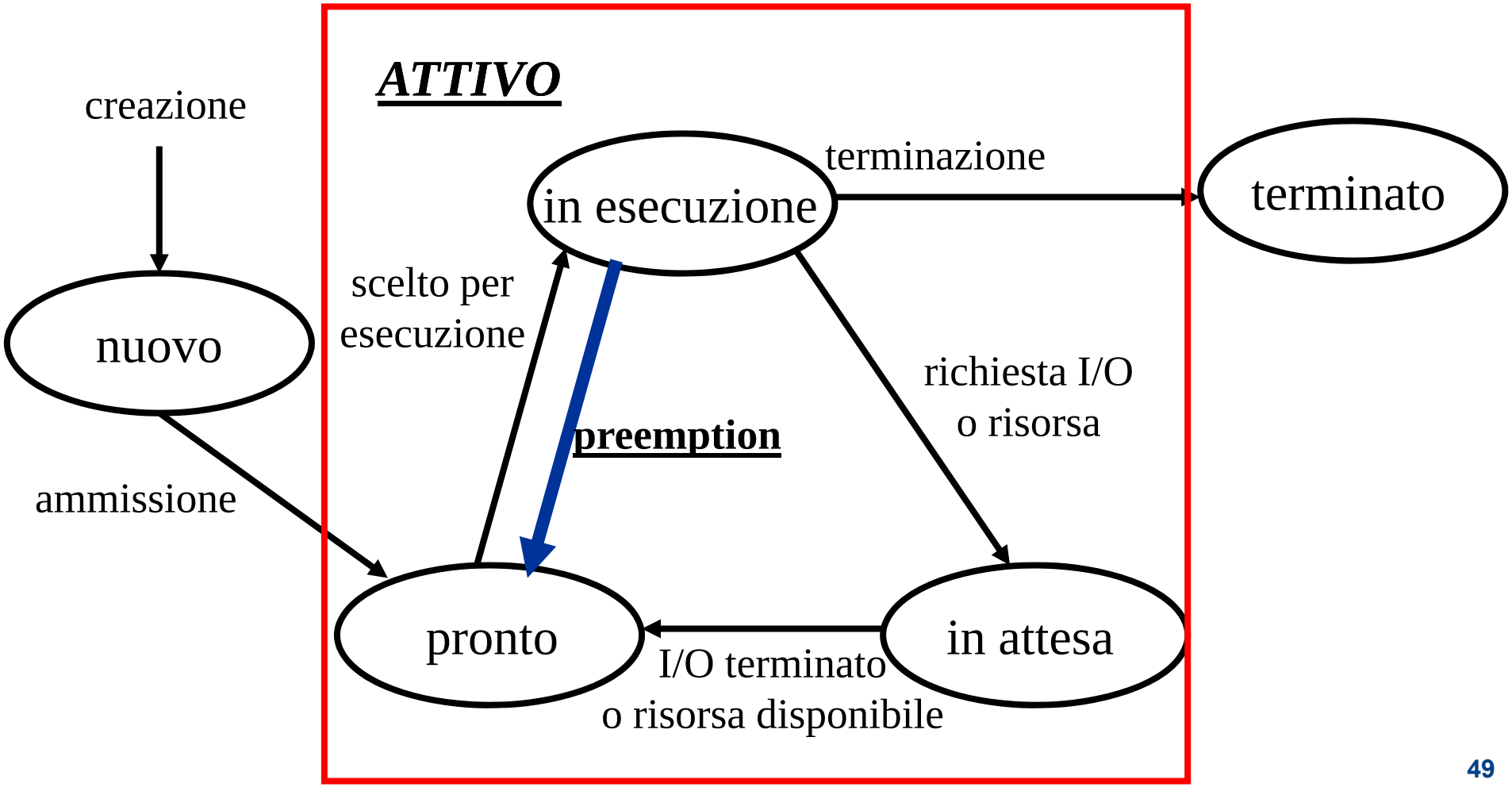
- **Una politica di schedulazione della CPU si dice preemptive se il sistema operativo può bloccare un processo in esecuzione, riaccodarlo nella coda dei pronti e poi selezionare in tale coda un processo a cui attribuire la CPU**
- **Una politica di schedulazione della CPU si dice non preemptive se l'abbandono dell'esecuzione da parte di un processo può avvenire solo "volontariamente", cioè per terminazione o perché il processo va in attesa del verificarsi di qualche evento**

- **Una politica di schedulazione della CPU si dice fair (equa) se essa garantisce che ad ogni processo nella coda dei pronti sarà prima o poi assegnata la CPU**
- **Una politica di schedulazione della CPU si dice unfair (non equa) se essa non garantisce che ad ogni processo nella coda dei pronti sarà prima o poi assegnata la CPU (quindi ammette la possibilità che un processo attenda indefinitamente nella coda dei pronti)**

Gestione del processore (e dei processi) (18)
[Possibili transizioni di stato in caso di politica non preemptive]



Gestione del processore (e dei processi) (19)
[Possibili transizioni di stato in caso di politica preemptive]



Gestione del processore (e dei processi) (20)
[Politica FIFO (First In First Out) o FCFS (First Come First Served)]

- **Si tratta di una politica non-preemptive in cui i processi sono eseguiti nell'ordine in cui vengono accodati nella coda dei pronti**

- **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 dei pronti**

Gestione del processore (e dei processi) (21)
[Politica FIFO (First In First Out) o FCFS (First Come First Served)]

- **Vantaggi:**
  - è di semplice implementazione
  - è fair

- **Svantaggi:**
  - in genere comporta un tempo medio di attesa nella coda dei pronti piuttosto lungo
  - l'uso della CPU e dei dispositivi di I/O può essere eccessivamente basso
  - non adatta a sistemi interattivi

Gestione del processore (e dei processi) (22)
[Politica SJF (Shortest Job First)]

- **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**

- **Se vi sono più processi che richiedono lo stesso tempo minimo per terminare o fermarsi in attesa, la scelta fra questi avviene con politica FCFS**

Gestione del processore (e dei processi) (23)
[Politica SJF (Shortest Job First)]

- **Vantaggi:**
  - si può dimostrare che questa politica minimizza il tempo medio di attesa nella coda dei pronti di un dato insieme di processi

- **Svantaggi:**
  - è solo teorica (cioè non può essere realizzata in pratica!): siccome non si puo` conoscere il futuro, si puo` solo approssimare la politica SJF, stimando il tempo di CPU che ciascun processo richiederà per terminare o fermarsi in attesa
  - è unfair

Gestione del processore (e dei processi) (24)
[Politica SRTF (Shortest Remaining Time First)]

- **Variante *preemptive* 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 o per fermarsi in attesa**

- **Il confronto deve essere fatto tra il tempo di esecuzione del nuovo processo (prima di terminare o di fermarsi in attesa) e quello che manca al processo in esecuzione per terminare o fermarsi in attesa**

Gestione del processore (e dei processi) (25)
[Politica SRTF (Shortest Remaining Time First)]

- **Come la SJF e` solo teorica e ne può essere realizzata solo un'approssimazione (come per la SJF)**

- **In generale, ha vantaggi e svantaggi analoghi a quelli della SJF**

Gestione del processore (e dei processi) (26)
[Politica A Priorità]

- **Ad ogni processo è associata una priorità**
- **Si seleziona dalla coda dei pronti il processo a piu` alta priorità (se vi sono più processi con la stessa priorità più alta, si sceglie fra questi con politica FCFS)**
- **Può essere sia preemptive che non preemptive:**
  - preemptive: se durante l'esecuzione di P1 entra nella coda dei pronti P2 con priorita` piu` alta di P1, il processore viene sottratto a P1 e assegnato a P2;
  - non-preemptive: a nessun processo P1 in esecuzione viene sottratta la CPU, neanche se nella coda dei pronti e` sopraggiunto qualche processo P2 con priorita` piu` alta di P1

Gestione del processore (e dei processi) (27)
[Politica A Priorità]

- **Vari criteri per assegnare le priorità ai processi:**
  - i processi “critici” hanno priorità più alte;
  - processi interattivi prioritari rispetto ai batch;
  - processi certi di utenti prioritari rispetto a quelli di altri;
  - tutti i processi ammessi con la stessa priorita`, ma la loro priorita` decresce col passare del tempo;
  - ...

Gestione del processore (e dei processi) (28)
[Politica A Priorità]

- **Vantaggi**
  - è una politica generale e flessibile (tutte quelle viste finora possono essere considerate casi particolari di questa)

- **Svantaggi**
  - senza qualche “correttivo” (es. meccanismi di “invecchiamento” che aumentino la priorità di un processo in funzione del tempo passato dal processo nella coda dei pronti) è unfair

Gestione del processore (e dei processi) (29)
[Politica RR (Round Robin)]

- **È una politica preemptive basata sul concetto di** quanto di tempo **(**time-slice**)**
- **Ad ogni processo che viene mandato in esecuzione è assegnato lo stesso quanto di tempo prefissato (di solito circa una decina di millisecondi)**
- **Se termina o va in attesa entro il quanto di tempo, viene selezionato il primo processo nella coda dei pronti**
- **Se non termina e non va in attesa entro il quanto di tempo (cioè scade il quanto di tempo), esso deve abbandonare lo stato di esecuzione e il processore viene assegnato al il primo processo nella coda dei pronti**
- **Ogni processo a cui venga sottratta la CPU o termini il proprio stato di attesa viene messo al fondo della coda dei pronti**

Gestione del processore (e dei processi) (30)
[Politica RR (Round Robin)]

- **Vantaggi**

  adatta in un sistema interattivo con molti processi (e, magari,utenti diversi) e in cui si vuole dare l'impressione che l'esecuzione di più processi avvenga in modo contemporaneo

  è fair

- **Svantaggi**

  Il tempo medio di attesa nella coda dei pronti può essere elevato

  La scelta della lunghezza del quanto di tempo influisce pesantemente sulle prestazioni della politica (quindi, bisogna sceglierlo con cura!)

Gestione del processore (e dei processi) (31)
[Process Control Block (Table)]

- **Per gestire i processi, il sistema operativo necessita di mantenersi informazioni su ogni singolo processo**

- **➔ ad ogni processo è associato un *descrittore di processo* (detto Process Control Block) contenente le informazioni necessarie al S.O.. Questi descrittori sono raggruppati nella *tabella dei processi* (detta anche Process Control Block Table)**

Gestione del processore (e dei processi) (32)
[Process Control Block]

- **Ogni Process Control Block contiene molte informazioni, fra le quali:**
  1. l'identificatore del processo: ogni processo è identificato da un nome (di solito un numero progressivo) univoco
  2. l'identificatore dell'utente proprietario; ogni processo ha un proprietario e questa informazione è particolarmente importante nel caso di sistemi multi-utente in quanto permette di realizzare meccanismi di protezione
  3. lo stato del processo ("nuovo", "pronto", "in esecuzione", "in attesa")
  4. il valore del registro Program Counter, questo permette di ricordare a che punto è arrivato il processo nel corso dell'elaborazione
  5. il contenuto di tutti i registri della CPU
  6. informazioni sui *file* e sulle *risorse hardware* attualmente in uso
  7. informazioni sull'utilizzo della *memoria centrale* (es. i valori dei registri fance) e della *memoria secondaria*
  8. informazioni per la *schedulazione*
  9. ...

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Gestione della memoria principale (1)

- **Un programma, per essere eseguito, deve risiedere (almeno parzialmente) in memoria principale**

- **Molti processi devono essere eseguiti *contemporaneamente* ➔ devono condividere l'uso della memoria principale**

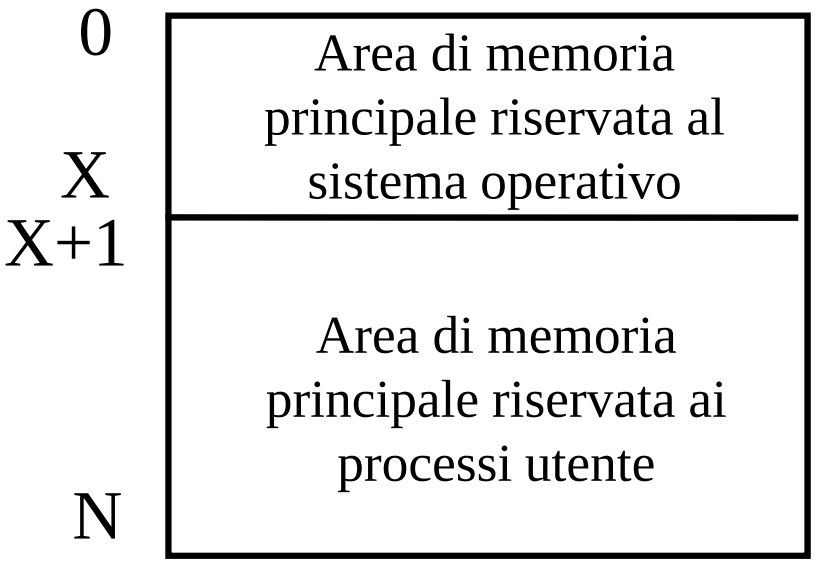
- **Problemi:**
  - Protezione della memoria: come assicurare una corretta condivisione della memoria principale fra i processi contemporaneamente attivi (cioè, come garantire che i processi utente non sovrascrivano aree di memoria principale riservate al sistema operativo? Come garantire che essi non si danneggino a vicenda nell'uso della memoria principale?
  - Ottimizzazione dell'uso delle memoria principale: come ottimizzare l'uso della memoria principale da parte di più processi contemporaneamente attivi?

Gestione della memoria principale (2)

- **Anche il semplice caso di sistema mono-programmato comporta problemi di protezione**

Problema di protezione: impedire che il processo utente acceda alla porzione di memoria riservata al S.O.

X = Indirizzo FENCE (staccionata), memorizzato in un registro del processore (➔ supporto hardware): per ogni accesso in memoria da parte del processo utente, l'indirizzo specificato è accettato solo se è > X, altrimenti il sistema operativo blocca l'esecuzione del processo

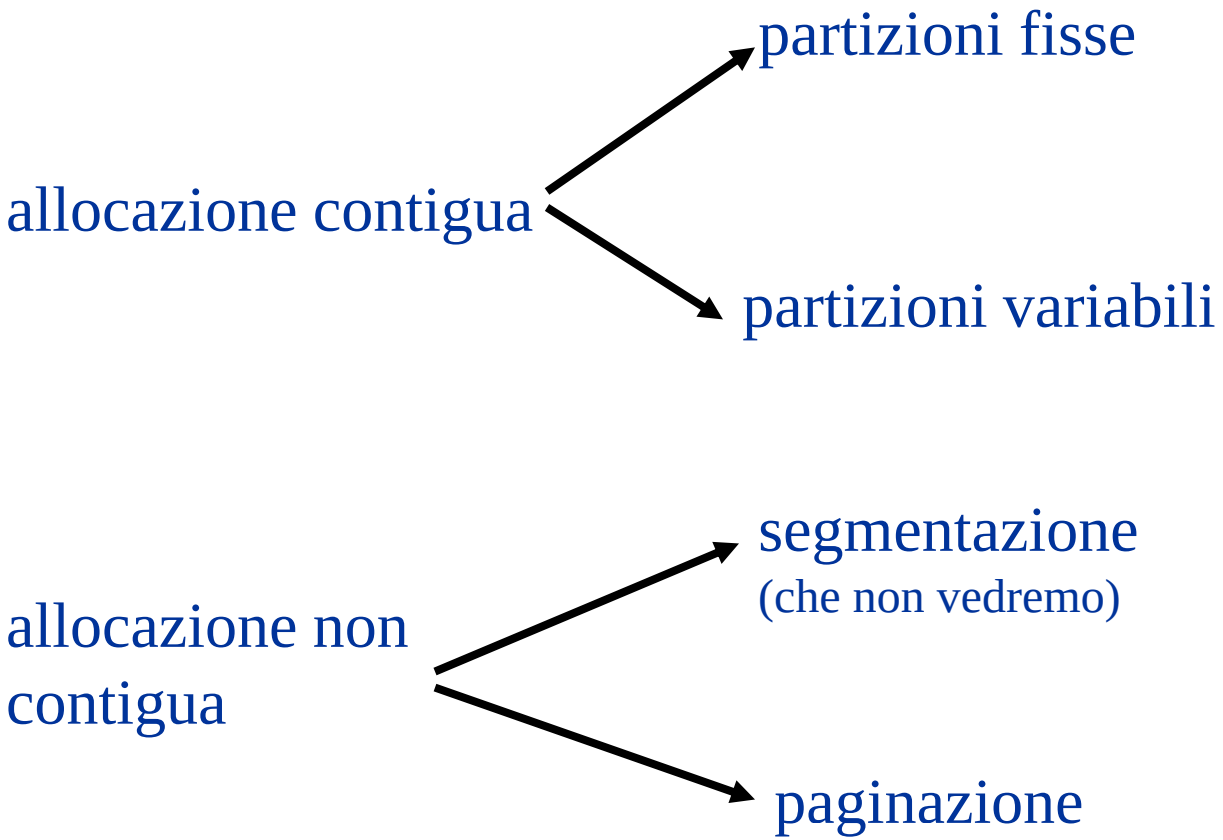


Gestione della memoria principale (3)

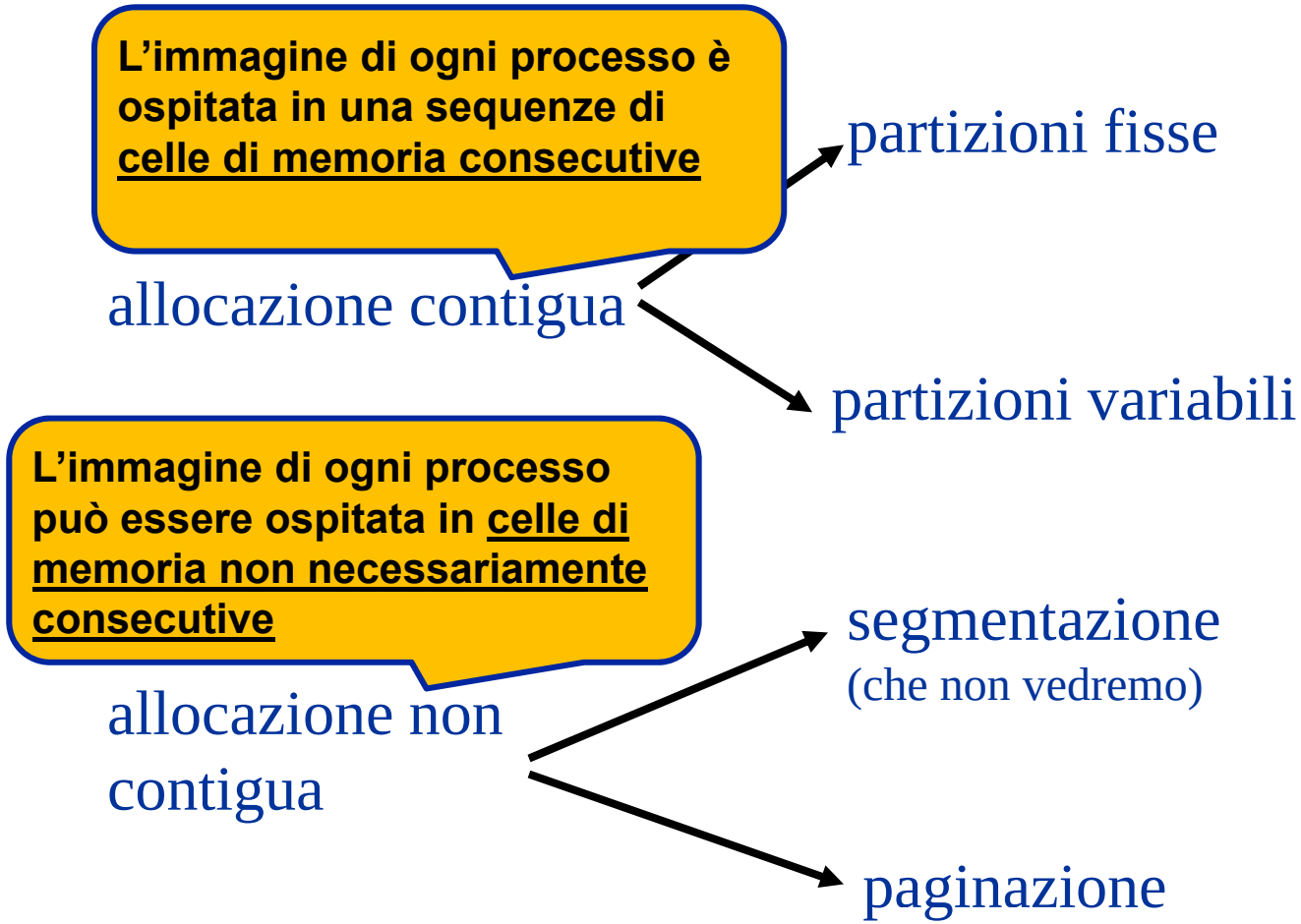
- **Nei sistemi multi-programmati, più *immagini di processi* (immagine del processo = istruzioni del programma + dati), o porzioni di immagini, sono contemporaneamente presenti in memoria principale**

- **Vi possono essere varie strategie per mantenere in memoria principale più immagini (o porzioni di immagini) di processi contemporaneamente:**

Gestione della memoria principale (4)



Gestione della memoria principale (4)



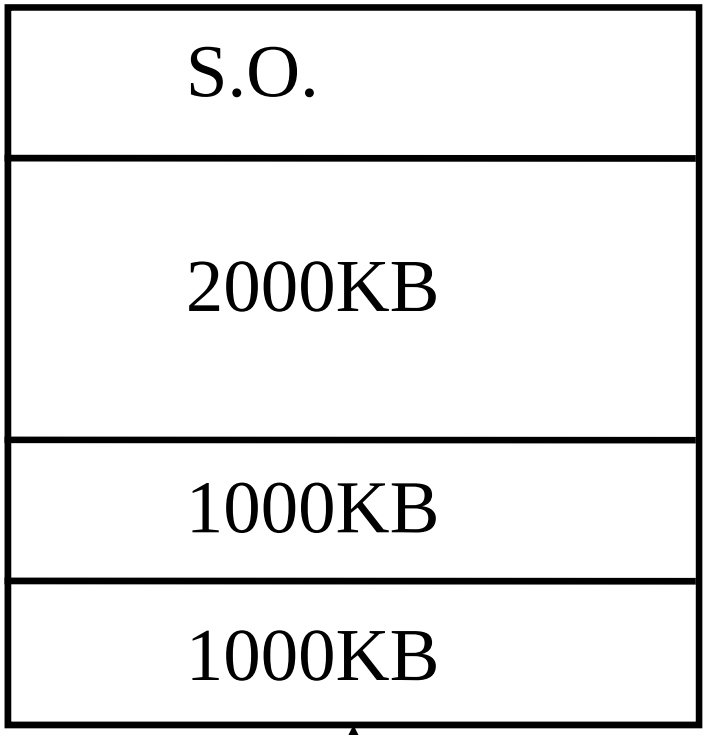
Gestione della memoria principale (5)
[Allocazione contigua con partizioni fisse]

- **Suddivisione della memoria processi in partizioni di dimensione fissa (scelta una volta per tutte nella fase di configurazione del sistema operativo; non necessariamente tutte le partizioni hanno la stessa dimensione)**

- **L'immagine di ogni processo in memoria risiede interamente in un'unica partizione (allocazione contigua)**

- **Problema di protezione: ogni processo deve poter accedere solo alla partizione in cui risiede ➔ due registri fence per ogni processo in esecuzione che ne delimitano la partizione (➔ supporto hardware)**

Gestione della memoria principale (6)
[Allocazione contigua con partizioni fisse]



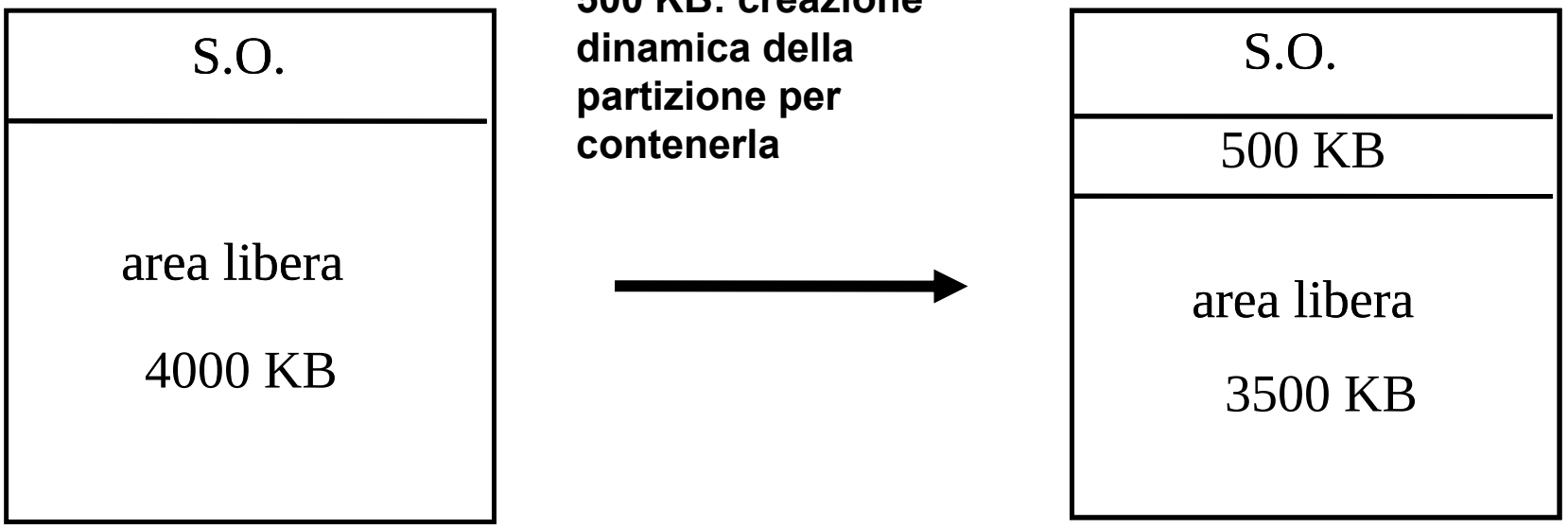
Vantaggio: è piuttosto semplice e non comporta grandi problemi di gestione per il S.O.

Svantaggi:
- è molto rigida (ad esempio, si pongono limiti sulle dimensioni dei processi eseguibili)
- può comportare spreco di memoria sia all'interno di ogni partizione che complessivamente un processo che richiede 2500 KB non può essere eseguito!!

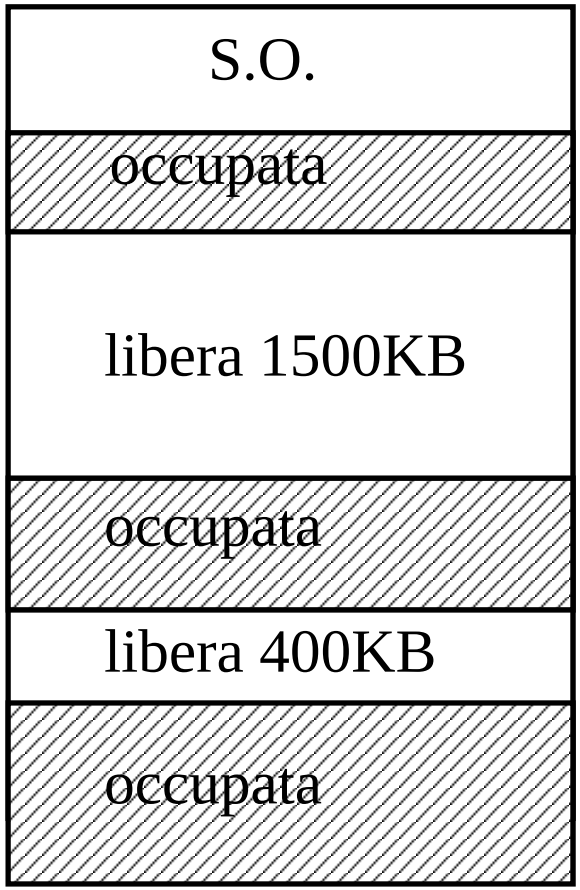
Gestione della memoria principale (7)
[Allocazione contigua con partizioni variabili]

- **Il numero delle partizioni e le loro dimensioni non è scelto a priori**
- **Le partizioni vengono create dinamicamente secondo le necessità del momento**
- **Come nel caso delle partizioni fisse, l'immagine di ogni processo in memoria è interamente contenuta in un'unica partizione (allocazione contigua)**
- **Il meccanismo di protezione è analogo a quello per le partizioni fisse, basato su due registri fence per il processo in esecuzione**

Gestione della memoria principale (8)
[Allocazione contigua con partizioni variabili]



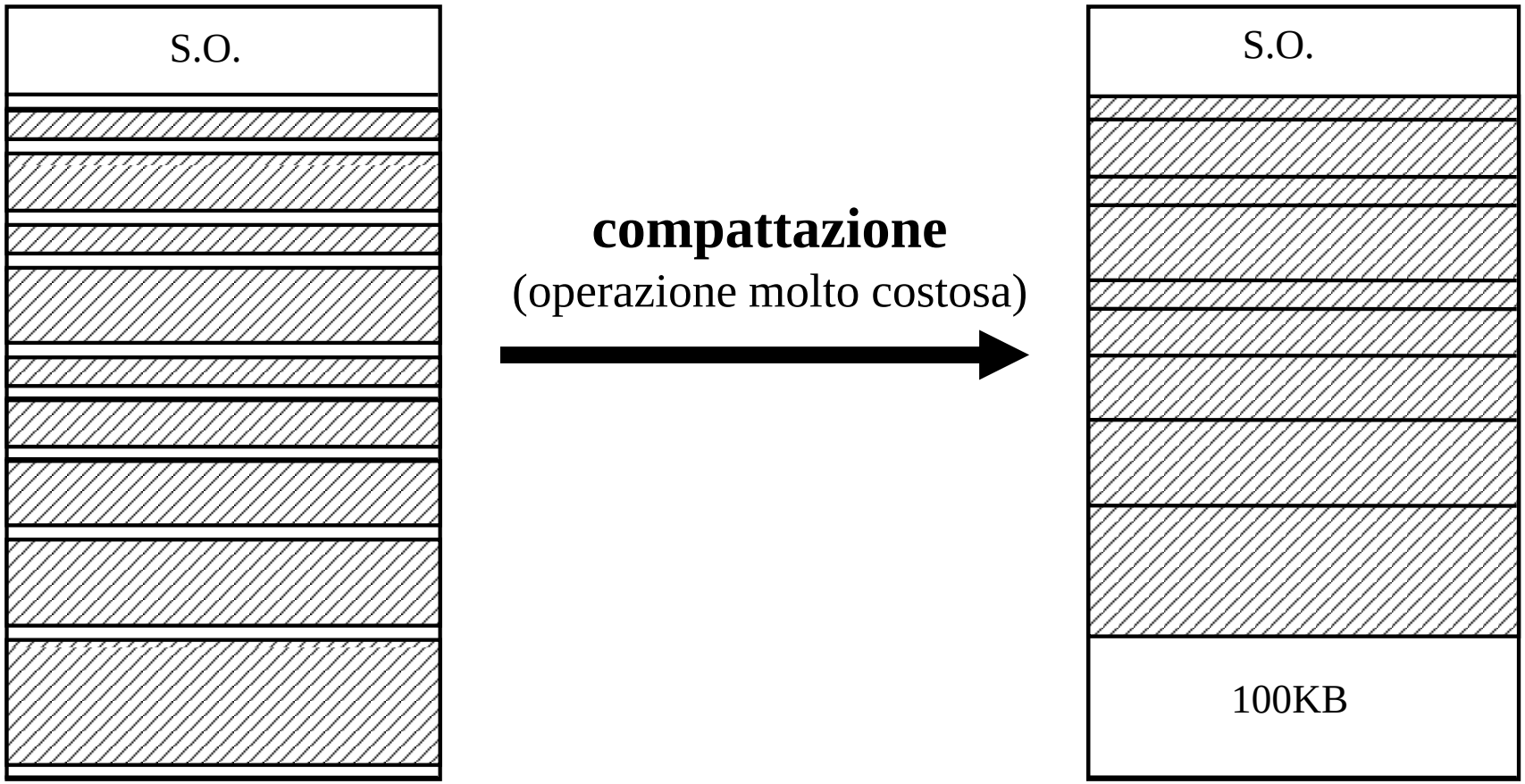
Gestione della memoria principale (9)
[Allocazione contigua con partizioni variabili]



•In un generico istante, la memoria e` suddivisa in partizioni libere e partizioni occupate, in numero e dimensione variabili

•**Non c'è spreco di spazio internamente alle partizioni, ma può esservi uno di spazio globale** (es.: nel caso illustrato, supponiamo di voler caricare un processo di 1600 KB: lo spazio necessario sarebbe fisicamente disponibile, ma è mal distribuito, pertanto inutilizzabile per il nuovo processo)...possibile soluzione, la compattazione

Gestione della memoria principale (10)
[Allocazione contigua con partizioni variabili]



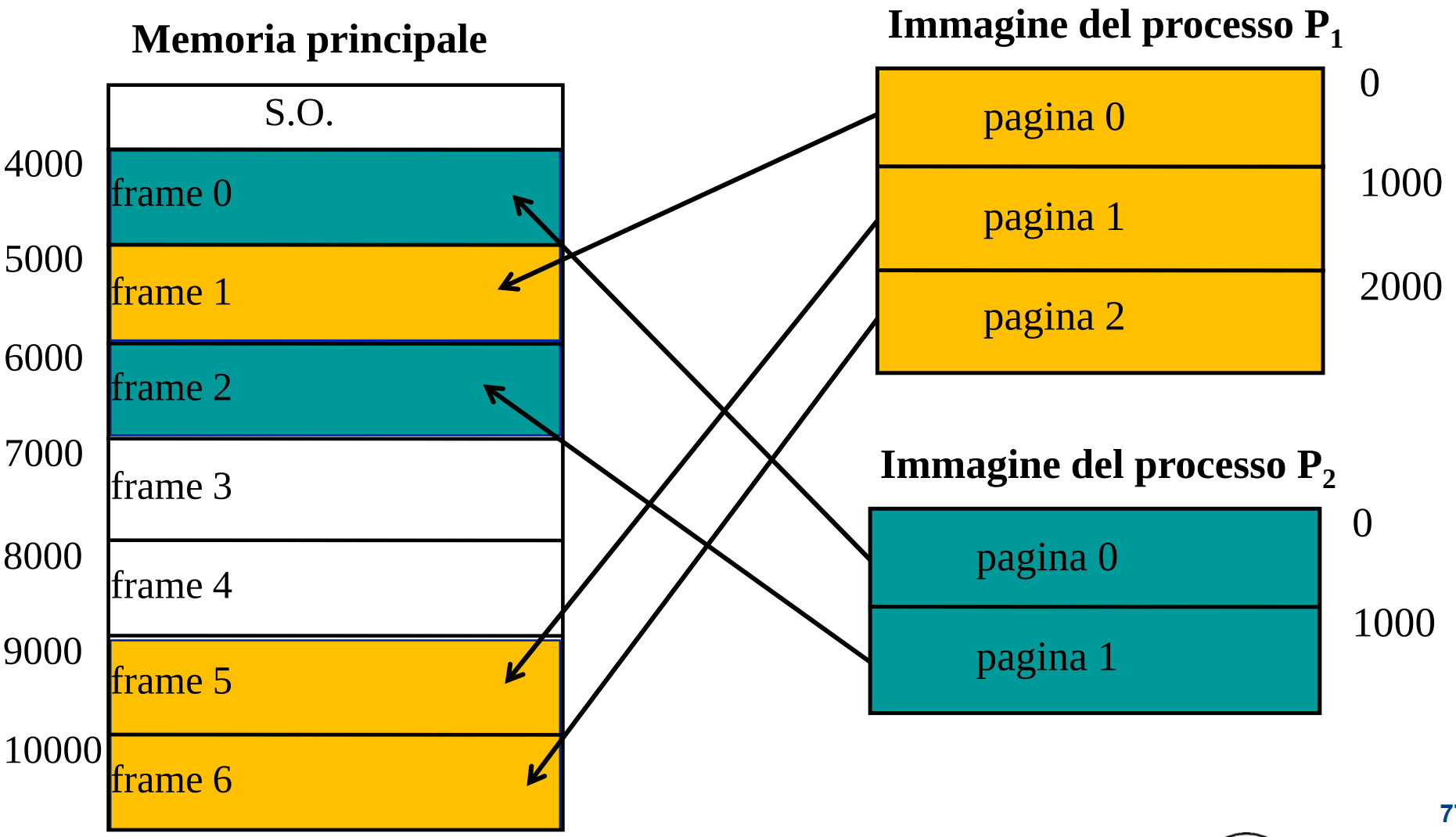
Gestione della memoria principale (11)
[Allocazione non contigua con paginazione]

- **Nell'allocazione contigua si assume che l'immagine di un processo sia un blocco unico da caricare in memoria in modo indivisibile**
- **Questa restrizione cade nel caso dell'allocazione non contigua**
- **Vediamo la tecnica di *paginazione***

Gestione della memoria principale (12)
[Allocazione non contigua con paginazione]

- La memoria principale viene divisa in un insieme di **blocchi** (detti **frame**), tutti della stessa dimensione (solitamente tra 512 Byte e 16 MB)
- Le immagini dei processi vengono suddivise in **pagine** uguali tra loro e aventi la stessa dimensione dei blocchi di memoria
- Le pagine che costituiscono l'immagine di un processo vengono caricate in memoria principale (ciascuna in un distinto frame) in modo sparso
- Il sistema operativo tiene traccia, per mezzo del Process Control Block di ogni processo, di dove sono caricate le pagine del processo

Gestione della memoria principale (13)
[Allocazione non contigua con paginazione]



Gestione della memoria principale (14)
[Allocazione non contigua con paginazione]

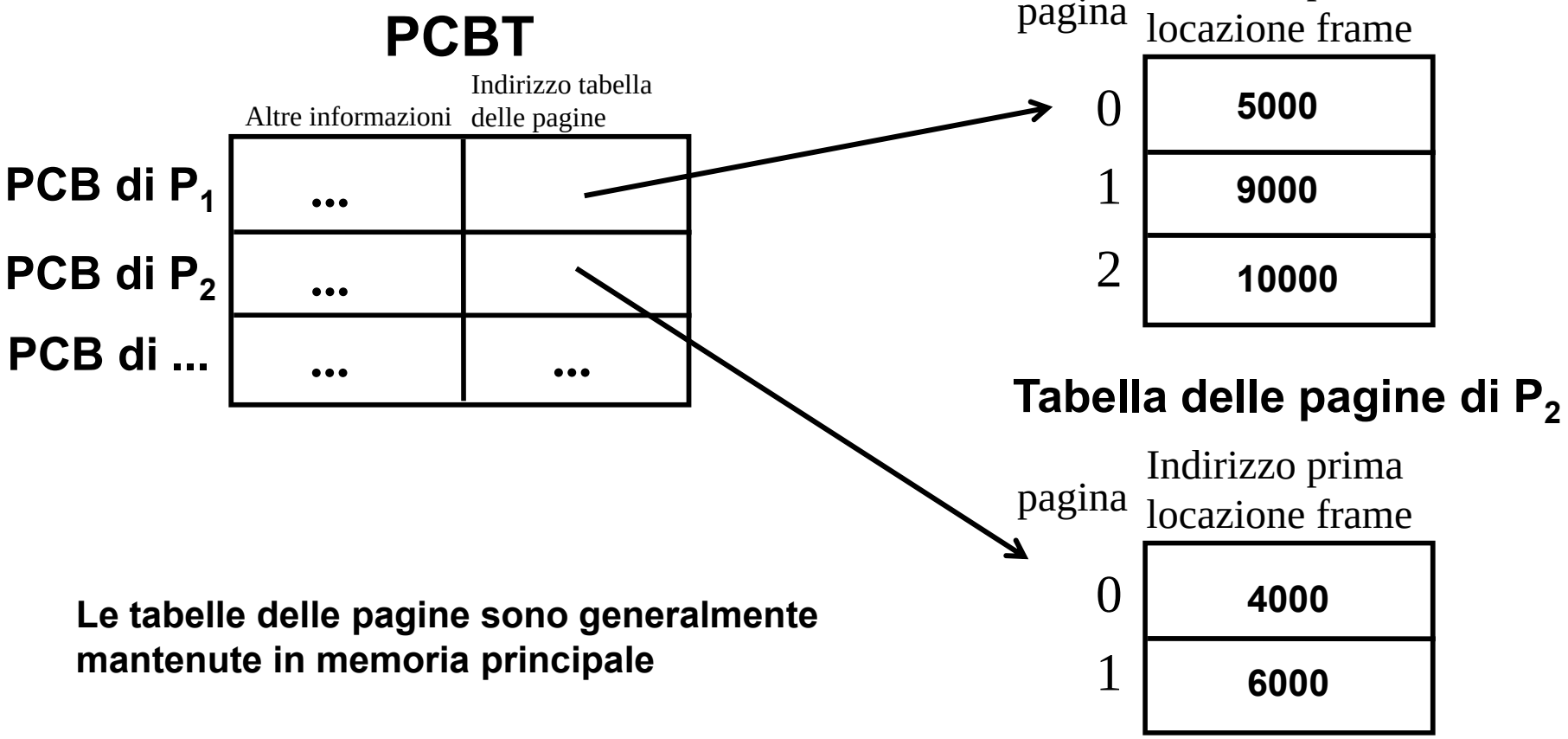
- **Per realizzare il meccanismo di paginazione, il sistema operativo deve conoscere qual è la situazione di ogni frame (libero, occupato, ...) e, per ogni processo la cui immagine è presente in memoria principale, deve conoscere in quale frame è memorizzata ogni pagina**

- **A tal fine, il sistema operativo mantiene aggiornate una tabella dei frame che specifica (assieme ad altre informazioni che qui non riportiamo), per ogni frame, se esso è libero oppure no**

- **Inoltre, esso mantiene, per ogni processo, una tabella delle pagine che specifica (assieme ad altre informazioni, alcune delle quali vedremo in seguito) in quale frame essa è memorizzata (ad esempio specificando l'indirizzo della prima locazione del frame che contiene la pagina)**

Gestione della memoria principale (15)
[Allocazione non contigua con paginazione: tabelle delle pagine]

Nel caso dell'ultimo esempio:



Le tabelle delle pagine sono generalmente mantenute in memoria principale

Gestione della memoria principale (16)

[Allocazione non contigua con paginazione: indirizzi logici (o virtuali) e indirizzi fisici]

- **Siccome l'immagine di un processo può essere caricata in un qualunque insieme di frame liberi in memoria principale, le istruzioni nel codice dei processi non possono fare riferimento direttamente agli indirizzi di memoria principale...nel momento in cui il codice è creato, non si sa quali frame gli verranno riservati**

- **In un sistema paginato (ma questo è vero anche per altre situazioni, che qui non abbiamo esplicitato), le istruzioni macchina di accesso alla memoria non specificano i veri indirizzi di memoria principale (*indirizzi fisici*), ma fanno riferimento ad un'immagine logica della memoria come se essa fosse costituita dalle pagine che compongono l'immagine del processo ed esprimono gli indirizzi come coppia di numeri**

  <*num pag*, *offset*> (*indirizzi logici o virtuali*)

  ***num pag* = il numero della pagina in cui si trova la cella;**

  ***offset* = l'indirizzo relativo all'interno di tale pagina (offset)**

Gestione della memoria principale (17)

[Allocazione non contigua con paginazione: indirizzi logici (o virtuali) e indirizzi fisici]

- ➔ le istruzioni (e quindi la CPU) specificano indirizzi logici
- Le locazioni di memoria possono essere identificate solo in base ad indirizzi fisici

- La traduzione da indirizzo logico ad indirizzo fisico richiede un supporto hardware ed è eseguito dalla Memory Management Unit, sfruttando le informazioni contenute nelle tabelle delle pagine

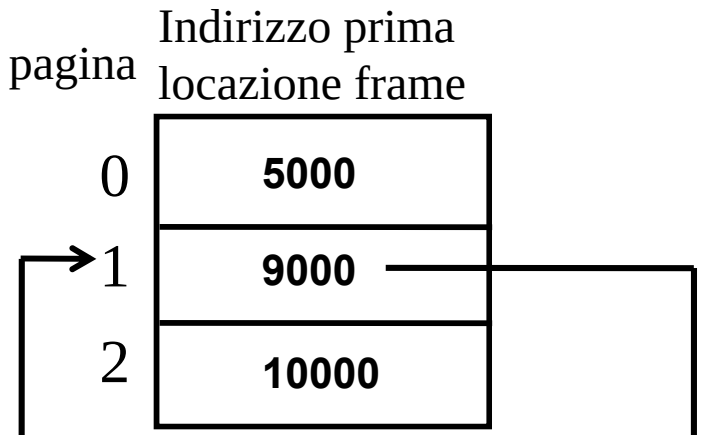
- Es., supponiamo che un'istruzione macchina nel codice del processo P_1 dell'esempio precedente specifichi l'indirizzo logico <1, 205> (cioè, sia un'istruzione che specifica un accesso alla locazione 205 della pagina 1 del processo P_1)[1]:

1 Le locazioni all'interno di ogni pagina sono numerate a partire da 0 (zero)

Gestione della memoria principale (18)

[Allocazione non contigua con paginazione: indirizzi logici (o virtuali) e indirizzi fisici]

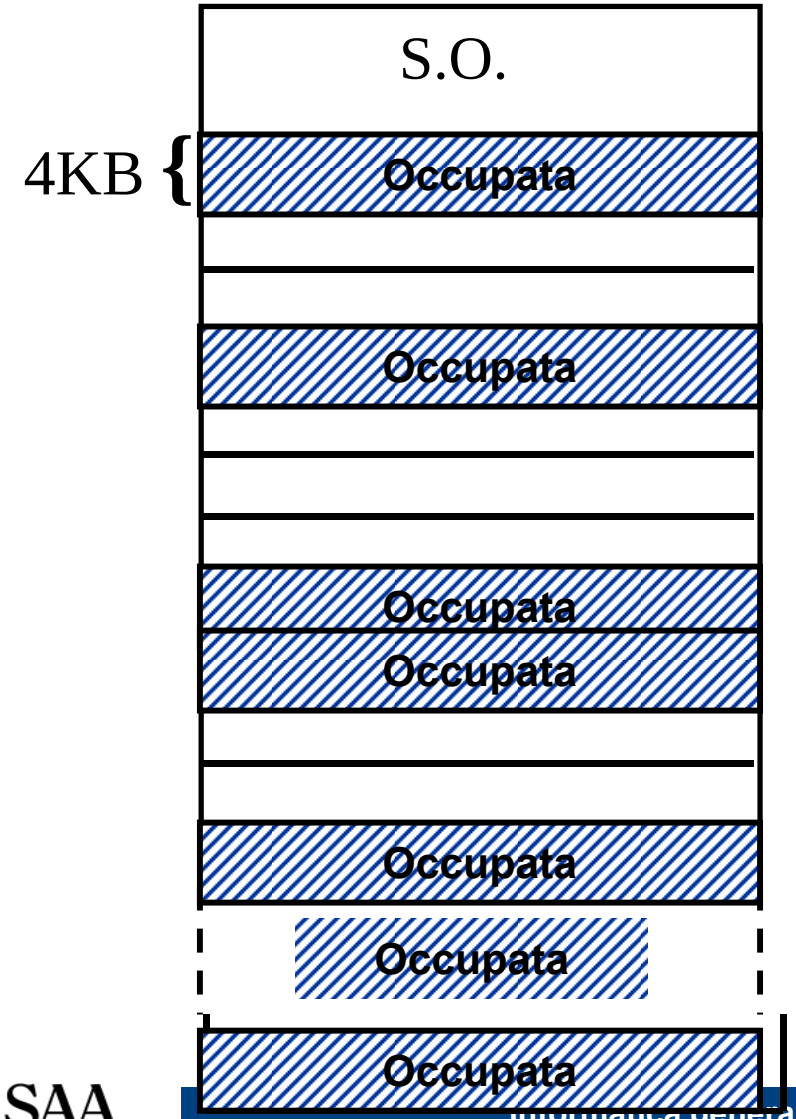
Tabella delle pagine di P_1



Indirizzo logico <1, 205> → **offset** + **Indirizzo fisico della prima locazione del frame che contiene la pagina 1** =

= **9205 Indirizzo fisico**

Gestione della memoria principale (19)
[Allocazione non contigua con paginazione]



Nel caso illustrato in questo esempio, un processo con un' immagine da 13 KB può essere caricato

N.B. Lo spreco di spazio è molto contenuto e limitato ad un eventuale spreco di nel frame che contiene l'ultima pagina di ogni processo

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Memoria virtuale (1)

- Obiettivo: **poter eseguire più processi le cui immagini, complessivamente, possono richiedere più memoria principale di quella fisicamente disponibile o addirittura poter eseguire processi le cui immagini singolarmente sono più grandi della dimensione fisica della memoria principale disponibile ➔** astrazione dalla dimensione della memoria fisica**!**
- **➔ ad ogni processo è nascosto il fatto che altri processi usano la memoria e quindi esso "avrà l'impressione" di avere la memoria tutta per sé...addirittura, è possibile fare in modo che ogni processo possa avere l'impressione di disporre di più memoria principale di quella fisicamente presente**
- **Questa visione astratta della memoria è detta** memoria virtuale

Memoria virtuale (2)

Come si realizza la memoria virtuale? Vi sono due tecniche (non mutuamente esclusive):

1. **Swapping: le immagini dei processi vengono alternate all'interno della memoria centrale, nello stesso modo in cui il processore viene alternato tra i processi (non la vedremo)**
2. **Demand paging (o paginazione su richiesta): è basata sul meccanismo di paginazione e le immagini dei processi vengono caricate nella memoria centrale "a pezzi"**

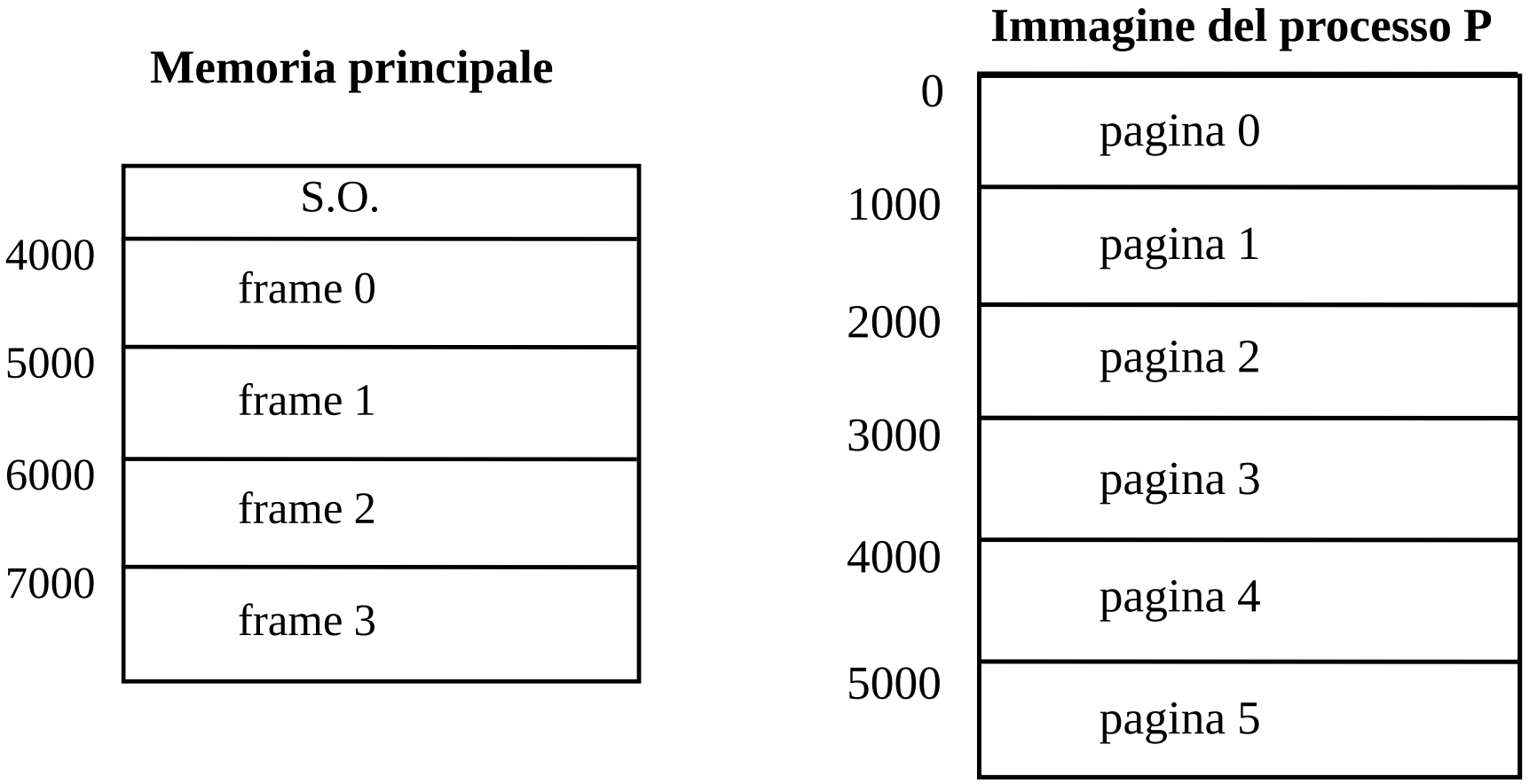
Memoria virtuale (3)
[Demand Paging o Paginazione su richiesta]

- **Idea**: ***se è presente il meccanismo di paginazione, è possibile tenere in memoria principale solo alcune pagine delle immagini dei processi, e caricare altre pagine (copiandole dalla memoria secondaria a quella principale) solo quando servono***

- **Questa tecnica consente di eseguire processi con immagini di dimensione superiore a quella dell'area di memoria centrale riservata ai processi utente**

- **➔ astrazione dalla macchina fisica!!**

Memoria virtuale (3)
[Demand Paging o Paginazione su richiesta]



Memoria virtuale (4)
[Demand Paging o Paginazione su richiesta]

- **Il processo P puo` assumere di avere a disposizione una memoria centrale grande quanto la sua immagine (6000 bytes, nell'esempio), anche se l'area di memoria riservata ai processi utente ha, in realta`, dimensione inferiore (4000 bytes, nell'esempio)**

- **In altre parole, il processo P ha a disposizione una memoria virtuale di 6000 bytes e a questa P fa riferimento. La dimensione della memoria fisica non lo riguarda (quasi) piu`!**

Memoria virtuale (5)
[Demand Paging o Paginazione su richiesta]

- **In un certo istante, un sottoinsieme delle pagine di un processo si trovano effettivamente in memoria principale (per poter iniziare l'esecuzione di un processo è necessario caricare in memoria principale almeno la pagina che contiene la prima istruzione)**

- **Ogni processo, come detto prima, fa riferimento ad uno spazio di memoria virtuale e usa indirizzi virtuali (detti anche logici) per accedere alle celle delle sue pagine**
- **Per via del meccanismo di paginazione, un processo "non si preoccupa" di DOVE si trovino (cioè di quale sia il loro indirizzo fisico) le sue pagine che sono state caricate in memoria principale (già detto prima)...**

Memoria virtuale (6)
[Demand Paging o Paginazione su richiesta]

- **La novità, con il meccanismo della paginazione su richiesta è che un processo “non si preoccupa” nemmeno di QUALI sue pagine si trovano realmente in memoria principale in un certo istante**

- **...chi si preoccupa di cio`? Il S.O.!**

- **Per supportare la paginazione su richiesta, le tabelle delle pagine devono contenere alcune informazioni in più, come illustrato dal seguente esempio**

Memoria virtuale (7)
[Demand Paging o Paginazione su richiesta]

Tabella delle pagine di P

pagina	In Memoria principale?	Indirizzo prima locazione frame	Indirizzo di Memoria Secondaria
0	SI	7000	α
1	NO		β
2	NO		γ
3	NO		δ
4	NO		ε
5	NO		ϕ

Memoria virtuale (7)
[Demand Paging o Paginaz

Specifica se una pagina è correntemente presente in memoria principale (SI'), oppure non lo è (NO)

Specifica l'indirizzo fisico della prima cella del frame che contiene la pagina, se questa è correntemente presente in memoria principale, altrimenti non contiene informazione

delle pa… di P

Specifica l'indirizzo di memoria secondaria in cui è contenuta la pagina

pagina	In Memoria principale?	Indirizzo prima locazione frame	Indirizzo di Memoria Secondaria
0	SI	7000	α
1	NO		β
2	NO		γ
3	NO		δ
4	NO		ε
5	NO		ϕ

Nell'esempio: solo la pagina zero del processo P è correntemente presente in memoria principale e si trova nel frame la cui prima locazione ha indirizzo 7000

Memoria virtuale (7)
[Demand Paging o Paginazione su richiesta: page fault]

- **Si utilizza la page table sia per trasformare un indirizzo virtuale in indirizzo fisico (nello stesso modo visto in precedenza), sia per sapere se si è verificato un page fault, cioè il riferimento, da parte del processo in esecuzione, ad una sua pagina che non è correntemente in memoria principale**

- **Il Sistema Operativo interpreta il page fault come la richiesta da parte del processo in esecuzione di caricare in memoria principale una pagina ed effettua tale caricamento (di qui il nome "demand paging" o "paginazione su richiesta")**

- **Quando si verifica un page fault, il processo P in esecuzione viene bloccato e messo nello stato di attesa**

- **Non appena la pagina richiesta viene caricata in memoria, il processo P può ritornare nello stato di pronto**

Memoria virtuale (8)
[Demand Paging o Paginazione su richiesta: page fault]

DA INDIRIZZO VIRTUALE A INDIRIZZO FISICO E GESTIONE DEI PAGE FAULT

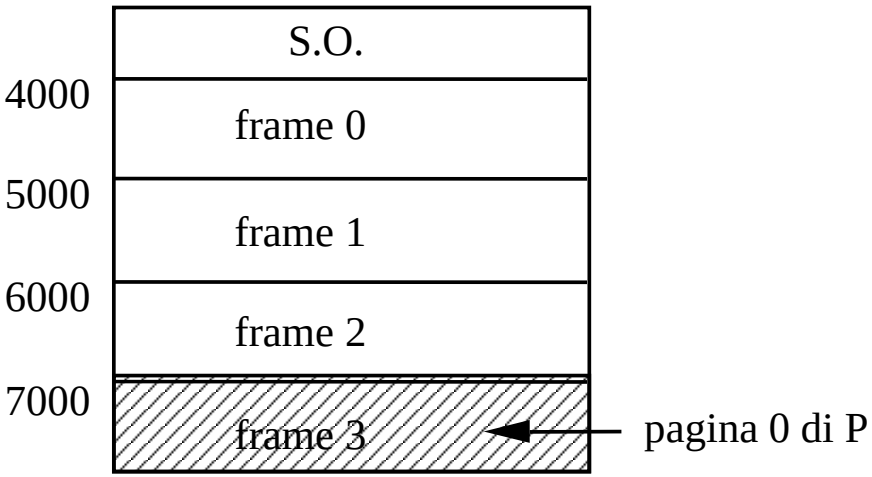
Indirizzo virtuale: <num pag,offset>.

- si guarda se la pagina *num pag* è già in memoria principale **(campo "In Memoria Principale? della tabella delle pagine);**
  - **se SI**, si prende l'indirizzo *i* della prima locazione del frame e gli si somma offset: indirizzo fisico = i+offset (come visto in precedenza);
  - **se NO**, la pagina deve essere caricata e si procede come segue:
    1. viene richiesto il caricamento della pagina dal disco in memoria principale;
    2. il processo P va in attesa;
    3. quando e` terminato il caricamento della pagina, P torna pronto

Memoria virtuale (9)
[Demand Paging o Paginazione su richiesta: page fault]

Esempio...

memoria principale



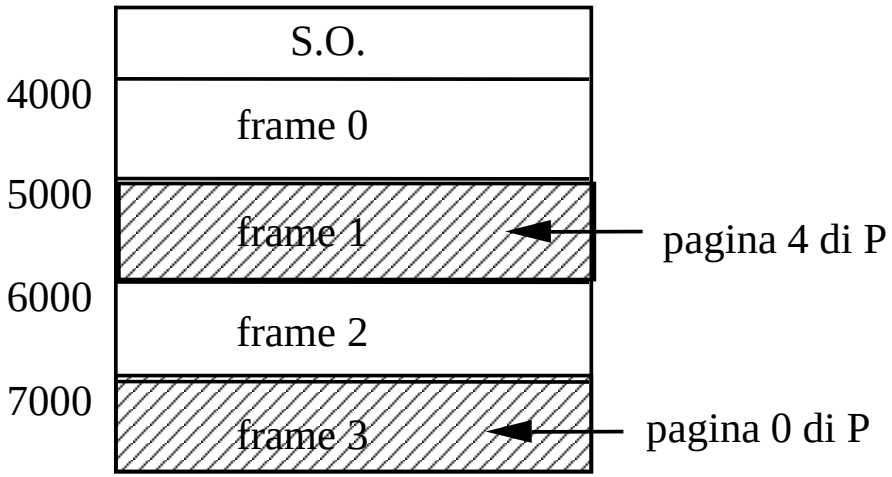
Page table del processo P in esecuzione

pagina	In Mem?	Blocco	Ind Mem Sec
0	SI	7000	α
1	NO		β
2	NO		γ
3	NO		δ
4	NO		ε
5	NO		ϕ

<0, 127> (indirizzo virtuale) → 7127 (indirizzo fisico)
<4, 215> (indirizzo virtuale) → **page fault** → caricamento della pagina 4

Memoria virtuale (10)
[Demand Paging o Paginazione su richiesta: page fault]

Esempio...



Page table del processo P in esecuzione

pagina	In Mem?	Blocco	Ind Mem Sec
0	SI	7000	α
1	NO		β
2	NO		γ
3	NO		δ
4	SI	5000	ε
5	NO		ϕ

<4, 215> (indirizzo virtuale) → 5215 (indirizzo fisico)

Memoria virtuale (11)
[Demand Paging o Paginazione su richiesta: page fault]

- **Cosa succede se si verifica una page fault quando in memoria centrale non vi sono frame liberi?**

- **Il Sistema Operativo sceglie un frame occupato e lo libera! Cioe`, si verifica un rimpiazzamento di pagina: se la pagina da rimpiazzare e` stata modificata, la si copia in memoria secondaria; nel frame contenente la pagina da rimpiazzare viene caricata la pagina richiesta**
- **Come scegliere la pagina da rimpiazzare? L'ideale e` scegliere una pagina che sara` referenziata piu` tardi nel futuro**
- **...siccome non si puo` conoscere il futuro, si approssima la scelta ottimale, ad esempio, scegliendo come pagina da rimpiazzare quella che e` stata usata meno di recente (questa informazione si trova nella frame table): questa politica si chiama Least Recently Used (LRU) e non e` l'unica possibile**

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Gestione delle periferiche

Relativamente alle periferiche (tastiera, video, mouse, stampanti, dischi, nastri, ecc.), il S.O. gioca due ruoli fondamentali:

1. **fornisce all'utente una visione astratta delle periferiche e offre dei semplici comandi per interagire con esse (in questo modo l'utente non e` costretto a conoscerne le caratteristiche fisiche per poterle utilizzare);**
2. **ottimizza l'uso delle periferiche (che devono essere condivise fra vari processi)**

Gestione delle periferiche

- **Le periferiche sono *schiave* della CPU che ne coordina le operazioni**
- **Quando una periferica "vuole attirare l'attenzione" della CPU (ad esempio per notificare l'avvenuta stampa di un insieme di dati, la disponibilita` di un dato di input, ecc.) invia un segnale di *interrupt*;**
- **Ad ogni iterazione, la CPU controlla se sono pervenuti segnali di interrupt; se no, procede nella propria esecuzione; se sì, il processo in esecuzione e` sospeso e viene eseguito il programma di gestione dell'interrupt (ovviamente, in generale, tale programma, detto *driver del dispositivo*, varia da dispositivo a dispositivo)**

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Gestione memoria di massa (File System)

- **Dischi ottici e nastri usati per backup o per distribuire sw, mai come supporto in linea**
- **Dischi magnetici, invece, usati come supporto in linea per**
  - supporto alla memoria principale (es. per memoria virtuale)
  - memorizzazione permanente di dati e programmi
- **Primo punto gia` visto, ci concentriamo sul secondo (File System)**

File System

- **I dati sono logicamente organizzati in file e memorizzati su supporti di memoria di massa: qui considereremo i file memorizzati su disco**
- **Il file system e` un insieme di programmi con due scopi:**
  - fornire all'utente una visione logica dei file
  - gestire i file nella memoria di massa

- Il file system e` un insieme di programmi con due scopi:
  - fornire all'utente una visione logica dei file
  - gestire i file nella memoria di massa

Organizzazione logica dei file

- **Il File System deve permettere all'utente di:**
  - identificare **ogni file con un nome (filename) astraendo completamente dalla sua memorizzazione fisica**
  - **avere un insieme di** comandi **per lavorare sui file: creare o rimuovere un file, copiarlo, cambiargli nome, inserire informazioni in un file (il tutto indipendentemente dal tipo di memorizzazione)**
  - **avere la possibilità di** organizzare **un insieme di file in sottoinsiemi secondo le loro caratteristiche, per avere una visione ordinata e strutturata delle informazioni sul disco**
  - proteggere **i propri file, per impedire ad altri di leggerli, scriverli o cancellarli**

Organizzazione logica dei file

- Logicamente, **i file sono organizzati in una struttura ad albero che ne consente l'organizzazione e suddivisione logica.**

Organizzazione logica dei file

- L'astrazione che permette di raggruppare logicamente un insieme di file e' detta *directory* (detta anche *folder* o *cartella*)
- la directory e' costituita da un insieme di file e altre directory
- Tutti i file system forniscono funzioni per creare, cancellare, rinominare, elencare il contenuto delle directory e copiare o spostare i file da una directory all'altra
- I nomi dei file sono locali alle directory (si possono avere più file con lo stesso nome purché siano in directory diverse)

Organizzazione logica dei file

NOMI DEI FILE

Costituiti da (almeno) due parti:

- nome: **scelto dall'utente, il S.O. in genere impone dei vincoli sulla lunghezza massima (8 caratteri in DOS, 255 in Unix e Windows) e sull'insieme di caratteri usabili (no punteggiatura)**
- estensione: **separata dal nome dal carattere '.', identifica il formato del file o l'applicazione che lo ha creato**

pippo.bmp indica un file immagine in formato bitmap

pippo.doc indica un documento Word

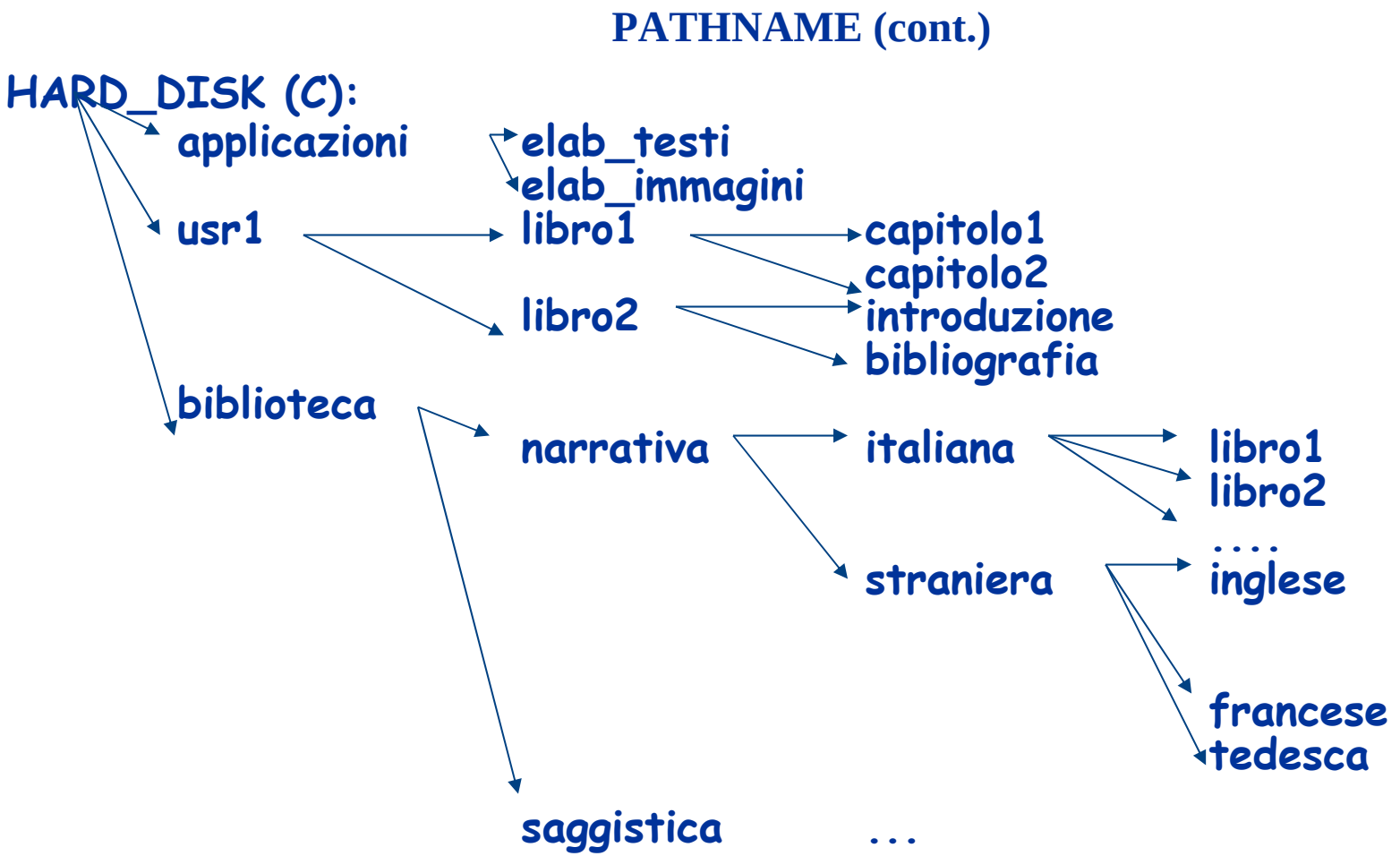
pippo.xls indica un documento Excel

Organizzazione logica dei file

PATHNAME

- Descrive il percorso che si deve compiere per raggiungere un particolare file attraverso i diversi rami dell'albero.
- Ossia: descrive la successione di cartelle che si deve attraversare per raggiungere il file, a partire dalla radice del file system
- Nella stessa cartella non ci possono essere due "elementi" (file o cartelle) con lo stesso nome, quindi il pathname di un determinato file e' sempre unico.

Organizzazione logica dei file



Organizzazione logica dei file

PATHNAME (cont.)

- Ad esempio il file "libro1" di narrativa italiana è univocamente identificato dalla sequenza:

C:\biblioteca\narrativa\italiana\libro1

- Il file "libro1" di narrativa inglese è identificata dalla sequenza

C:\biblioteca\narrativa\straniera\inglese\libro1

Il carattere "\" (backslash) viene usato come separatore nei sistemi DOS e Windows; nei sistemi Unix si usa "/" (slash).

Organizzazione logica dei file

PROTEZIONE

- Ogni utente ha un nome (*login* o *username*) che lo identifica all'interno del sistema; a tale nome è associato un codice segreto, *password*, che l'utente deve digitare per poter accedere al calcolatore.
- All'interno dell'albero dei file, ogni utente ha una directory di cui è proprietario. Egli può gestire il sotto-albero della sua directory creando e rimuovendo i file e le directory solo in questo sotto-albero.
- Di solito questa directory ha lo stesso nome dell'utente.

Organizzazione logica dei file

PROTEZIONE

- Per ogni file (directory) che crea, l'utente può specificare le operazioni consentite su quel file (directory): può permettere solo la lettura, solo la scrittura, sia la lettura che la scrittura.
- In alcuni casi gli utenti vengono suddivisi in *gruppi* e l'utente proprietario del file può specificare quali sono le operazioni consentite ai membri del suo gruppo e quali a tutti gli altri utenti.

- Il file system e` un insieme di programmi con due scopi:
  - fornire all'utente una visione logica dei file
  - gestire i file nella memoria di massa

Organizzazione fisica dei file

Problemi:

- **tenere traccia di quali file siano memorizzati su disco e dove**
- **ottimizzare l'uso dello spazio su disco**

Organizzazione fisica dei file

DEVICE DIRECTORY E FILE DESCRIPTOR

- Il file system riserva una parte dell'hard disk per memorizzare la *device directory*, una tabella che contiene un elemento per ogni file e ogni directory del file system
- Ogni elemento e' detto *file descriptor*
- Il file descriptor di un qualsiasi file e' aggiornato ogni volta che viene compiuta una operazione sul file corrispondente

Organizzazione fisica dei file

FILE DESCRIPTOR

Per un file, il file descriptor indica: **il nome del file, le date di creazione e ultima modifica, la dimensione in byte, il nome del proprietario, la descrizione delle protezioni, il tipo di file, dove si trova fisicamente sull'hard disk.**

Per una directory, il file descriptor indica: **il nome, le date di creazione e ultima modifica, il nome del proprietario, la descrizione delle protezioni, l'elenco dei file e delle sotto-directory contenuti nella directory**

Organizzazione fisica dei file

Problemi: come memorizzare i file su disco e come tener traccia dell'indirizzo fisico del file nel file descriptor?

Analizziamo tre possibilita` :

- Allocazione contigua
- allocazione non contigua linkata
- allocazione non contigua indexata

Organizzazione fisica dei file

- l'hard disk e' suddiviso in blocchi (di dimensione compresa fra 512 byte e 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.**

Vantaggi:

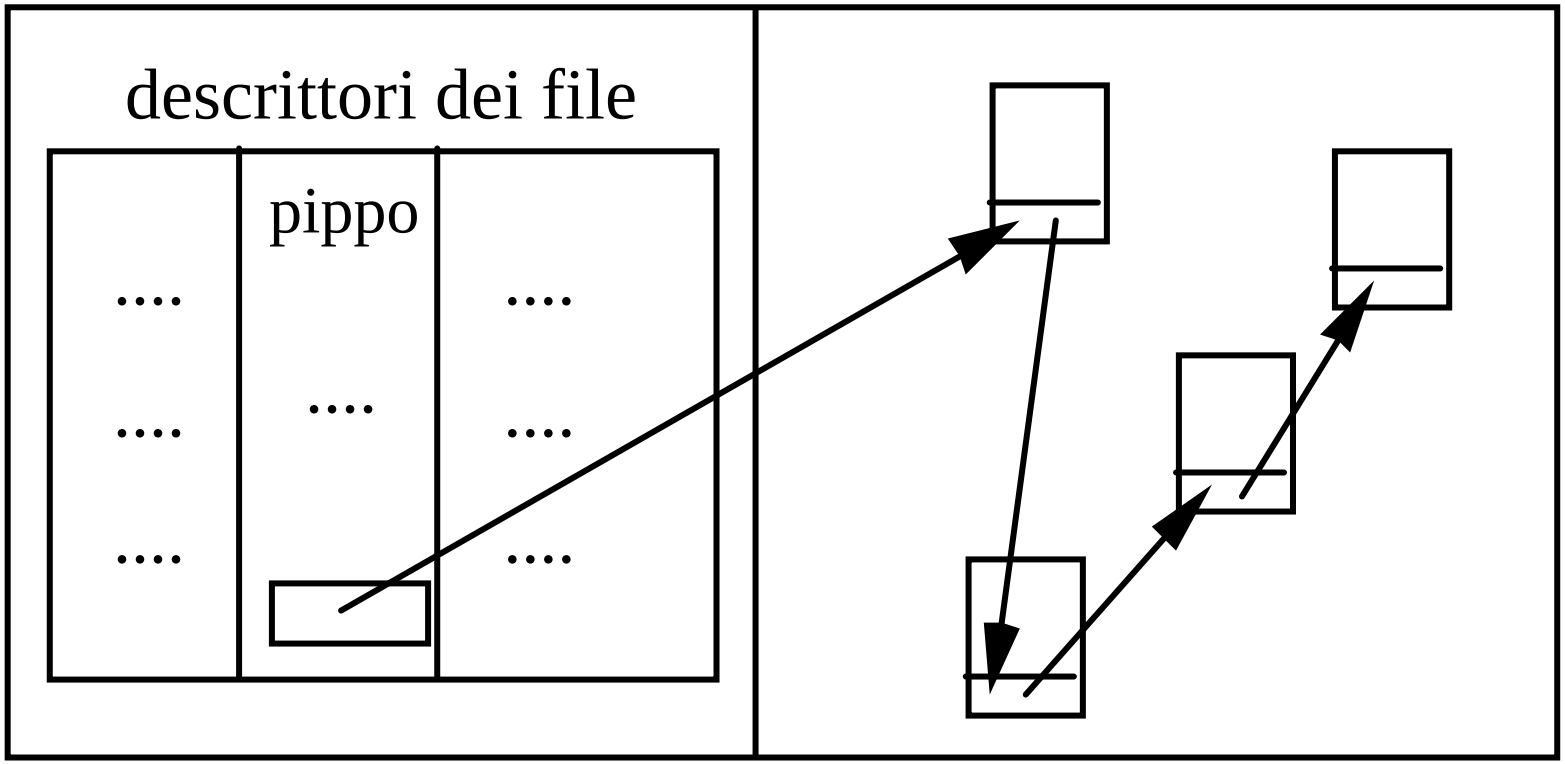
- l'accesso ad ogni blocco e' veloce (e` sufficiente memorizzare l'indirizzo del primo blocco)
- facile l'accesso diretto ai record dei file

Svantaggi:

- Si ha uno spreco di memoria quando nessuna sequenza di blocchi contigui e' sufficiente ad ospitare un file (➔ anche qui frammentazione esterna)
- Se il file aumenta di dimensioni si puo` dover riallocare il file.

ALLOCAZIONE LINKATA

device directory spazio di memorizzazione dei file



ALLOCAZIONE LINKATA (cont.)

- **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**

Vantaggi:

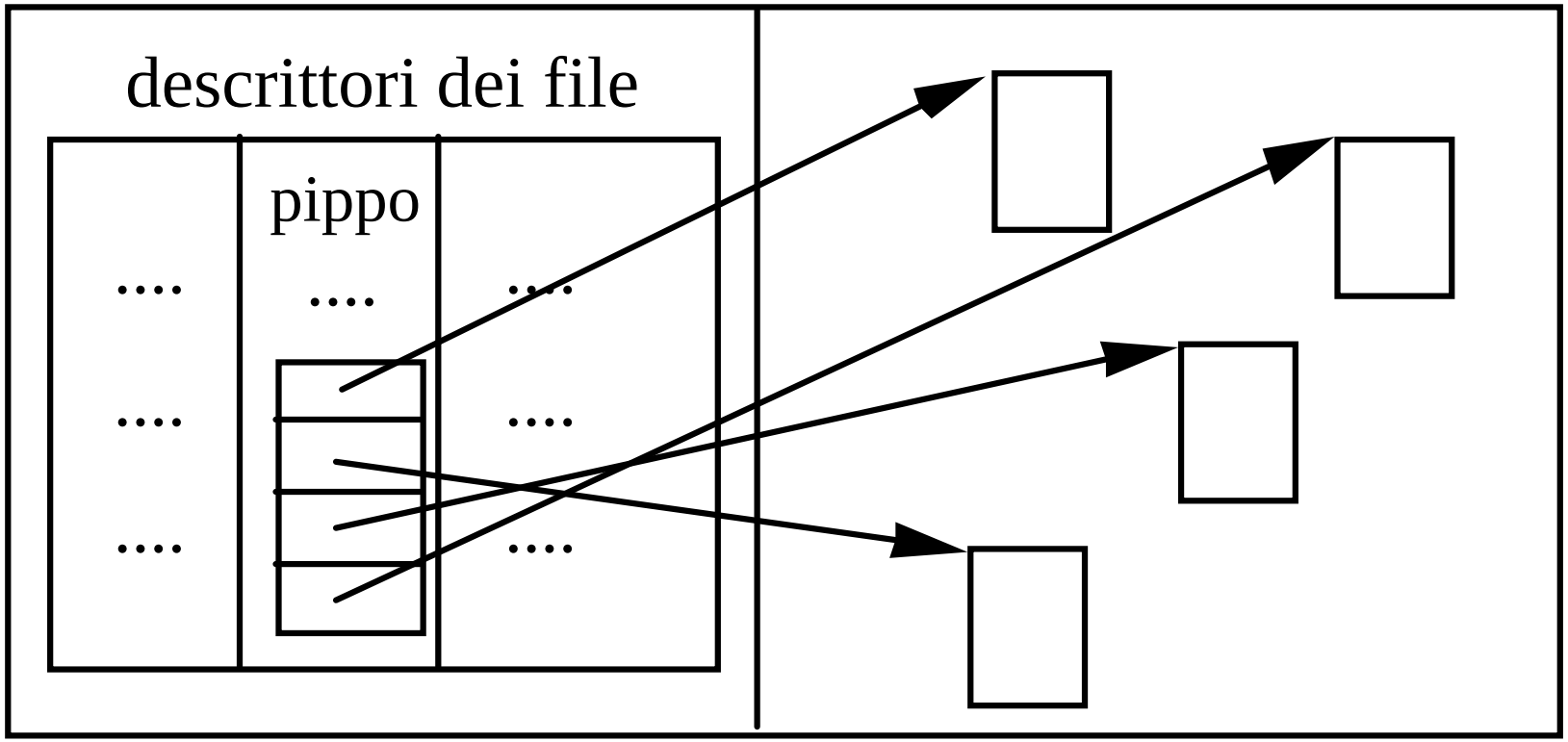
- non c'e` frammetazione esterna
- si evita la riallocazione dei file

Svantaggi:

- si ha un piccolo spreco in ogni blocco per via del puntatore al blocco successivo
- non e` possibile l'accesso diretto

ALLOCAZIONE INDEXATA

device directory spazio di memorizzazione dei file



ALLOCAZIONE INDEXATA (cont.)

- **Nel file descriptor viene riportato l'elenco di tutti i blocchi in cui e' memorizzato il file**

Vantaggi:

- non c'e` frammetazione esterna
- si evita la riallocazione dei file
- e` possibile l'accesso diretto

Svantaggi:

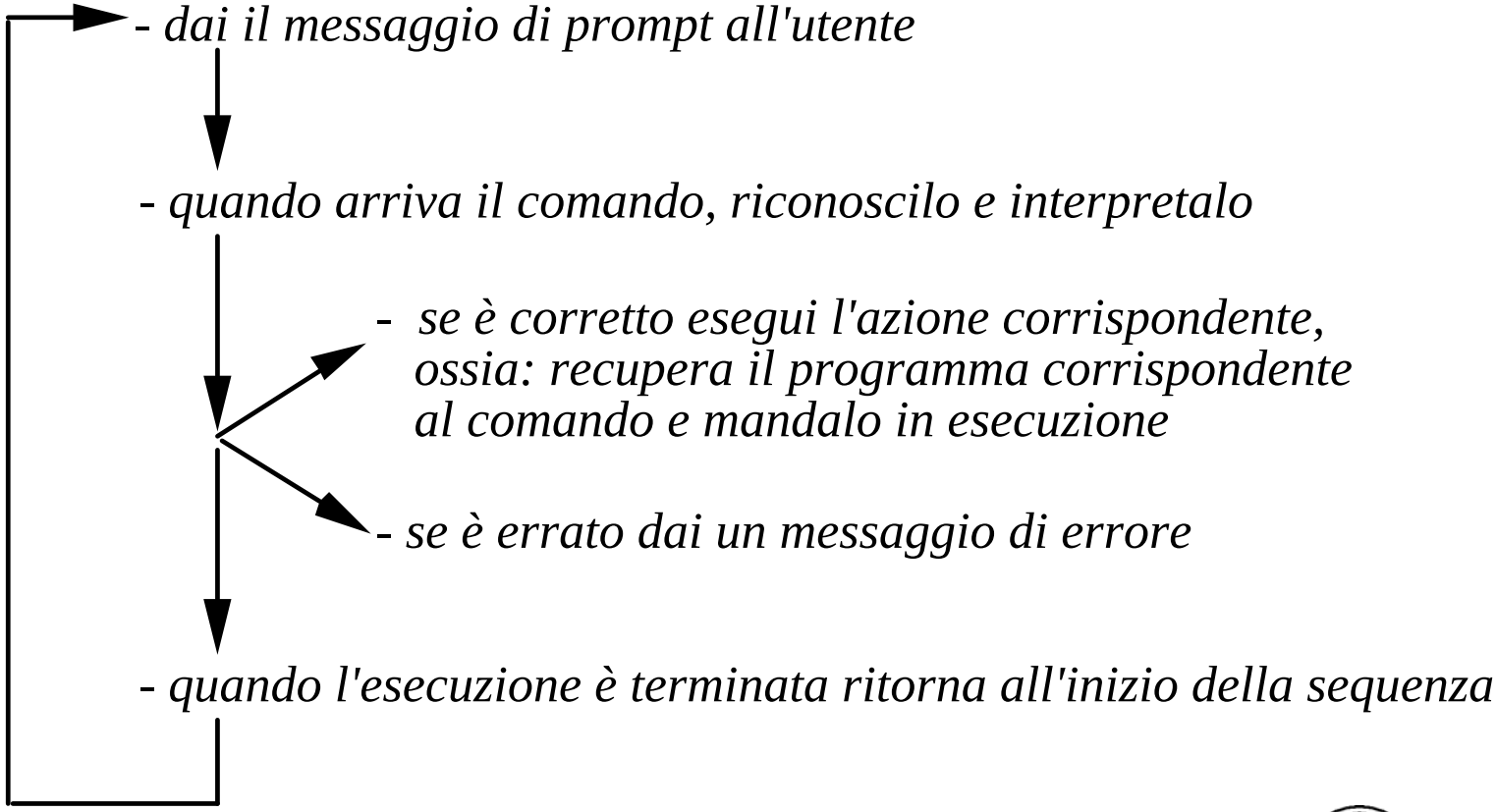
- difficolta` nella scelta del numero di indirizzi nel file descriptor

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Realizzazione dell'interprete dei comandi

<ripeti finché la macchina non viene spenta>



Realizzazione dell'interprete dei comandi

- **interfaccia testuale**

- **interfaccia grafica.**

Realizzazione dell'interprete dei comandi

- Il processo di sistema che realizza l'interprete dei comandi e la sua interfaccia viene mandato in esecuzione all'avvio della macchina e rimane sempre in attesa di comandi. Quindi per la maggior parte del tempo sarà bloccato in attesa di input dall'utente.
- All'arrivo di un comando il processo va brevemente in esecuzione per effettuare il riconoscimento del comando e, in caso esso sia corretto, reperire il programma corrispondente in memoria principale oppure su un supporto di memoria secondaria.

Realizzazione dell'interprete dei comandi

- Una volta trovato il programma da eseguire, si crea un nuovo processo cui viene demandato il compito di eseguirlo; il nuovo processo viene inserito nella coda dei processi pronti.
- Il processo di sistema che gestisce l'interazione si ferma e viene attivato il processo che esegue il comando dell'utente; questo processo gestirà l'interazione con l'utente.
- Al termine dell'esecuzione del comando il processo di sistema ritorna attivo, comunica all'utente che è pronto ad accettare un nuovo comando e si rimette in attesa.

Principali funzionalità

1. Avvio dell'elaboratore
2. Gestione del processore (e dei processi)
3. Gestione della memoria principale (e realizzazione della memoria virtuale)
4. Gestione dei dispositivi di input-output
5. Gestione della memoria di massa
6. Realizzazione dell'interprete dei comandi
7. Gestione della comunicazione in rete

Gestione della comunicazione in rete

- **Un calcolatore connesso ad una rete di calcolatori, per comunicare con gli altri calcolatori della rete, deve attenersi ad un insieme di regole (protocolli), come vedremo in seguito**

- **Vi sono alcuni moduli del sistema operativo che si occupano di realizzare l'interfaccia fra il calcolatore e la rete e, fra l'altro, di attuare i protocolli di reti necessari alla comunicazione**