

Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica
Sistemi operativi e reti
A.A. 2020-2021

Pietro Frasca

Lezione 4

Giovedì 15-10-2020

Principali componenti del SO

- Da quanto fino ad ora descritto, a grandi linee, possiamo distinguere i SO nelle seguenti classi principali, in base al tipo di applicazioni per cui sono stati progettati:
 - **Sistemi operativi batch** per applicazioni di calcolo intensivo (applicazioni CPU-bound) con l'obiettivo di ottimizzare l'efficienza d'uso delle risorse.
 - **Sistemi operativi time-sharing** per applicazioni interattive (I/O-bound) con l'obiettivo di minimizzare i tempi medi di risposta.
 - **Sistemi operativi real-time** per applicazioni di controllo, con l'obiettivo di rispettare tutti i loro vincoli temporali (deadline).
 - **Sistemi operativi per personal computer** per l'uso di un singolo utente al fine di massimizzare la semplicità d'uso e le prestazioni.
 - **Sistemi operativi distribuiti e di rete** per applicazioni distribuite con l'obiettivo di condividere risorse.

- Pur avendo architetture e obiettivi diversi, le suddette classi di SO hanno molte caratteristiche comuni.
- In particolare esamineremo le componenti di un SO che svolgono le seguenti funzioni fondamentali:
 - **Gestione dei processi.**
 - **Gestione della memoria.**
 - **Gestione dell'I/O**
 - **Gestione del file system**
 - **Protezione e sicurezza**
 - Comunicazione su rete

Gestione dei processi

- I sistemi operativi multiprogrammati consentono l'esecuzione di più programmi in modo concorrente.
- Un sistema è costituito da un insieme di programmi che condividono nel tempo l'uso del processore o di più processori nel caso di architetture multiprocessore.
- Un compito importante di un sistema operativo è pianificare e fare avanzare l'esecuzione dei programmi utente.
- Tuttavia, per via della sua elevata complessità, un sistema operativo è realizzato in un elevato numero di processi di sistema ciascuno dei quali fornisce ai programmi applicativi determinati servizi. Quando un computer è acceso, si avvia per primo il kernel, e anche se gli utenti del sistema non hanno avviato alcun programma, al termine del caricamento sono presenti decine o centinaia di processi. Si tratta di processi di sistema che insieme al kernel consentono al computer di funzionare.
- Tanto più è complesso un sistema operativo, tanto sono più numerosi i processi di sistema.

- I processi di sistema nei sistemi UNIX sono spesso indicati con il nome *daemon* mentre nei sistemi Windows si identificano con il termine *service* (*servizi*).

Concetto di processo

- I termini programma e processo sono spesso usati come sinonimi. Informalmente, finora, si è definito un processo come un programma in esecuzione.
- Il concetto di processo è legato alla multiprogrammazione. In un sistema multitasking molti programmi sono caricati in memoria pronti per essere eseguiti. La CPU passa rapidamente ad eseguire istruzioni di un programma a istruzioni di un altro programma. Quando la CPU sospende di eseguire le istruzioni di un programma, è necessario salvare il valore dei suoi registri, in modo che il programma interrotto possa, in seguito, riprendere la sua esecuzione come se l'interruzione non fosse avvenuta.

- Pertanto, un processo non si identifica soltanto con il codice di un programma ma comprende anche l'attività del processore rappresentata, istante per istante, dai valori contenuti nei suoi registri.
- Generalmente, un processo include anche una **sezione dati** per memorizzare le variabili globali e statiche del processo e lo **stack**, che contiene i dati temporanei, come ad esempio i parametri e le variabili locali di funzione. Può comprendere anche un **heap** nel caso in cui, durante l'esecuzione, il processo allochi memoria dinamicamente.
- Per tener traccia del ciclo di vita di un processo, tutte le sue proprietà sono mantenute in una struttura dati, il **PCB (Process Control Block)**.
- L'insieme di queste componenti è detta **immagine del processo**.

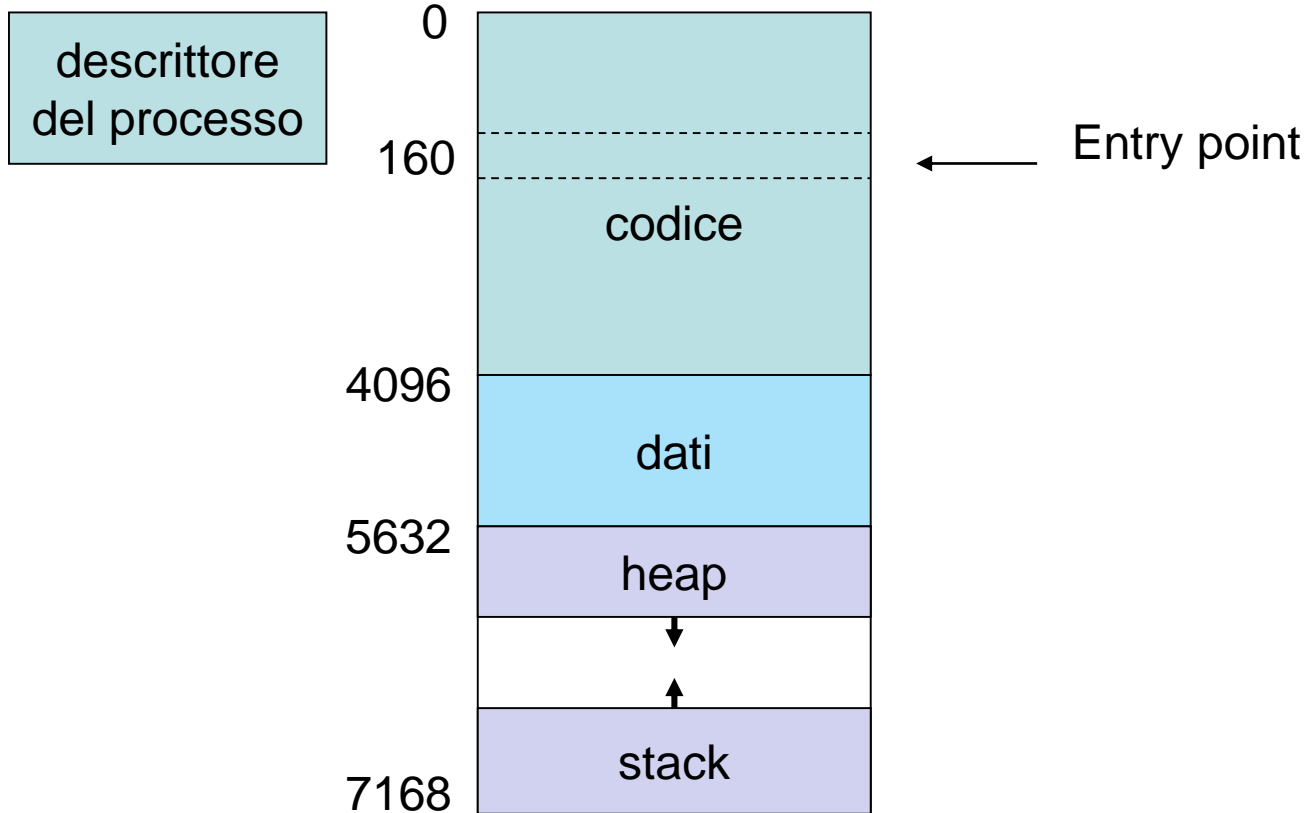


Immagine di un processo

Stati di un processo

- Un processo durante la sua esecuzione, esegue transizioni di stato che sono causate sia dall'esecuzione delle istruzioni del processo stesso, come ad esempio quando esso esegue operazioni di I/O, sia da eventi esterni asincroni con la sua esecuzione, come ad esempio la ricezione di segnali.
- Un processo entra nello **stato nuovo** quando è creato, ad esempio quando si avvia un programma.
- Un processo passa dallo stato nuovo allo **stato di pronto** dopo che il SO ha verificato che esso può essere effettivamente eseguito (ad esempio il processo ha i diritti di esecuzione).
- Un processo passa dallo stato di **pronto** allo stato di **esecuzione** quando ad esso è assegnato il processore.
- Un processo è nello **stato attivo** quando si trova nello stato di **pronto** o di **esecuzione**.
- Un processo passa dallo **stato di esecuzione** allo **stato di bloccato** quando è in attesa di qualche evento.

- Il processo ritorna nello stato di pronto quando l'evento atteso si è verificato.
- La transizione dallo stato di esecuzione allo stato di pronto è chiamato **prerilascio** (preemption) o **revoca**.
- Il prerilascio può avvenire per vari motivi, come ad esempio, quando un processo ha esaurito il suo quanto di tempo (nei sistemi time-sharing), o è presente nella coda di pronto un altro processo con priorità più alta.
- Il passaggio dallo stato di pronto allo stato di esecuzione è gestito dallo scheduler, un componente del kernel che seleziona un processo cui assegnare la CPU, tra tutti i processi che si trovano nello stato di pronto. Lo **scheduler** ha il compito di garantire che tutti i processi pronti possano avanzare nella loro esecuzione.
- Il processo passa nello stato **terminato** quando ha terminato l'esecuzione del suo programma o quando si è verificato un'eccezione di vario tipo.

Transizioni di stato

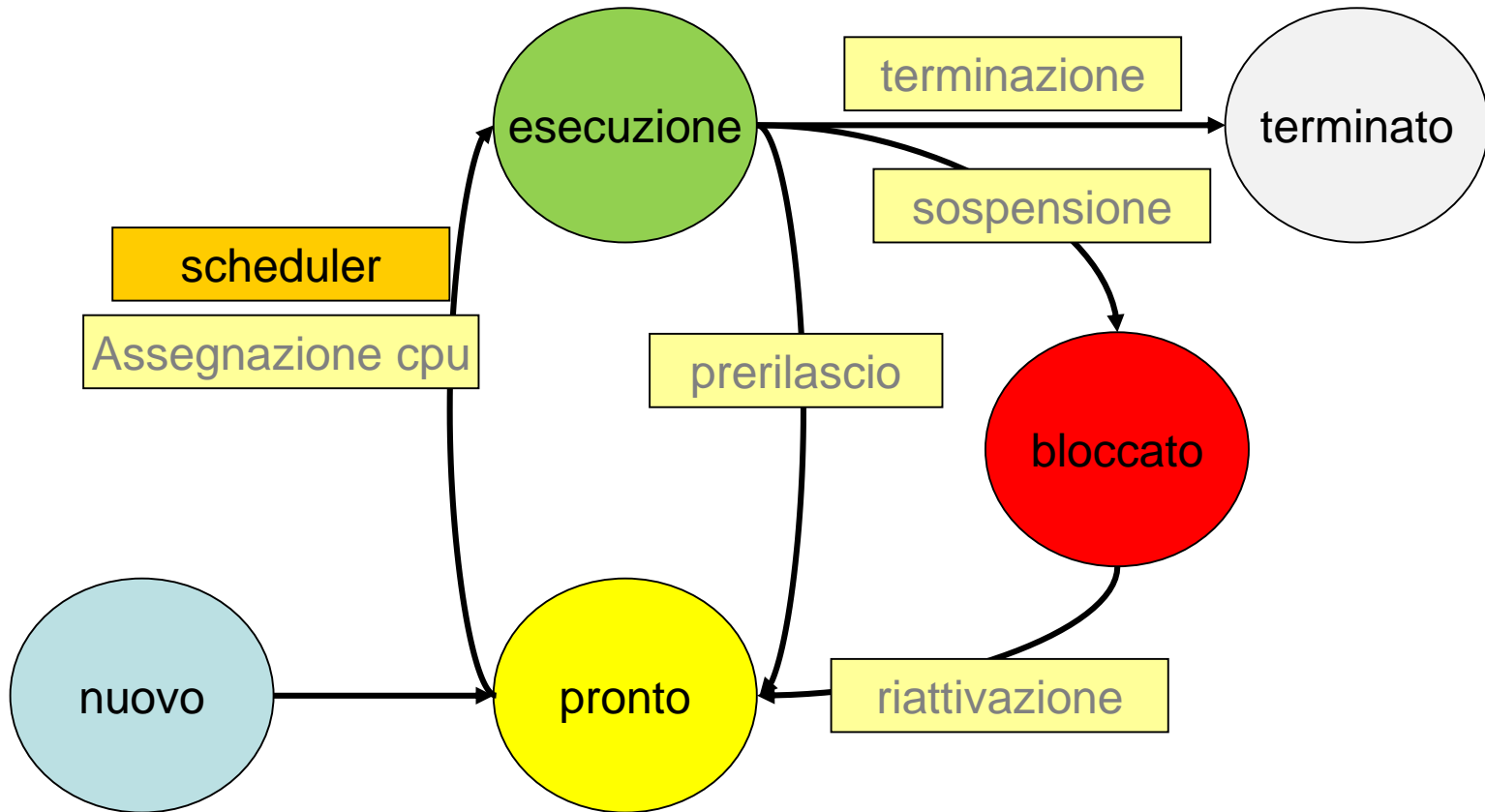


Diagramma di transizione a 5 stati

- In molti sistemi è previsto che un processo sia trasferito dalla memoria principale alla memoria secondaria (disco) in modo da ottenere spazio per altri processi. Tale operazione prende il nome di **swapping (scambio)** e lo stato relativo è detto **swapped**.
- Vedremo un diagramma di transizione di stato con lo stato swapped quando parleremo della gestione della memoria.

Descrittore del processo (PCB)

- Ogni processo è rappresentato nel sistema da una struttura dati, detta **descrittore del processo (PCB Process Control Block)**.
- I descrittori dei processi sono memorizzati in una tabella (lista concatenata), detta **tabella dei processi**.
- I campi presenti nel descrittore di processo dipendono dal particolare sistema operativo e dall'architettura dell'hardware.
- Generalmente il descrittore del processo contiene le seguenti informazioni:
 - **Identificatore del processo**. Spesso si identifica il processo con un numero intero detto **PID**.
 - **Stato del processo**. Questo campo identifica lo stato in cui si trova il processo in un determinato istante. Può anche non essere presente nel descrittore ma può essere ottenuto implicitamente dall'appartenenza del descrittore ad una delle code gestite dal kernel.

- **Informazioni sullo scheduling di CPU.** La scelta del processo pronto a cui assegnare la CPU può essere effettuata secondo diversi criteri.
 - **FIFO.** Il criterio più semplice è il FIFO (*First-In-First-Out*), che prevede di assegnare la CPU al processo pronto in attesa da più tempo.
 - **Priorità.** Ad ogni processo è assegnata una priorità, fissa o dinamica, che indica la sua importanza relativa nei confronti degli altri processi.
 - **Deadline (scadenza).** La scelta del prossimo processo può essere basata anche in termini di intervallo di tempo (**deadline**) in cui l'esecuzione del processo deve essere portata a termine. Nel descrittore del processo, in questo caso, è contenuto un valore che, sommato all'istante della richiesta di servizio da parte del processo, determina il tempo massimo entro il quale la richiesta deve essere soddisfatta (*sistemi in tempo reale*).

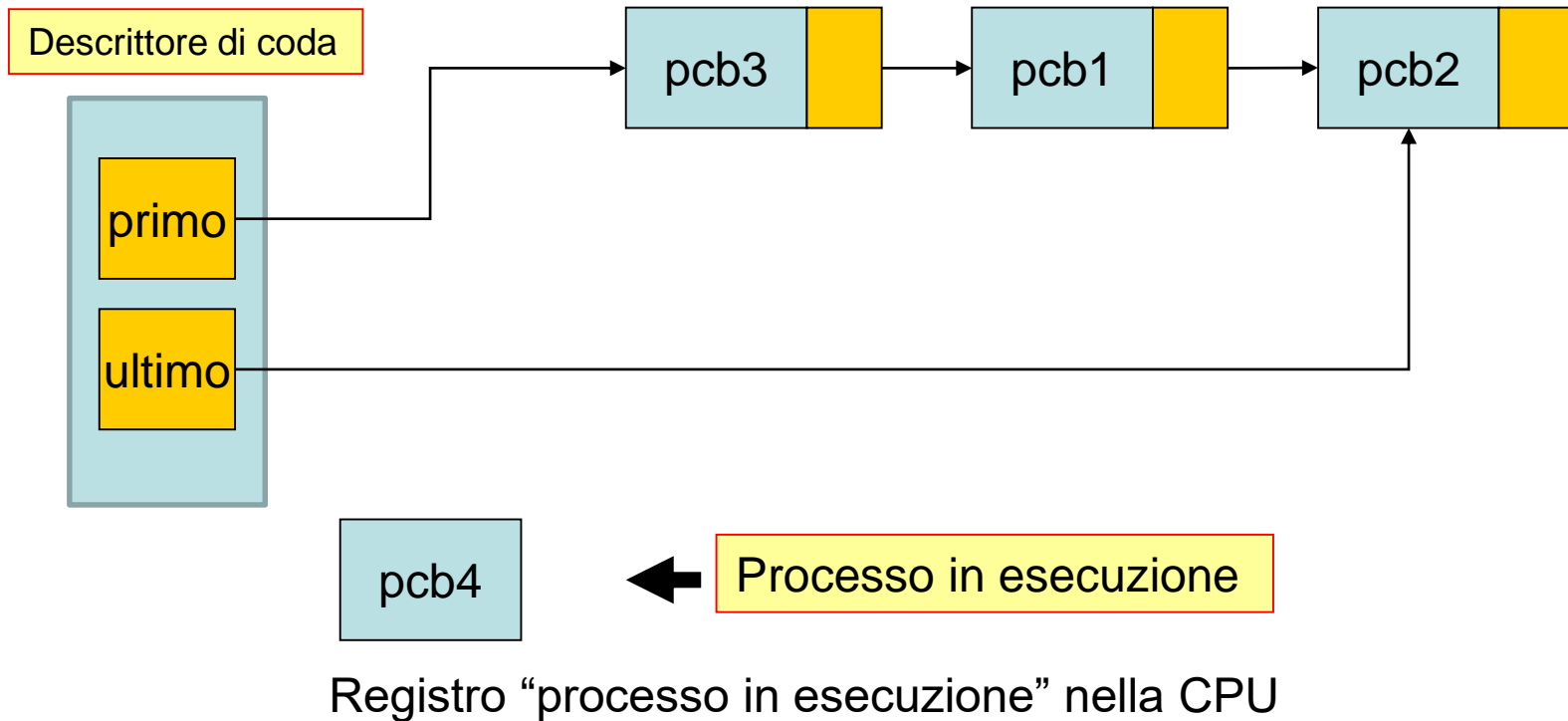
- **Quanto di tempo.** Nei sistemi time-sharing, nel descrittore è contenuto un valore che rappresenta l'intervallo di tempo (quanto) consecutivo in cui la CPU è assegnata allo stesso processo.
- **Informazioni sulla gestione della memoria.** Contiene le informazioni relative all'area di memoria principale nella quale sono caricati il codice, i dati e lo stack del processo. Queste informazioni dipendono dalla particolare tecnica di gestione della memoria usata dal sistema operativo.
- **Contesto del processo.** L'insieme dei valori dei registri del processore al momento della sospensione dell'esecuzione di un processo è salvato nel suo descrittore. Questo insieme di valori prende il nome di **contesto del processo** e dipende dall'architettura del processore. Tipici registri presenti in una CPU sono il *PSW (Program Status word)*, lo *SP (puntatore allo stack - stack pointer)*, registri indice, accumulatori, registri di uso generale. Il contesto è recuperato dal descrittore e riportato ai registri quando il processo torna in esecuzione.

- **Utilizzo delle risorse.** Queste informazioni comprendono la lista dei dispositivi di I/O allocati al processo, i file aperti, il tempo di uso della CPU ecc.
- **Identificatore del processo successivo.** Come si è detto, a seconda del loro stato (pronto o bloccato), i processi sono inseriti, in apposite code. Ogni descrittore contiene pertanto un puntatore al processo successivo (e spesso anche un puntatore al precedente) nella stessa coda.
- **Informazioni sulla sicurezza**
- **Informazioni sulle variabili di ambiente**
- **Informazione utente**
- Per la loro fondamentale importanza i descrittori sono memorizzati in un'area di memoria accessibile solo dal kernel.

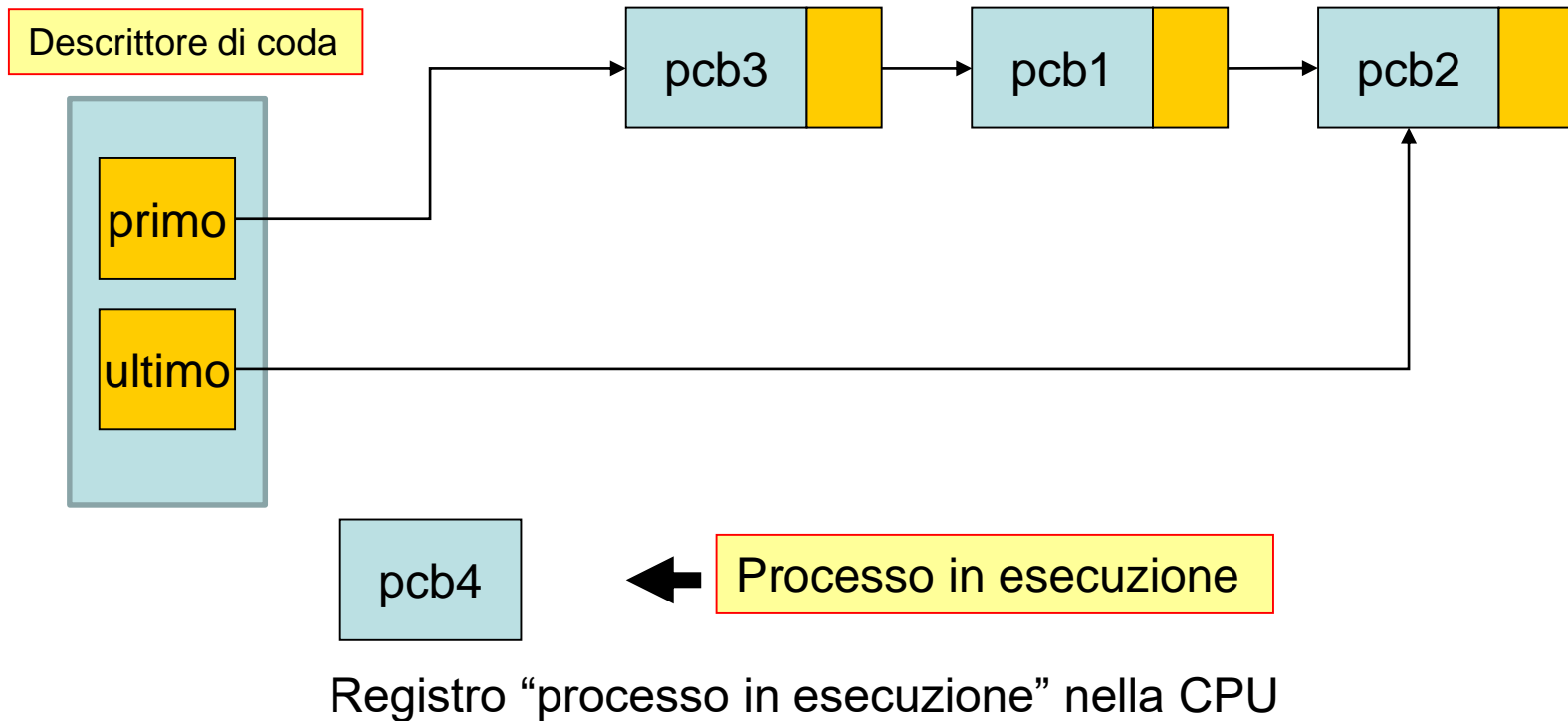
Code di processi

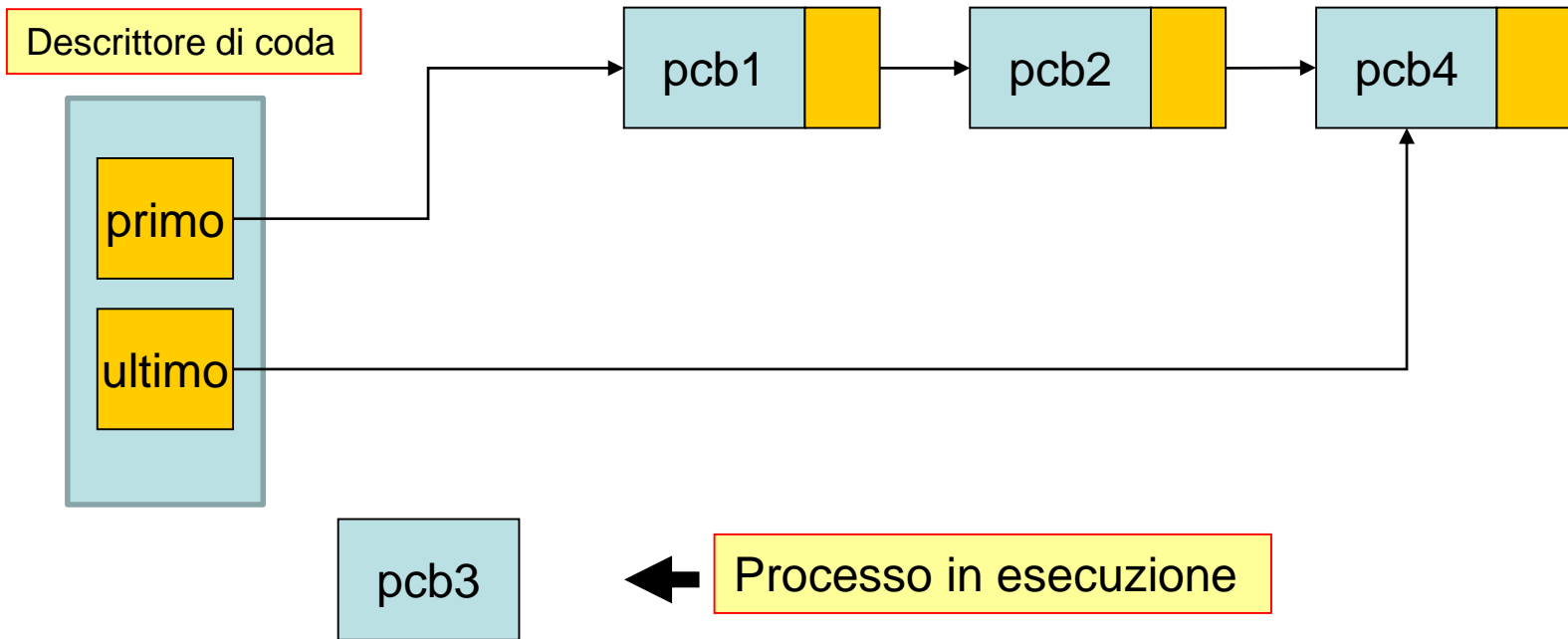
- Il diagramma di transizione degli stati, fornisce una visione astratta del funzionamento dei processi quando questi compiono transizioni di stato. A livello d'implementazione, ciascuno stato è realizzato con una o più code.
- Assumendo che il computer abbia una sola CPU, in ogni istante un solo processo può trovarsi nello stato di esecuzione. Gli altri processi possono essere o nello stato di pronto o nello stato di bloccato in attesa che si verifichi un determinato evento che li riporti nello stato di pronto.
- Generalmente, le CPU possiedono un registro detto **registro del processo in esecuzione** nel quale il sistema operativo memorizza il puntatore al descrittore del processo che è in esecuzione.
- I processi presenti nella memoria principale che sono pronti per essere eseguiti sono organizzati in una o più code dette **code dei processi pronti**.

- Ad ogni coda è associato un **descrittore di coda** che contiene due campi che specificano rispettivamente l'indice del primo e dell'ultimo descrittore di processo in coda.
- Ogni descrittore di processo ha un campo per indicare il processo successivo contenuto nella coda dei processi pronti.



- L'operazione di **inserimento** di un processo in una coda dei processi pronti è dovuta alla transizione nello stato di pronto di un processo dallo stato nuovo, bloccato o in esecuzione.
- L'operazione di prelievo corrisponde al passaggio dallo stato di pronto allo stato di esecuzione.

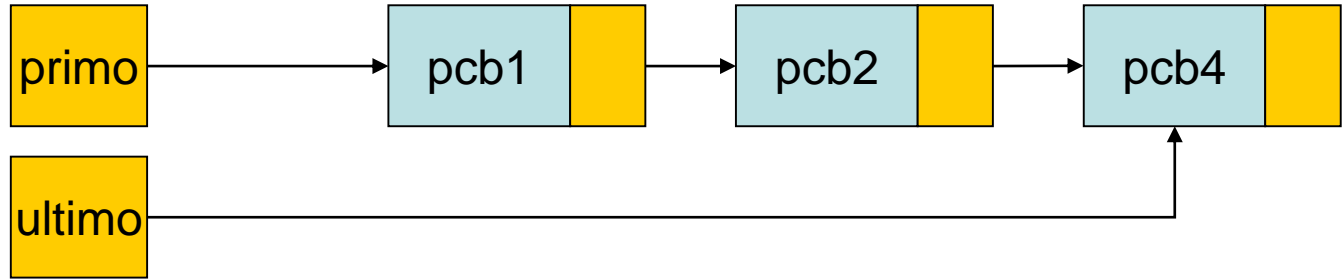




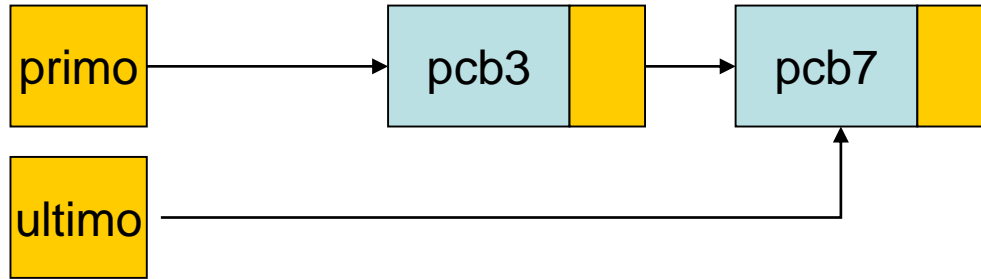
Registro "processo in esecuzione" nella CPU

- Nella coda dei processi pronti è sempre presente un particolare processo: il **processo di inattività (idle)**, che va in esecuzione quando tutti i processi sono bloccati.
- Il processo *idle* resta sempre nella coda di pronto e ha sempre la priorità più bassa. Rimane in esecuzione fino a quando qualche altro processo entra nella coda di pronto.
- Esistono inoltre **code per i processi bloccati**, una per ogni evento o condizione per cui i processi possono attendere. Ad esempio, esistono code per ogni dispositivo di I/O nelle quali sono inseriti i descrittori dei processi in attesa della conclusione di una richiesta di I/O su quel determinato dispositivo.
- L'operazione di creazione di un processo implica la creazione di un nuovo descrittore e l'operazione di terminazione l'eliminazione del descrittore.

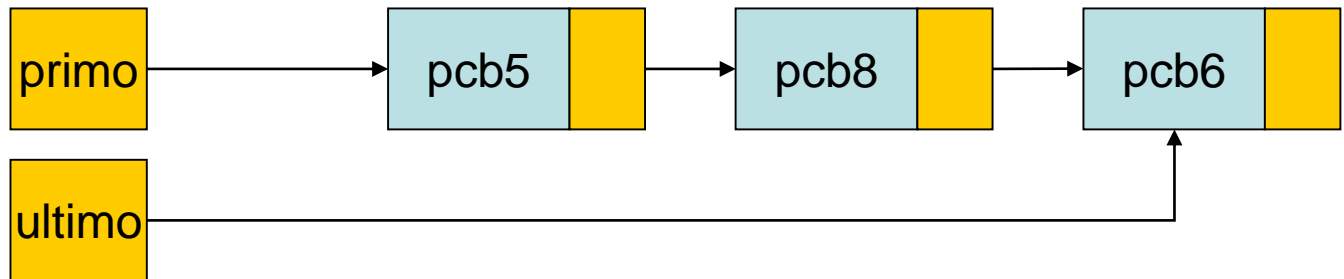
Dispositivo 1



Dispositivo 2

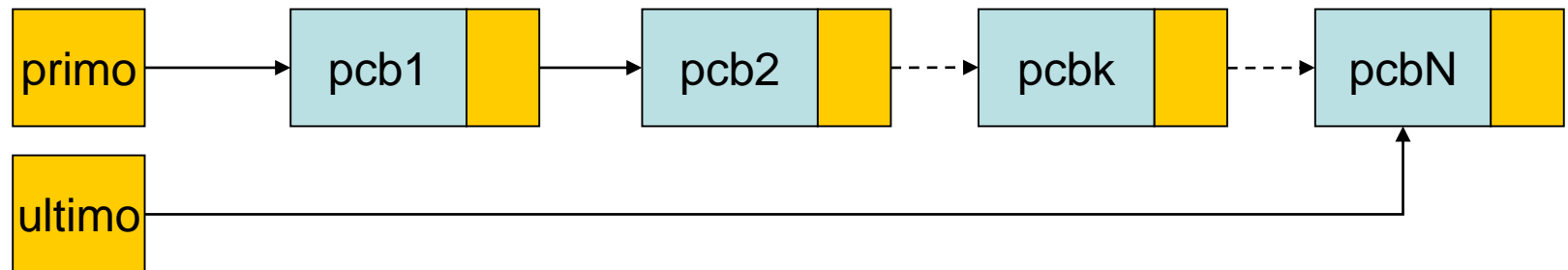


Dispositivo N



Code per i processi bloccati

- In alcuni sistemi può essere presente una coda nella quale sono presenti un numero fisso di **descrittori disponibili** per la creazione di nuovi processi. In tal caso il sistema avrà un numero massimo di processi predefinito. L'operazione di creazione estrae da questa coda il descrittore del processo da creare che sarà successivamente inserito nella coda dei processi pronti. Viceversa l'operazione di eliminazione di un processo riporta nella coda dei descrittori disponibili il descrittore del processo eliminato.



Coda di descrittori disponibili

Scheduler

- Un processo, durante il suo ciclo di vita, transita da una coda all'altra. Il sistema operativo, per scopi di pianificazione, deve elaborare queste code secondo qualche strategia.
- Generalmente, per la pianificazione dei processi sono previsti più livelli di scheduler.
- Tipicamente, nei sistemi batch, più programmi sono memorizzati in una particolare parte del file system, su disco, detta *area di spool*, per l'esecuzione. Lo scheduler a *lungo termine*, o *job scheduler*, seleziona i programmi da quest'area e li carica in memoria per l'esecuzione.
- Lo ***scheduler a breve termine***, o ***CPU scheduler***, seleziona dalla coda di pronto il prossimo processo cui assegnare la CPU. La differenza principale tra questi due scheduler sta nella frequenza di esecuzione. Lo scheduler a breve termine deve scegliere frequentemente un nuovo processo per la CPU.

- Generalmente, lo scheduler a breve termine è eseguito ogni 50-100 millisecondi. A causa del breve tempo tra le esecuzioni, lo scheduler a breve termine deve essere veloce.
- Se, ad esempio, lo scheduler impiegasse 10 millisecondi per scegliere il prossimo processo che resta in esecuzione per 50 millisecondi, allora, sarebbe utilizzato circa il $10 / (50 + 10) = 0.166$, cioè circa il 17 per cento della CPU soltanto per selezionare il prossimo processo cui assegnare il processore.
- Lo scheduler a lungo termine, tipicamente va in esecuzione molto meno frequentemente, con un periodo di qualche minuto.
- È importante che lo scheduler a lungo termine effettui un'attenta selezione.
- In generale, i processi possono essere classificati come I/O bound o CPU-bound. Un processo è detto I/O-bound quando esegue prevalentemente operazioni di I/O piuttosto che di calcolo. Un processo CPU-bound, al contrario, genera rare richieste di I/O, utilizzando più del suo tempo per operazioni di computazione. E

- E' importante che uno scheduler a lungo termine selezioni una combinazione di processi I/O-bound e CPU-bound in modo da bilanciare il carico di lavoro tra la CPU e i dispositivi di I/O.
- Se tutti i processi sono I/O bound, la coda di pronto sarà spesso vuota, e la CPU sarà poco impegnata. Se tutti i processi sono CPU-bound, le code di attesa di I/O saranno spesso vuote, i dispositivi non saranno utilizzati, e di nuovo il sistema sarà sbilanciato. Pertanto, il sistema con le migliori prestazioni avrà una giusta combinazione di processi I/O-bound e CPU-bound.
- In alcuni sistemi, lo scheduler a lungo termine può essere assente o minimo. Ad esempio, i sistemi Linux e Microsoft Windows non hanno scheduler a lungo termine.
- Alcuni sistemi operativi, come ad esempio i time-sharing, possono avere un altro livello intermedio di pianificazione detto scheduler a *medio termine*. L'idea di aggiungere uno scheduler a medio termine è che a volte può essere vantaggioso rimuovere un processo dalla memoria riducendo così il grado di multiprogrammazione.

- In seguito, il processo può essere riportato in memoria, e la sua esecuzione può essere ripresa da dove era stata interrotta. Questo schema si chiama *swapping* (*scambio*). Lo scambio può essere necessario per migliorare la combinazione di processi CPU-bound e I/O-bound o perché la memoria disponibile è in esaurimento, e quindi deve essere liberata.

Cambio di contesto

- Il cambio del processo che è in esecuzione con un altro, scelto dalla coda dei processi pronti, si realizza mediante un insieme di operazioni eseguite dal kernel che prende il nome di **cambio di contesto**. Tali operazioni sono:
 1. Salvataggio del contesto del processo in esecuzione nel suo PCB (**salvataggio dello stato**). il contesto dipende dall'architettura del processore.
 2. Inserimento del PCB nella coda dei processi pronti o dei processi bloccati. L'operazione di inserimento del descrittore nella coda dei processi pronti si ha se il cambio di contesto è dovuto ad una revoca mentre l'inserimento in una coda di bloccato si ha se il cambio di contesto è dovuto alla sospensione del processo.
 3. Selezione di un altro processo dalla coda dei processi pronti e caricamento del PCB di tale processo nel registro processo in esecuzione.

Questa operazione è eseguita dallo scheduler a breve termine, secondo una determinata strategia la quale può essere basata su un algoritmo di tipo FIFO nel caso esista una sola coda per tutti i processi pronti o utilizzando l'informazione relativa alla priorità dei processi nel caso di più code di processi (una per livello di priorità).

4. Caricamento del contesto del nuovo processo nei registri del processore (***ripristino dello stato***).
 5. Aggiornamento di tutte le strutture dati che rappresentano le risorse utilizzate dai processi: memoria, dispositivi di I/O, file aperti ecc.
- Le operazioni di *salvataggio dello stato* e *ripristino dello stato*, comportano il trasferimento di dati dai registri del processore alla memoria centrale e viceversa.

- I tempi di context switch variano da macchina a macchina e sono altamente dipendenti dal supporto hardware. Ad esempio, alcuni processori hanno più set di registri. Un cambio di contesto in questo caso si ottiene semplicemente cambiando il set dei registri corrente.
- Naturalmente, se il numero di processi attivi è superiore al numero di set di registri, è necessario salvare la copia dei registri da e verso il PCB, come descritto prima.
- Il tempo di commutazione dipende anche dall'esistenza di un supporto hardware che consente di caricare o memorizzare tutti i registri con una singola istruzione, dal numero di registri che deve essere copiato e dalla velocità di accesso della memoria principale.
- Il tempo necessario per eseguire il *context switch* è tipicamente di alcuni millisecondi. È un overhead, perché durante la commutazione la CPU non esegue i processi utente o di commutazione.

- Inoltre, più è complesso il sistema operativo, maggiore è il tempo necessario per il cambio di contesto. In particolare, sistemi con tecniche di gestione di memoria avanzate richiedono che, ad ogni cambio di contesto, siano salvati molti altri dati in più nel PCB.

