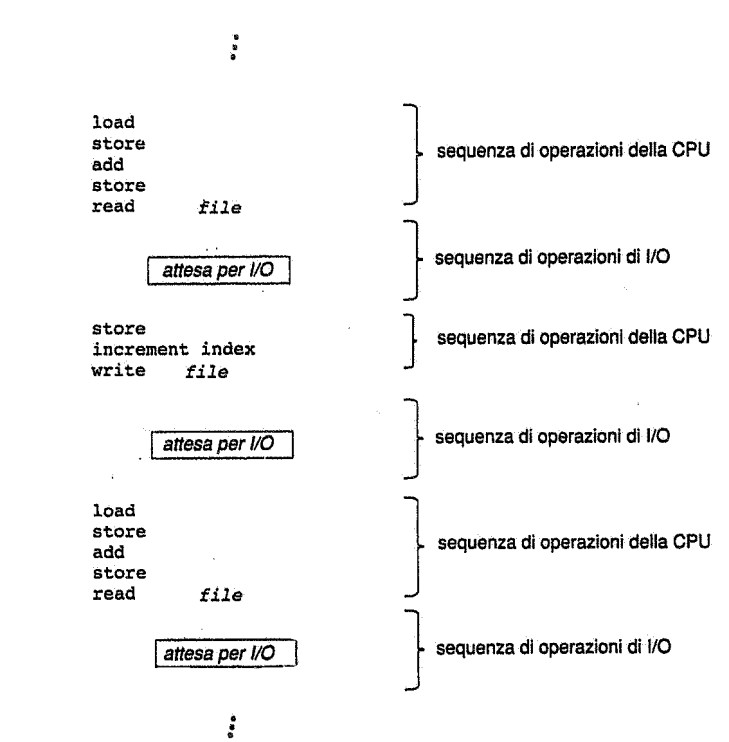
# CAPITOLO 5: SCHEDULING DELLA CPU

## Con la multi-programmazione si cerca di impiegare i tempi di attesa in maniera produttiva : si tengono più processi in memoria contemporaneamente e quando un processo deve attendere un evento , il SO gli sottrae il controllo della CPU per cederla a un altro processo.

Lo scheduling è una funzione fondamentale dei SO.

## Ciclicità delle fasi d’elaborazione e di IO



L’esecuzione di un processo consiste in un ciclo di elaborazione ed un ciclo di completamento. I processi si alternano in questi due cicli fino al loro completamento. Comincia con una sequenza effettuata dalla CPU (CPU burst), seguita da una sequenza di operazioni di IO (IO burst) e cosi via fino a quando non viene inviata una richiesta di terminazione.

Un programma con prevalenza di IO(IO bound) produce molta sequenza di operazioni della CPU a breve durata. Un programma con prevalenza di operazioni di elaborazione(CPU bound), produce poche sequenze di operazioni della CPU ma molto lunghe. Quindi deve sussistere un buon algoritmo per la scelta di queste operazioni.

## Scheduler della CPU

Ogniqualvolta la CPU passa nello stato d’inattività, il sistema operativo sceglie per l’esecuzione uno dei processi presenti nella coda dei processi pronti. In particolare lo scheduler a breve termine che sceglie tra i processi presenti in memoria ad assegnarli alla CPU.

**Scheduling con diritto di prelazione**

Le decisioni riguardanti lo scheduling della CPU si possono prendere nelle seguenti circostanze:

* Un processo passa dallo stato di esecuzione in uno stato di attesa (x es.:richiesta di I/O o richiesta di attesa(WAIT)per la terminazione di uno dei processi figli) NB : senza diritto di prelazione
* Un processo passa dallo stato di esecuzione allo stato di pronto (x es.:quando si verifica un segnale di interruzione) NB : con diritto di prelazione
* Un processo passa dallo stato di attesa allo stato di pronto (x es.:completamento di un'operazione di I/O) NB : con diritto di prelazione
* Un processo termina NB : senza diritto di prelazione

I casi 1 e 4 non comportano alcuna scelta di scheduling; a essi segue la scelta di un nuovo processo da eseguire , sempre che ce ne sia uno pronto.

Una scelta di scheduling si deve fare nei casi 2 e 3.

Nel caso dello scheduling **senza diritto di prelazione** , quando si assegna la CPU a un processo , questo rimane in possesso della CPU fino al momento del suo rilascio , dovuto al termine dell'esecuzione o al passaggio nello stato di attesa.

Lo scheduling con **diritto di prelazione** presenta un incoveniente: si consideri il caso in cui 2 processi condividono dati ; mentre uno di questi aggiorna i dati , si ha la sua prelazione in favore dell'esecuzione dell'altro. Il secondo processo in questo caso può tentare di leggere i dati che sono stati lasciati in uno stato incoerente dal primo processo.

## Dispatcher

Il dispatcher è il modulo che passa effettivamente il controllo della CPU ai processi scelti dallo scheduler a breve termine.

Siccome il dispatcher si attiva ad ogni cambio di contesto, esso deve essere il più rapido possibile.

Il tempo richiesto dal dispatcher per fermare un processo e avviare l'esecuzione di un altro è nota come **latenza di dispatch.**

## 2.CRITERI DI SCHEDULING

Esistono diversi algoritmi di scheduling della CPU e hanno proprietà differenti e possono favorire una determinata classe di processi. Nella scelta dell’algoritmo di scheduling bisogna tener conto:

Utilizzo della cpu: la CPU deve restare il più a lungo possibile in attività.

* Produttività: la CPU è attiva quando svolge del lavoro. Una misura del lavoro svolto è data dal numero dei processi completati nell’unita di tempo(THROUGHPUT)
* Tempo di completamento: ossia considerare il tempo che intercorre tra la sottomissione del processo e il completamento della sua esecuzione
* Tempo di attesa: l’algoritmo di scheduling della CPU non influisce sul tempo impiegato per l’esecuzione di un processo o di un’operazione di IO. Il tempo di attesa è la somma degli intervalli d’attesa passati nella coda dei processi pronti.
* Tempo di risposta: il tempo di risposta è il tempo che intercorre tra la sottomissione di una richiesta e la prima risposta prodotta.

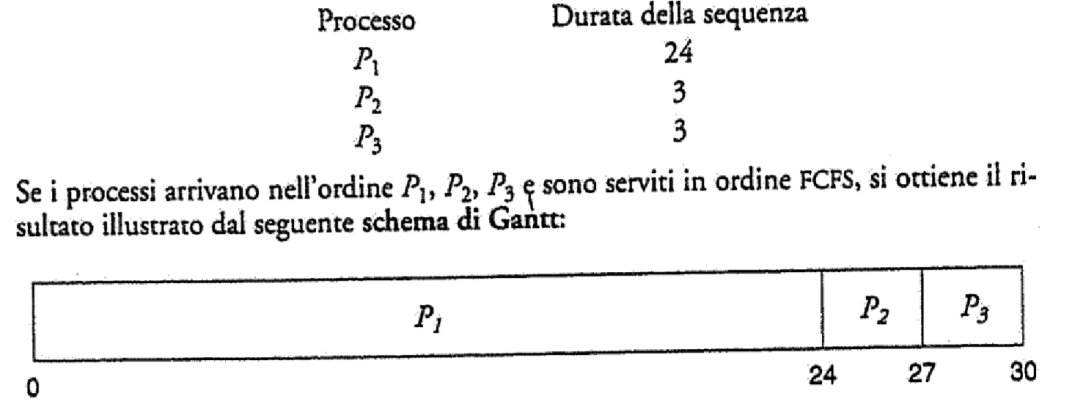
## 3.ALGORITMI DI SCHEDULING

Lo scheduling della CPU riguarda la scelta dei processi pronti cui assegnare la CPU

## Scheduling in ordine di arrivo FCFS

Il più semplice algoritmo di scheduling della CPU è quello della scelta dei processi in base all’ordine di arrivo. Tale algoritmo noto anche come FCFS sceglie il processo in testa alla coda FIFO. Quando un processo entra nella coda dei processi pronti, si collega il suo PC all’ultimo elemento della coda. Quando la CPU è libera si assegna al processo che si trova in testa alla coda. È spesso molto lungo.

Un problema che si può verificare con la scelta di tale algoritmo è l’effetto del convoglio: tutti i processi attendono che un lungo processo liberi la CPU mentre nella coda ci sono processi a breve termine che attendono un lungo periodo. È senza prelazione, una volta che la CPU viene assegna ad un processo, questo lo trattiene fino al suo rilascio.

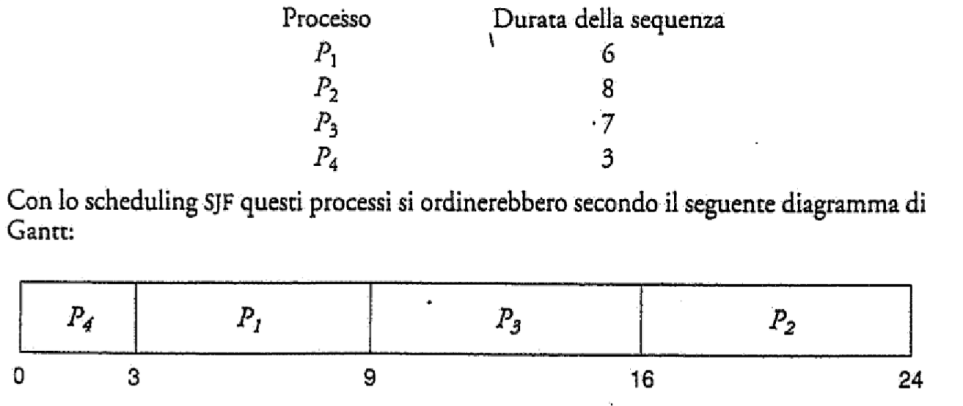


Il tempo di attesa medio è di 17 millisecondi (0+24+27)/3=17

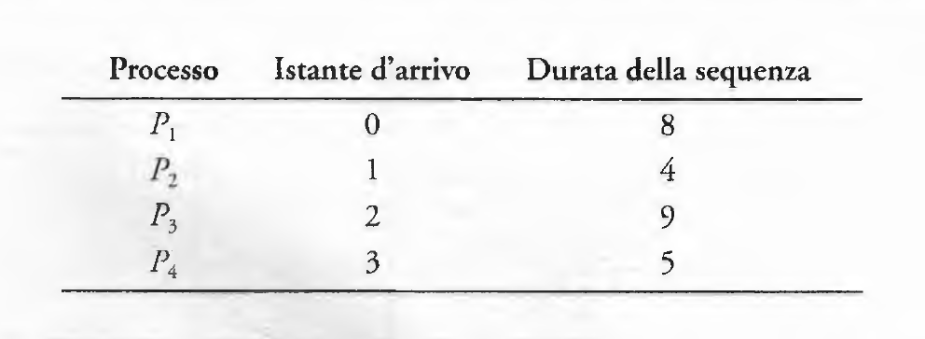
## 

## Scheduling per brevità SJF

Un criterio diverso è quello dello scheduling della CPU per brevità noto anche come SJF. Questo algoritmo associa per ogni processo la lunghezza della successiva sequenza di operazioni. Ogni volta si assegna la CPU al processo che ha la più breve sequenza. Se 2 processi hanno la stessa sequenza allora in questo caso si applica l’algoritmo FCFS.



In questo caso il tempo di attesa è (3+16+9+0)/4=7.

Si può dimostrare che è ottimale ma la difficoltà reale implica nel conoscere la durata della successiva richiesta di CPU. Tale algoritmo può essere sia con diritto di prelazione che senza. Nel secondo caso l’algoritmo continua la sua esecuzione così come è stato presentato, mentre nel primo caso (con diritto di prelazione) se durante l’esecuzione di un processo arriva un nuovo processo in cui la sua sequenza è minore della sequenza restante del processo in esecuzione, allora tale processo viene posto nella coda dei processi pronti e quello appena arrivato viene assegnato alla CPU.



All’istante 0 si avvia il processo *P1,* poiche e l’unico che si trova nella coda. All’istante 1 arriva

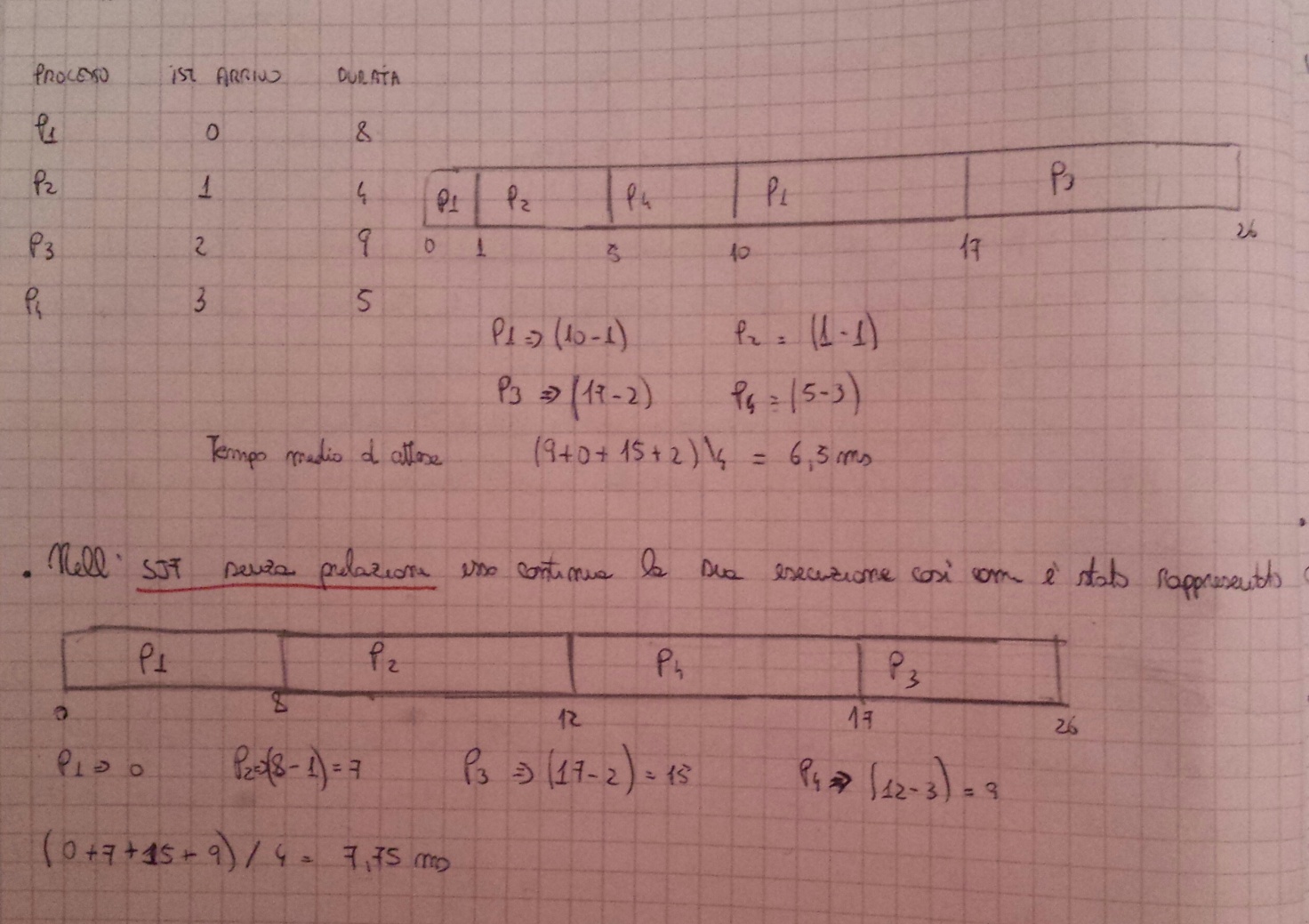
il processo *P2.* Il tempo necessario per completare il processo *P1* (7 millisecondi) è maggiore

del tempo richiesto dal processo *P2* (4 millisecondi), perciò si ha la prelazione sul processo

*P1* sostituendolo col processo *P2.* Il tempo d’attesa medio per questo esempio e

((10 - 1) + (1 - 1) + (17 - 2) + (5 - 3))/4= 26/4 = 6,5 millisecondi.

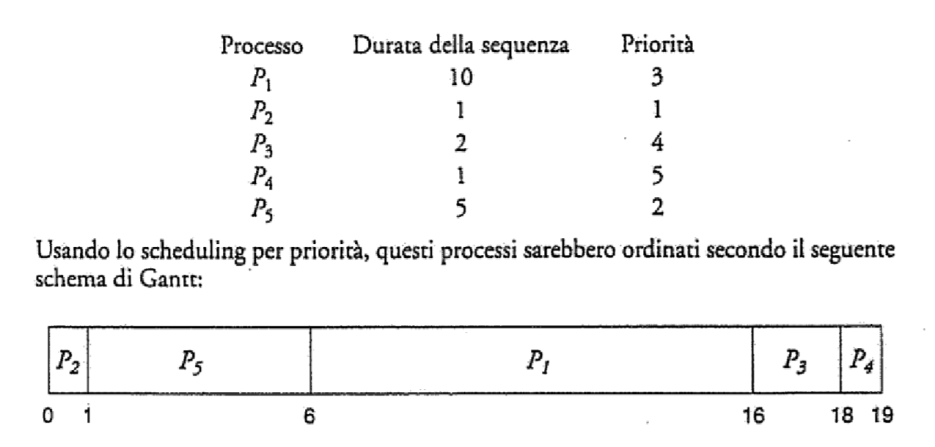
Con uno scheduling SJF senza prelazione si otterrebbe un tempo d’attesa medio di 7,75 millisecondi.



## Scheduling per priorità

L’algoritmo SJF è un caso particolare del più generale algoritmo di scheduling per priorità. Tale algoritmo oltre a presentare la durata della sequenza, si associa una priorità a ogni processo e si assegna la CPU al processo con priorità più alta.

Un algoritmo SJF non è altro che un algoritmo con priorità dove tale priorità è l’inverso della lunghezza.



Il tempo medio di attesa è di 8.2 millisecondi.

L’algoritmo per priorità è sia con diritto che senza diritto di prelazione.

L'algoritmo con **diritto di prelazione** sottrae la CPU al processo attualmente in esecuzione se la priorità dell'ultimo processo arrivato è superiore.

L'algoritmo **senza diritto di prelazione** si limita a porre l'ultimo processo arrivatoalla testa della coda dei processi pronti.

Un problema importante è il problema della **starvation** (attesa indefinita). Un processo pronto per l’esecuzione che non dispone della CPU si può considerare bloccato nell’attesa nel caso questo abbia una priorità bassa e non venga mai eseguito.("Un processo con priorità maggiore può impedire a un processo con priorità minore di accedere alla CPU") .

Una soluzione a questo problema è costituita dall’invecchiamento. Ogni qualvolta passa un periodo di tempo un processo che sta nell’attesa della CPU ottiene un aumento graduale della priorità facendo in modo che venga prima o poi eseguito.

## 

## Scheduling circolare Round Robin

L’algoritmo circolare noto anche come Round Robin è stato progettato appositamente per i sistemi a partizione del tempo.

E' simile allo scheduling FCFS ma in più ha la capacità di prelazione per la commutazione tra processi .

Ciascun processo riceve una piccola quantità fissata del tempo della CPU , chiamata quanto di tempo(Time Slice); la coda dei processi pronti è trattata come una coda circolare.

Lo scheduler della CPU scorre la coda dei processi pronti , assegnando la CPU a ciascun processo per una durata massima pari al quanto di tempo (varia di solito tra 10-100 ms).

La coda dei processi pronti viene gestita come una coda FIFO. I nuovi processi si aggiungono alla fine della coda dei processi pronti.

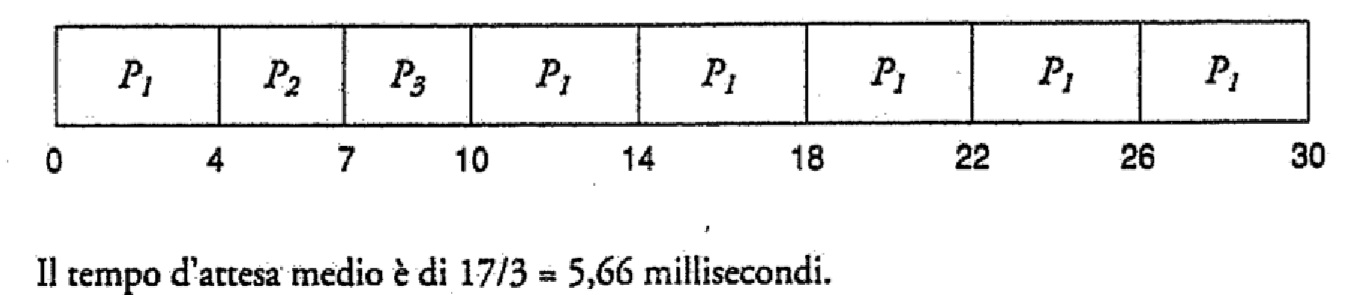
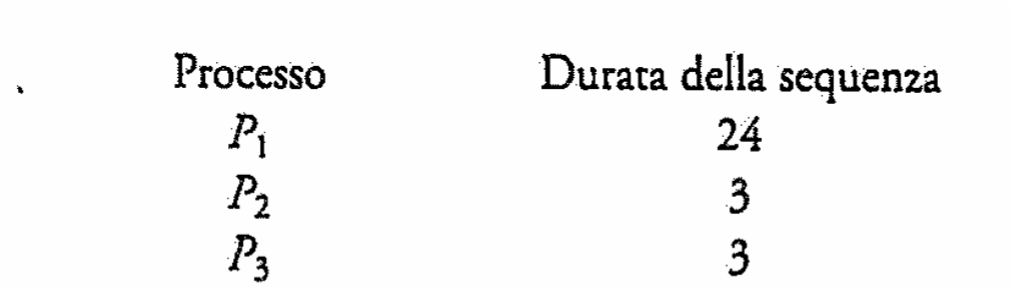
-L'algoritmo R-R è con prelazione , in quanto se un processo eccede nel Time-Slice il processo viene riportato nella coda dei processi pronti.

-Le prestazioni dell'algoritmo dipendono molto dalle dimensioni del quanto di tempo.

Nel caso in cui il quanto di tempo sia molto lungo(indefinito) , il criterio di scheduling R-R si riduce al criterio FCFS.

-Al diminuire del quanto di tempo , aumentano i cambi di contesto. Pertanto segue che il quanto di tempo deve essere ampio rispetto alla durata del cambio di contesto.

Inoltre il tempo di completamento non migliora necessariamente con l'aumento della dimensione del quanto di tempo ; in generale il tempo medio di completamento può migliorare se la maggior parte dei processi termina la sequenza in un solo quanto di tempo.



p1 -> 10-4= 6 ms , p2-> 4 ms , p3-> 7 ms ; 6+4+7=17 -> 17/3=5,66 ms

## 

## Scheduling a code multiple

E’ stata creata una classe di algoritmi di scheduling adatta a situazioni in cui i processi si possono classificare facilmente in gruppi diversi. Una distinzione diffusa è per esempio quella che si fa tra i processi che si eseguono in primo piano (foreground) e quelli che si eseguono in sottofondo (background).

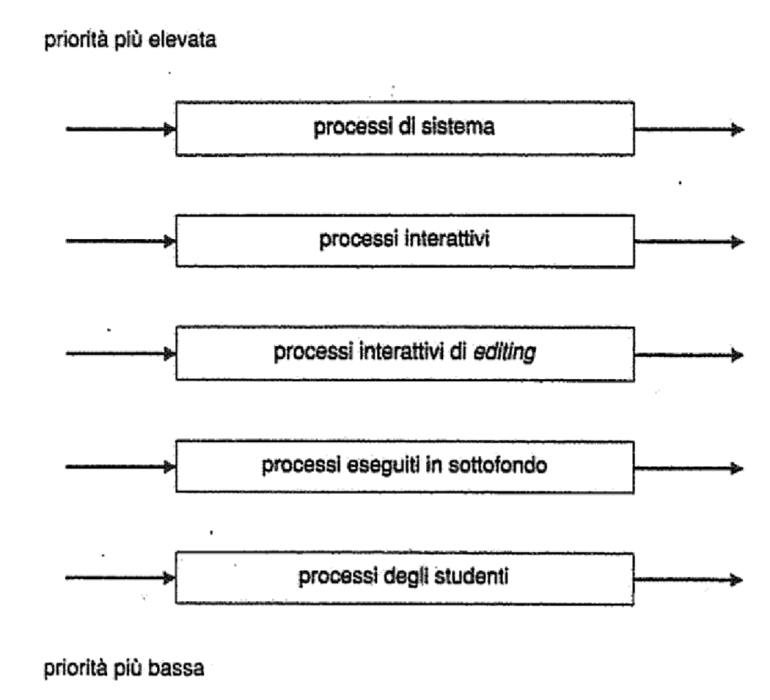
Questi 2 tipi di processi hanno tempi di risposta differenti e pertanto possono aver bisogno di diversi scheduling.

L'algoritmo di scheduling a code multiple suddivide la coda dei processi pronti in code distinte ; i

processi si assegnano in modo permanente ad una coda , di solito in base a qualche caratteristica , ed ogni coda ha un proprio algoritmo di scheduling.

Ogni coda ha la priorità assoluta sulle code di priorità più bassa.

Generalmente è necessario avere uno scheduling tra le code : in genere si usa uno scheduling per priorità fissa e con prelazione . Esiste anche la possibilità di impostare quanti di tempo per le code.

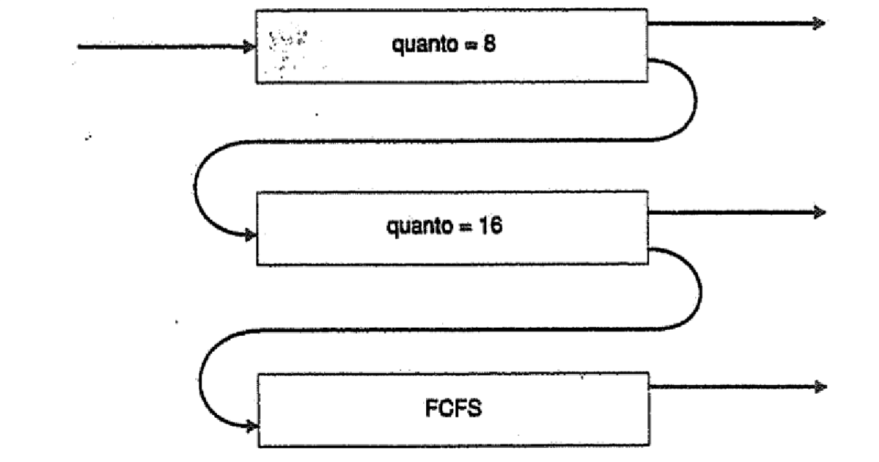


## Scheduling a code multiple con retroazione

Di solito in un algoritmo di scheduling a code multiple i processi vengono assegnati ad una coda e non possono quindi cambiarla.

Lo scheduling a code multiple con retroazione permette ai processi di spostarsi tra le code. L’idea consiste nel separare i processi che hanno caratteristiche diverse nelle sequenze delle operazioni della CPU.

- Si attua una forma di invecchiamento che impedisce il verificarsi dell’attesa indefinita.



Lo scheduler fa eseguire tutti i processi della coda 0 attuando l’algoritmo RR con quanto=8, non appena si svuota esegue quelli in coda 1 con RR a quanto=16, infine quelli nella coda 2 con algoritmo FCFS. All’inizio tutti vengono collocati nella coda 0, i processi che non terminano vengono collocati nella coda 1 che a loro volta se non terminano vengono collocati nella coda 2.

**La definizione di scheduling a code multiple con retroazione costituisce il più grande criterio di scheduling della CPU, che nella fase di progettazione si può adeguare a un sistema specifico.**

## 4.SCHEDULING PER SISTEMI MULTIPROCESSORE

## Soluzioni di scheduling per multiprocessore

Se sono disponibili più unita d’elaborazione, anche il problema dello scheduling è proporzionalmente più complesso.

Una prima strategia di scheduling della CPU per i sistemi multiprocessore affida tutte la decisione ad un CPU detta master e tutti i processi utenti alle altre cpu(MULTI-ELABORAZIONE ASIMMETRICA). Quando invece ciascun processore provvede al proprio scheduling, si parla di multielaborazione simmetrica SMP.

## Predilezione per il processore

Si consideri cosa accade alla memoria cache non appena che un processo abbia terminato la sua esecuzione nella CPU: i dati che ha trattato per ultimo permango nella cache, se il processo viene continuato da un altro processore, i contenuti della memoria cache devono essere invalidati sul processore di partenza e convalidati sull’altro. A causa di alti costi di svuotamento di memoria cache, si applicano delle regole impedendo ad un processo di passare da un processore ad un altro.

## Bilanciamento del carico

Sui sistemi SMP è importante che il carico sia distribuito uniformemente tra i vari processori in modo da non permettere che un processore sia saturo un altro sia inattivo. Il bilanciamento del carico quindi tenta di ripartire il carico di lavoro uniformante tra i processi. Ci sono 2 tecniche di bilanciamento: migrazione guidata e migrazione spontanea. La prima prevede che un processo controlli costantemente il carico di ogni processore e quando verifica che uno sia saturo, sposta i processi su altri processori. La 2 invece fa si che un processore inattivo sottragga spontaneamente processi a processori saturi.

## Multithread simmetrico

I sistemi SPM consentono l’esecuzione contemporanea, su processori fisici multipli di numerosi thread. Una possibile alternativa è quello che ha fatto l’intel noto come multithread simmetrico SMT che fornisce diversi processori logici. L’idea che caratterizza la SMT è la creazione di più processori logici basati sullo stesso processore fisico cosi da presentare al sistema operativo una serie di processori anche in presenza di uno solo fisico. Bisogna sapere che la SMT è a livello fisico, no software.