

# Appunti di sistemi operativi


Appunti per il corso universitario di sistemi operativi, riferito a sistemi Unix/Windows.  
Si discute su problemi di sincronizzazione, memoria e scheduling dei processi.


**LogMeIn Pro<sup>2</sup>**


Accedi al tuo PC ovunque ti trovi


Con LogMeIn Pro<sup>2</sup>, puoi:

- Condividere file e foto con altri
- Stampare documenti a distanza

 **Stampa**

 **Condividi**

 **Ascolta**

 **Incontra**

**Provalo gratis**

## ARGOMENTI

[INTRODUZIONE](#)

[INPUT/OUTPUT](#)

[GESTIONE DEI PROCESSI](#)

[ALGORITMI DI SCHEDULING](#)

[SCHEDULING MULTICPU](#)

[SISTEMI REAL TIME](#)

[SCHEDULING SU LINUX](#)

[SCHEDULING SU WINDOWS](#)

[OPERAZIONI SUI PROCESSI](#)

[COMUNICAZIONE TRA PROCESSI](#)

[THREAD](#)

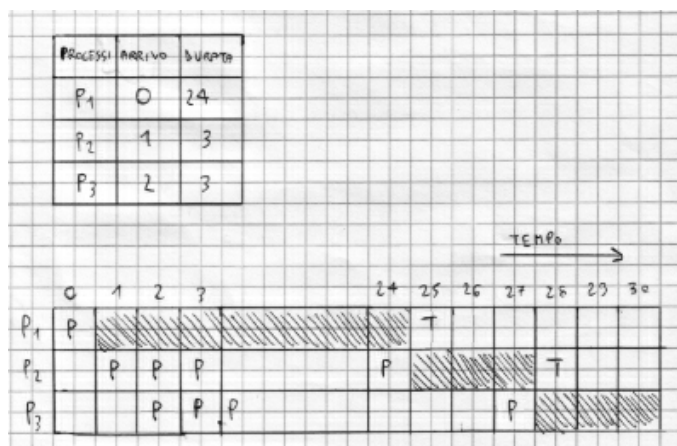
[SINCRONIZZAZIONE TRA PROCESSI](#)

[GESTIONE MEMORIA](#)

## FCFS (*first come first served*)

- algoritmo FIFO (sinonimo)
- non preemptive -> batch
- senza priorità (nella versione base)

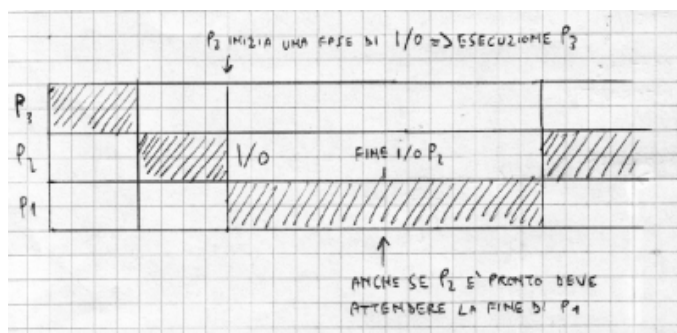
es. abbiamo 3 processi



(potevo segnare da 0 a 23 o da 1 a 24, basta che siano 24 istanti di tempo)

## DIFETTI FCFS:

1. difficile prevedere il risultato dello scheduling (molto dipendente dall'ordine con cui arrivano i processi -> non sempre efficiente)
2. come tutti gli scheduler batch sfavorisce i **processi corti** (basta pensare ai tempi di attesa)
3. sfavorisce i processi **I/O bound** (vedi figura qui sotto)



## SJF (*shortest job first*)

detto anche SPN (shortest process next), si tratta di un miglioramento dell'algoritmo FCFS: tra i processi pronti viene eseguito prima quello che userà la CPU per il **tempo minore**

è ottimo rispetto al criterio del tempo di attesa (non ci sono algoritmi batch che permettano un tempo di attesa inferiore)

è necessario avere una previsione del tempo di esecuzione dei prossimi processi; può essere specificato dall'utente ma in genere lo scheduler deve predire questo valore

la lunghezza della successiva sequenza di operazioni della CPU si ottiene calcolando la media esponenziale delle effettive lunghezze delle precedenti sequenze

$t_n$  = lunghezza dell'ennesima sequenza di operazioni

$\tau_{n+1}$  = il valore previsto per la successiva sequenza di operazioni

$\alpha$  = peso relativo sulla predizione della storia recente e quella passata (con  $\alpha = 0$  la storia recente non ha effetto; con  $\alpha = 1$  ha effetto solo la storia recente)

solitamente  $\alpha = 1/2$

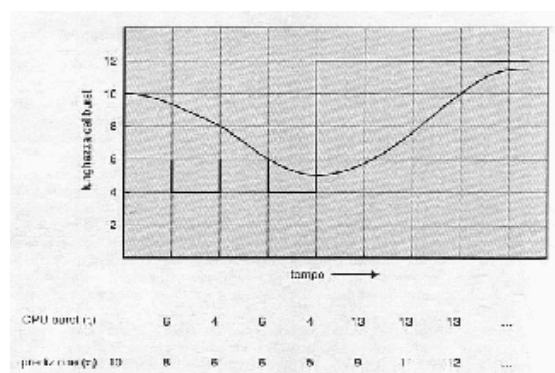
formula della media esponenziale:

$$\tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n$$

il  $\tau_0$  iniziale si può definire come una costante o come una media complessiva del sistema

la formula è semplice perché deve fornire velocemente al sistema i risultati (devo solo memorizzare l'ultimo valore reale del tempo di CPU e della sua stima)

il grafico rappresenta una media esponenziale con  $\alpha = 1/2$



sviluppo della formula precedente:

$$\tau_{n+1} = \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \dots + (1 - \alpha)^j \alpha t_{n-j} + \dots + (1 - \alpha)^{n+1} \tau_0$$

siccome sia  $\alpha$  sia  $1 - \alpha$  sono minori o uguali a 1, ogni termine ha peso inferiore a quello del

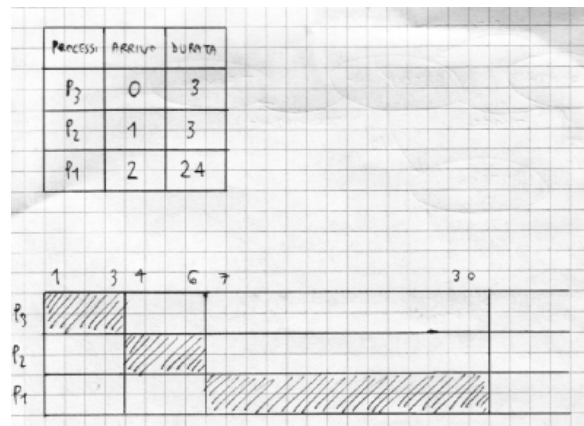
predecessore

L'SJF può essere realizzato sia con prelazione sia senza

## MISURE DI PRESTAZIONI

criteri:

- **produttività (throughput)**: sistema, numero di processi terminati nell'unità di tempo (nell'esempio sotto ho terminato 3 processi in 30 unità di tempo (poniamo millisecondi)  $\rightarrow 3 / 30 = 0,1 \text{ proc/msec} = 100 \text{ proc/sec}$ )
- **tempo di completamento** = durata del passaggio dallo stato nuovo allo stato terminato; si considera in genere turnaround medio = media fra i tempi di turnaround dei vari processi (nell'esempio i turnaround sono: 24, 26, 28  $\rightarrow$  turnaround medio = 26)
- **tempo di attesa (medio)** = tempo trascorso nella coda dei processi pronti (nell'esempio: 0, 2, 4  $\rightarrow$  2)
- non sono tanto importanti i valori medi, quanto la **varianza** (quanto si discostano dai valori medi): nel sistema dell'esempio la varianza è alta (i valori dei singoli processi si discostano molto dal valore medio calcolato)
- anche con processi con tempi differenti possiamo trovare la stessa produttività del sistema



turnaround: 3,5,28  $\rightarrow$  medio = 12

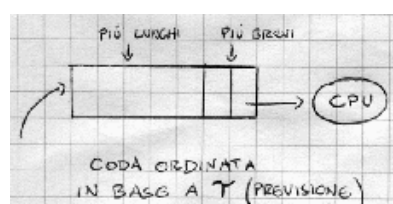
## PRIORITÀ

### PRIORITÀ INTERNA

è quella definita dal sistema in base a diversi requisiti (memoria, I/O, tempo di esecuzione, ecc.)

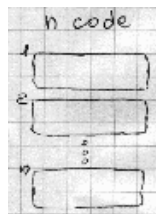
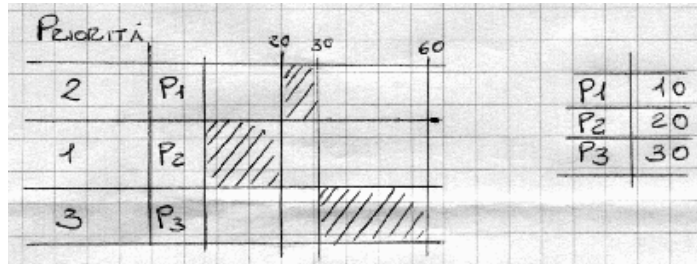
scheduling batch (non preemptive):

- FCFS senza priorità
- SJF con priorità statica (non varia prima del dispatch del processo)



## PRIORITÀ ESTERNA

è la priorità data dagli utenti (e non calcolata automaticamente dal sistema)



è meglio avere n code, una per ogni priorità; il processo ha all'interno della sua descrizione il livello di priorità, quindi è sufficiente leggere tale valore e porre il processo nella coda appropriata

il dispatcher scandisce le code in ordine di priorità decrescente (ricordare che le priorità sono numerate al contrario), e schedula finché non trova la prima coda non vuota; a questo punto esegue il primo processo della coda

## STARVATION

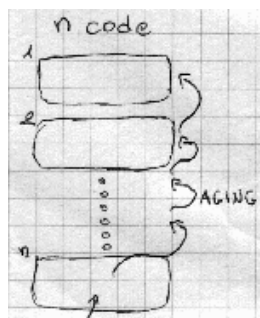
= (letteralmente) morte per fame

**non esiste un limite di tempo entro il quale P verrà sicuramente eseguito**

ossia se c'è un processo a priorità bassa P e molti processi a priorità più alta, i processi a priorità alta passano sempre davanti (sia come processo singolo che come coda); quindi non si riesce a dare un limite per l'esecuzione di P; se la situazione permane per molto tempo, P potrebbe aspettare molto per essere eseguito

come evitarla?

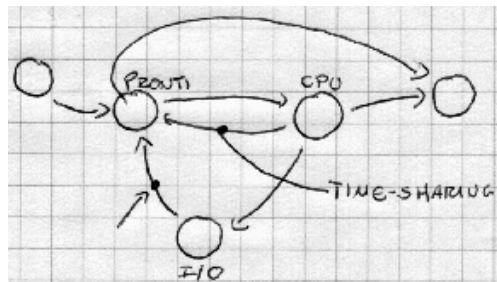
**AGING (priorità dinamiche):**



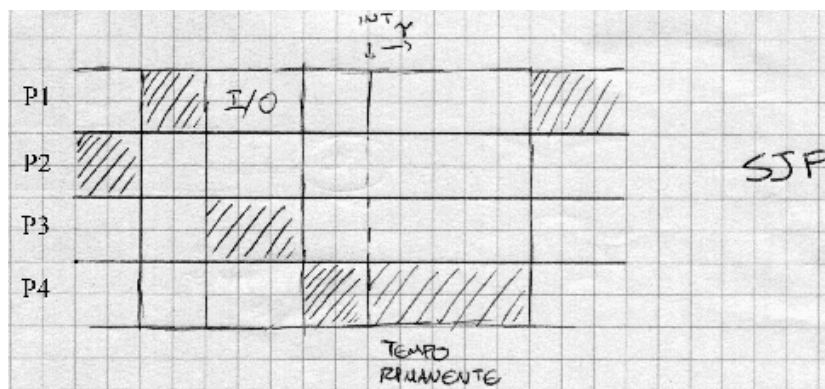
quando un processo è pronto da "troppo tempo" (soglia) la sua **priorità** viene **aumentata** periodicamente dallo scheduler; in questa maniera acquisirà priorità sufficiente da poter essere preso dal dispatcher ed eseguito

## PREEMPTION

se nella coda dei processi pronti ho un processo che ha una priorità maggiore di quello in esecuzione nella CPU, allora forzo il rilascio della CPU e passo ad eseguire il processo suddetto



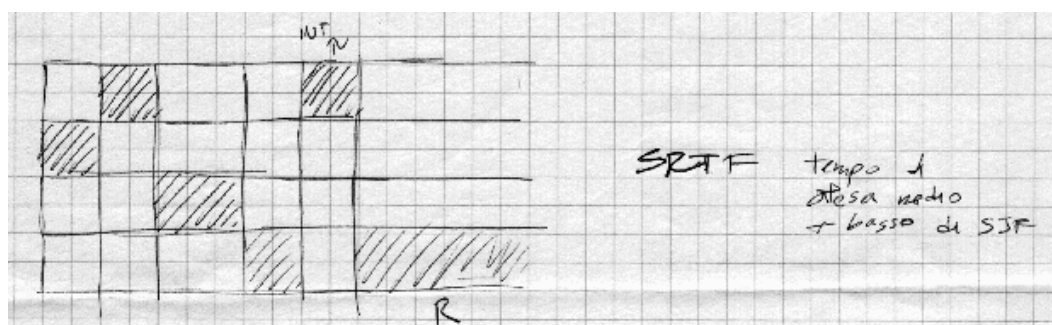
es. (sistema batch con scheduler FCFS e SJF, ma non dotato di preemption)



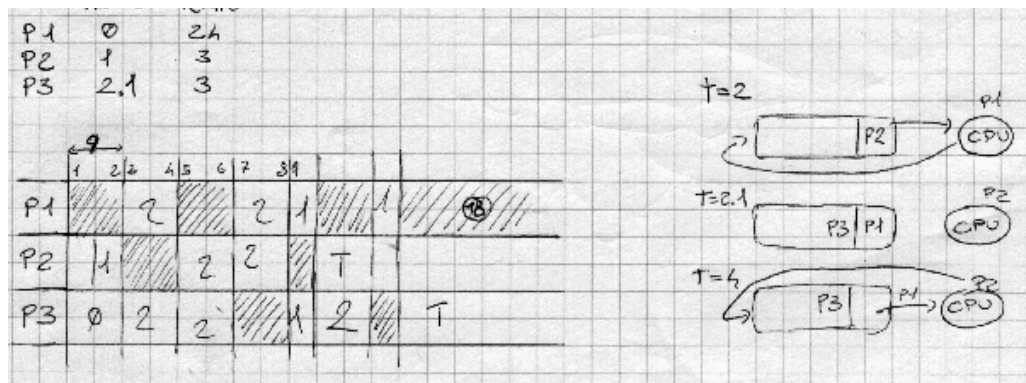
supponiamo che P1 esegua un operazione di I/O, se si risveglia al punto x allora dovrà aspettare la fine dell'esecuzione di P4; se inserisco il meccanismo della preemption, è sufficiente dare a P1 una maggiore priorità, e P4 verrà interrotto per far terminare P1

## SHORTEST REMAINING TIME FIRST (SRTF)

SRTF è l'acronimo con cui si indica uno **scheduling SJF con diritto di prelazione**



se arriva un nuovo processo pronto confronto  $t$  (tempo atteso di esecuzione del nuovo processo) con il tempo rimanente (atteso)  $R$  del processo che sto eseguendo, ed eseguo subito il processo se  $t < R$  (preemption)



## SCHEDULING TIME-SHARING

la base dello scheduling time-sharing è la presenza di algoritmi di tipo **preemptive con timer**

abbiamo un quanto indivisibile (intervallo di esecuzione), che viene assegnato ai processi

## ALGORITMI PER LO SCHEDULING TIME-SHARING

### ROUND ROBIN

Round-Robin (RR) = **girotondo** = FCFS preemptive con quanto di tempo

questo algoritmo:

- funziona meglio dell'FCFS, perché anche nel caso peggiore fa comunque girare i processi
- è meno efficiente del SRTF con preemption
- è più equo del SRTF con preemption perché i processi finiscono in momenti più vicini

### DIVISIONE DEL QUANTO

più il **quanto** è **grande** e **minore** è il livello di **interattività** -> vado verso algoritmo FCFS

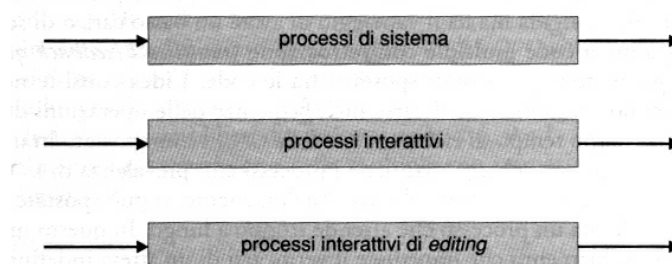
q = infinito FCFS (no timer)

se il quanto è troppo **piccolo** ho molti **context switch** -> perdo molto tempo per lo scheduling

## SCHEDULING A CODE MULTIPLE

- combinazione di scheduling diversi
- possiamo avere code separate di processi di tipo diverso con opportuni scheduler

priorità più elevata





quanto infinito

caratteristiche configurabili:

- n° di code
- scheduling della coda
- dinamica dei processi (come cambiano coda): in base al tipo di I/O seleziono in che coda metterli (es. una tastiera denota un grado di interattività maggiore rispetto ad un disco)

all'inizio i processi sono tutti nella coda più alta (RR)

[continua..](#)

[Ritorna sopra](#) | [Home page](#) | [Xelon](#)