La schedulazione

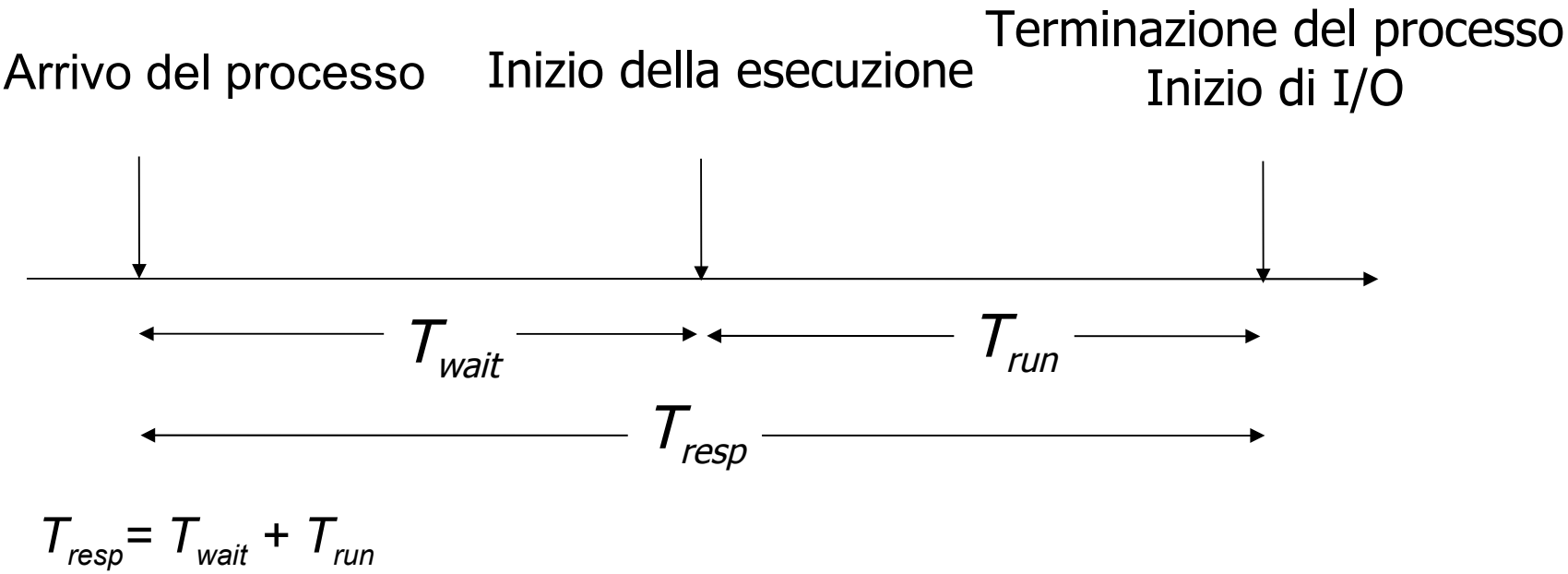
E.Mumolo

mumolo@units.it

Concetti fondamentali

- Multiprogrammazione:
  - esecuzione simultanea di più sequenze di esecuzione
  - Pseudo-parallelismo su una sola CPU
  - Esecuzione parallela su più CPU
  - Vantaggi della multiprogrammazione:
    - aumento della utilizzazione della CPU, convenienza operativa, convenienza espressiva
  - Costo della multiprogrammazione:
    - Overhead del context switch
    - Diminuzione delle prestazioni dovute alla contesa delle risorse
    - Problemi di mutua esclusione, stallo, sincronizzazione, determinatezza → complessità della programmazione

Metriche

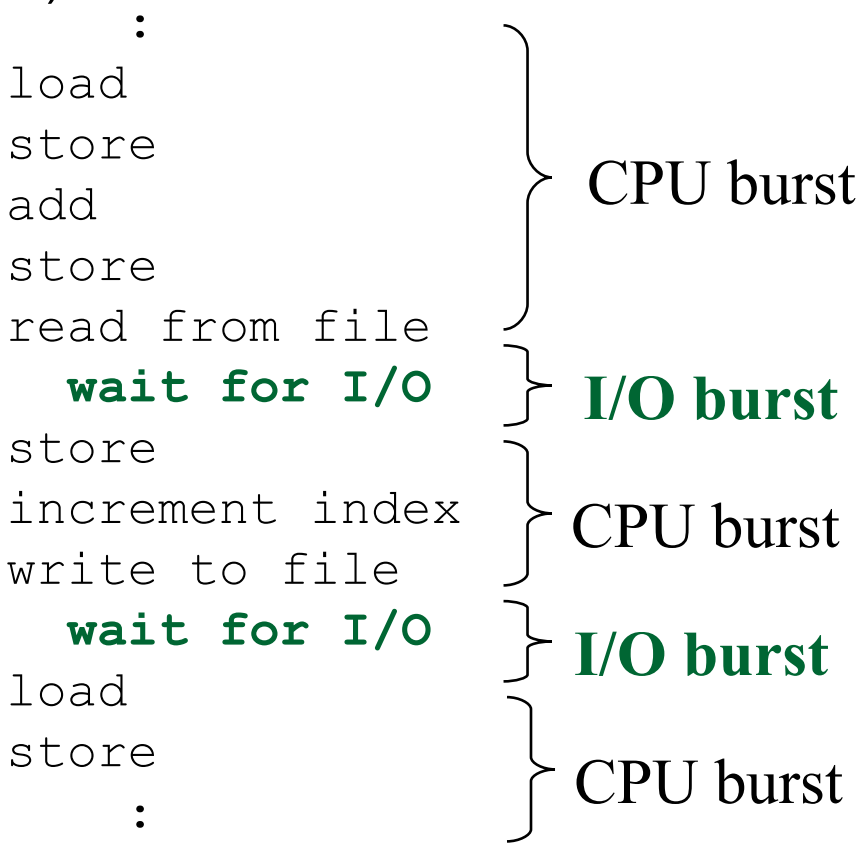


$T_{resp} = T_{wait} + T_{run}$

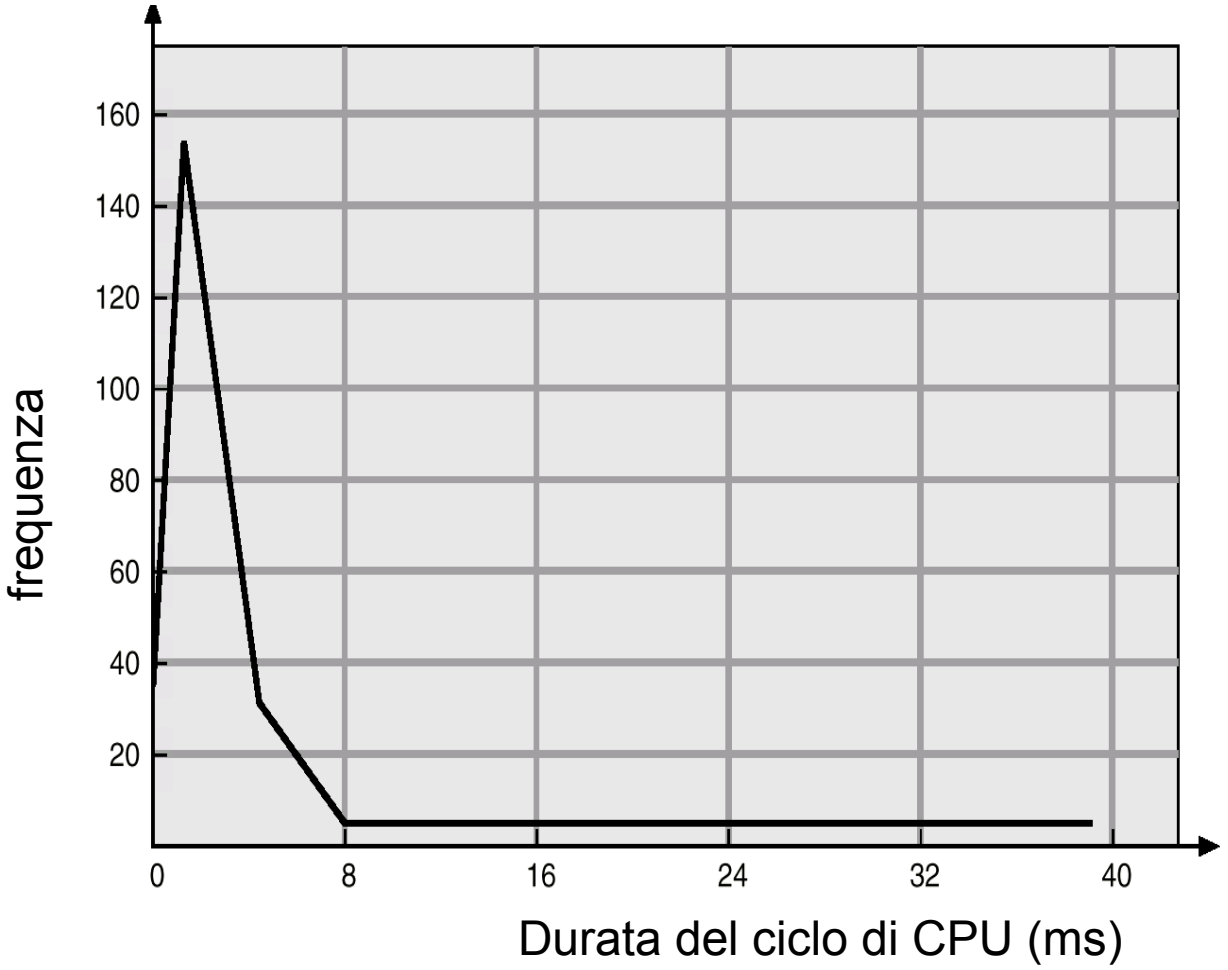
- Il tempo di turnaround è la media dei valori T_{resp}
- Il tempo d'attesa è la media dei valori T_{wait}
- Throughput: numero di terminazioni nell'unità di tempo
- Utilizzazione CPU: frazione di tempo nella quale la CPU è utilizzata

Cicli di CPU e cicli di attesa di I/O

Processo in esecuzione: sequenza di cicli CPU e cicli di attesa di I/O (CPU burst, I/O burst)



Esempio di istogramma di CPU burst



Stima approssimata del prossimo CPU burst

- La sequenza di CPU burst appare come una serie numerica: $t_0\, t_1\, t_2\, t_3 \ldots t_n\, t_{n+1}$
- Stimando t_{n+1} con predizione lineare: $\tau_{n+1} = \sum_{i=0}^{n} \alpha_i t_{n-i}$
- I coefficienti α_i possono essere calcolati con tecniche che utilizzano la cross-correlazione
- Tenendo conto del fatto che i coefficienti sono decrescenti perchè la predizione ha un orizzonte limitato, è ragionevole porre $\alpha_i = w(1-w)^i$, con $w < 1$.
- Predizione lineare con α_i:

  $\tau_{n+1} = \alpha_0 t_n + \alpha_1 t_{n-1} + \alpha_2 t_{n-2} + \ldots = wt_n + w(1-w)t_{n-1} + w(1-w)^2 t_{n-2} + \ldots = wt_n + (1-w)[wt_{n-1} + w(1-w)t_{n-2} + \ldots]$

- Questa espressione è rappresentata dalla media esponenziale:

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

- Significato dei termini:
  - τ_{n+1} = predizione del ($n+1^{th}$) ciclo di CPU; t_n = durata nota del ciclo di CPU corrente (n^{th})
  - τ_n = predizione precedente (basata sulla sequenza dei tempi t_i precedenti)
  - w = peso della media. Valore minore di 1 ($0 <= w <= 1$); di solito $w = 1/2$

Esempi di medie esponenziali

- w =0
  - $\tau_{n+1} = \tau_n$
  - La storia recente non conta
- w =1
  - $\tau_{n+1} = t_n$
  - Conta solo il valore precedente di CPU
- Espandendo la formula:

$\tau_{n+1} = w\ t_n + (1 - w)\ \alpha\ t_{n-1} + \ldots$

$+ (1 - w)^{j}\ \alpha\ t_{n-1} + \ldots$

$+ (1 - w)^{n=1}\ t_n\ \tau_0$

- Ogni peso della media è minore del precedente, perchè w e (1 - w) sono <= 1.

Criteri di schedulazione

- Ottimizzazione di uno dei seguenti parametri:
  - Utilizzazione della CPU
    - Scopo: mantenere la CPU più impegnata possibile
  - Throughput
    - Scopo: aumentare il numero di processi completati nell'unità di tempo
  - Tempo di Turnaround
    - Scopo: minimizzare il tempo medio richiesto per completare I processi
  - Tempo d'attesa
    - Scopo: minimizzare il tempo medio d'attesa in coda da parte dei processi
  - Tempo di risposta
    - Scopo: minimizzare il tempo richiesto time a process takes to start responding

Alcuni algoritmi di schedulazione

First-Come, First-Served (FIFO)

Shortest Job First (SJF)

Shortest Remaining Time first (SRT)

Round Robin (RR)

Multilevel Queue Scheduling

Multilevel Feedback Queue Scheduling

Schedulazione FIFO

- Quando la CPU è disponibile, assegnala al primo processo in coda d'attesa.
- Caratteristiche positive
  - Semplice da implementare (semplice coda FIFO)
  - Non richiede context switch perchè è una schedulazione 'non-preemptive'
- Caratteristiche negative
  - Il tempo medio d'attesa dipende dall'ordine dei processi
  - Processi di piccola durata possono restare in attesa a lungo dietro processi di lunga durata

Esempio FIFO

- Tutti I processi arrivano all'istante 1
- Process tempo di CPU

P1	24
P2	3
P3	3

- Carta di Gantt:

- Tempo d'attesa medio:

(0 + 24 + 27)/3 = 17 ms

Esempio FIFO

- Process tempo di CPU

P2	3
P3	3
P1	24

- Carta di Gantt:

- Tempo d'attesa medio:

(6 + 0 + 3)/3 = 3 ms

Schedulazione Shortest Job First (SJF)

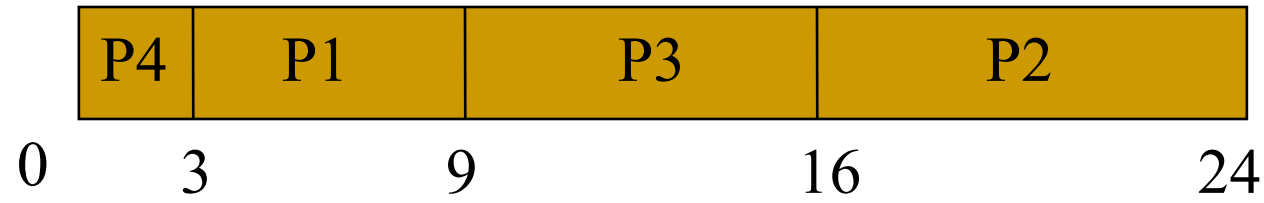
- Quando la CPU è disponibile, assegnala al processo con il più piccola tempo del prossimo ciclo di CPU
  - Un migliore nome dovrebbe essere “shortest next CPU burst”
- Può essere pre-emptive o no
- Si può dimostrare che SJF è una schedulazione ottima
- Problema: non si conosce la durata del tempo di CPU

Esempio SJF

- Non-preemptive
- Process tempo di CPU

P1	6
P2	8
P3	7
P4	3

- Carta di Gantt:



- Tempo medio d'attesa:

(3 + 16 + 9 + 0)/4 = 7 ms

  - FIFO darebbe 10.25 ms

Schedulazione con priorità

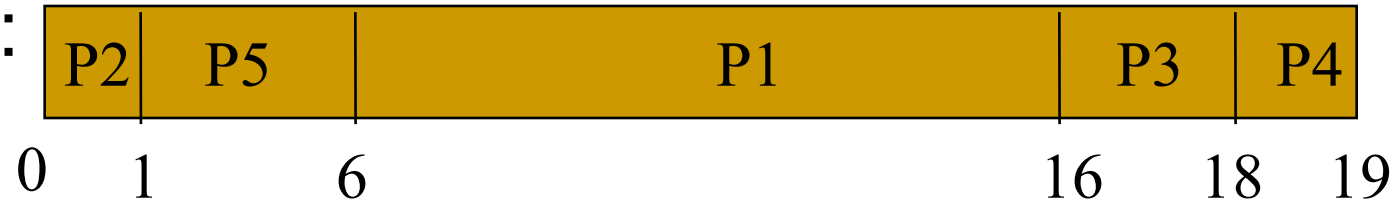
- Associa una priorità ad ogni processo e la CPU viene allocata al processo con la più alta priorità.
- Priorità interne/esterne.
- Preemptive o non-preemptive.
- Evitare l'attesa indefinita → aging

Esempio di schedulazione con priorità

- Processo Tempo di CPU Priorità

P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

- Gantt :



- Tempo medio d'attesa: 8.2 ms

Schedulazione Round Robin (RR)

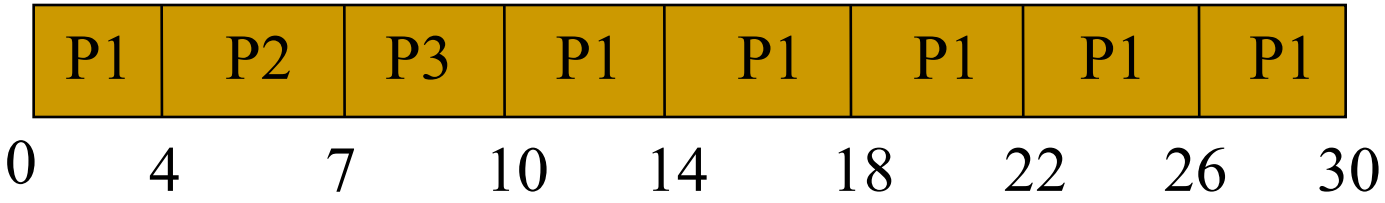
- Viene definito un quanto temporale, tipicamente 10-100 ms
- La coda dei processi pronti è trattata come una coda circolare.
- Lo schedulatore seleziona un processo alla volta eseguendolo per un quanto temporale.
- Se il quanto è grande le prestazioni tendono al FIFO
- Se il quanto è piccolo, deve essere comunque molto più grande del context switch, altrimenti l'overhead è troppo alto
- Tecnica preemptive

Esempio di RR

- Quanto = 4 ms.
- Processo Tempo di CPU (Burst Time)

P1	24
P2	3
P3	3

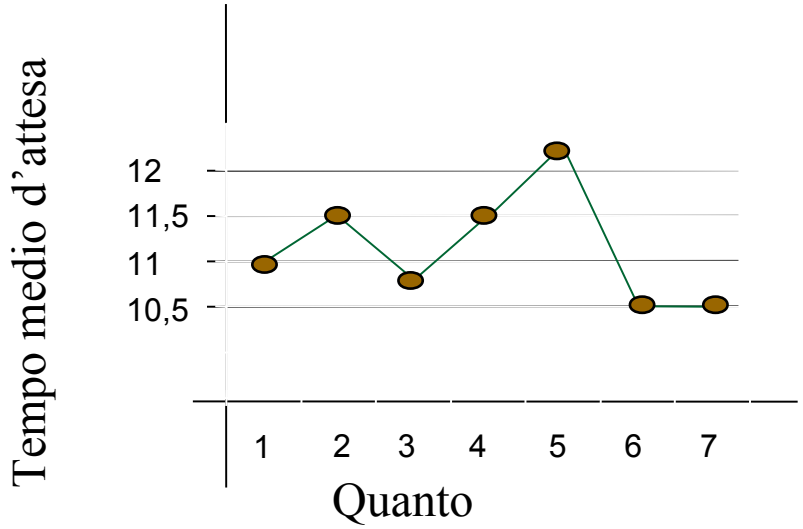
- Gantt:



- Tempo d'attesa medio: 17/3 = 5.67 ms

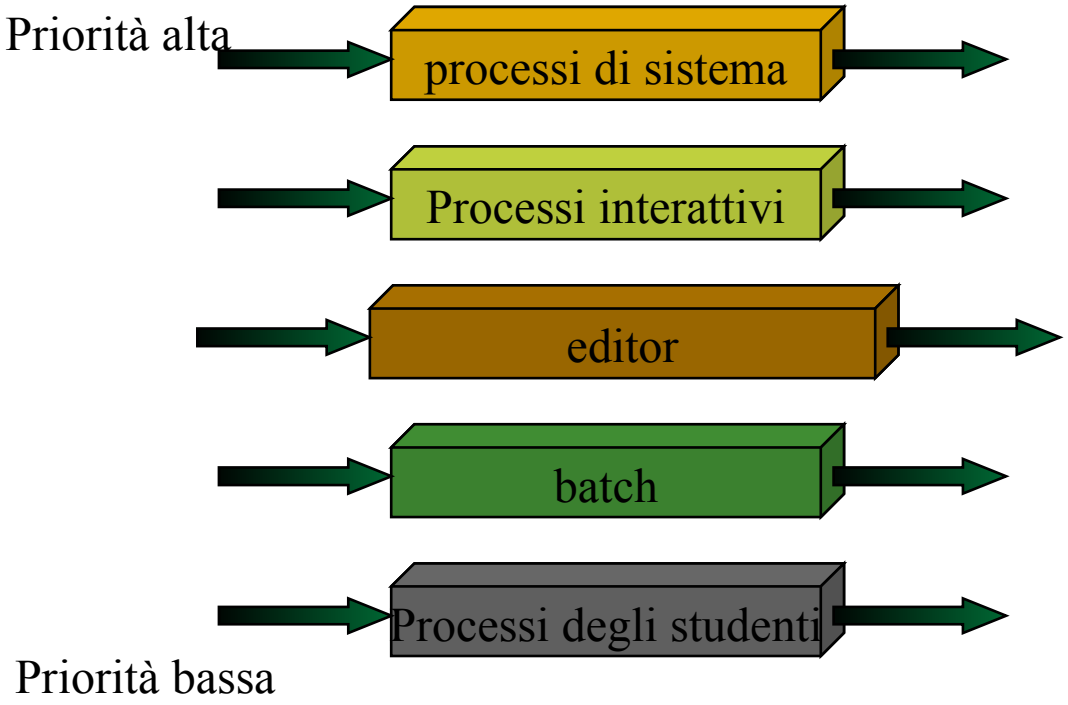
Discussione

- Il tempo medio d'attesa può essere piuttosto lungo
- Il context switch è importante quando il quanto è piccolo
- Il tempo di turnaround medio non aumenta necessariamente se aumenta il quanto:



Processo	Time
P1	6
P2	3
P3	1
P4	7

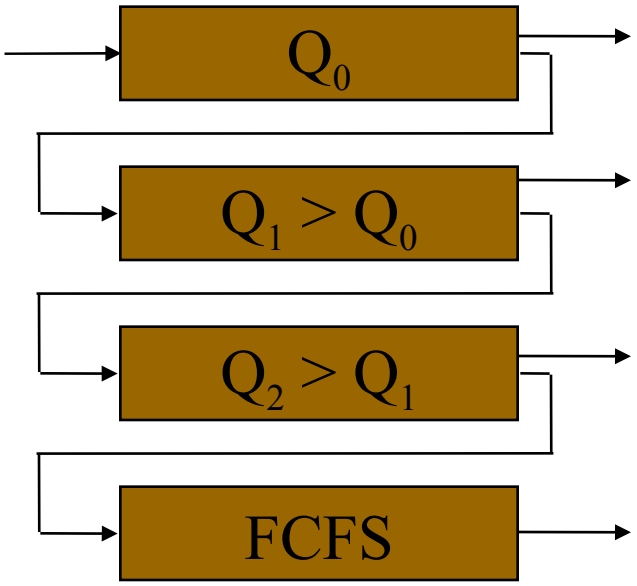
Multilevel Queue Scheduling



- Schedulazione tra le code:
  - Schedulazione pre-emptive a priorità fissa
  - Quanti variabili tra le code

Multilevel Feedback Queue Scheduling

- I processi si spostano da coda a coda
  - Ogni coda ha associato un quanto temporale
  - Se un processo dura più del quanto si sposta sulla coda più bassa
  - Agevola i processi più brevi
  - I processi in una coda vengono interrotti dalle code superiori



Schedulazione LINUX

- Ogni processo ha un quanto di esecuzione
- Priorità dinamica $\tilde{=}$ Base + K/τ_{n+1}
- Aggiornamento dinamico della priorità: lo schedulatore ricalcola le priorità prima di scegliere il prossimo processo
- Schedulazione equa