

SCHEDULING DEI PROCESSI

- **Algoritmo First Come First Served (FCFS)** : i processi vengono eseguiti nell'ordine di arrivo
- **Algoritmo Shortest Job First (SJF)** : i processi vengono eseguiti da quello con il minor al maggior tempo di esecuzione rimanente
- **Algoritmo con priorit ** : i processi vengono eseguiti secondo un ordine definito da una priorit 
- **Algoritmo Round Robin (RR)** : viene assegnato un quanto di tempo ad ogni processo, scaduto il quale la cpu viene assegnata al successivo processo

- **Algoritmo First Come First Served (FCFS)** : e' senza prelazione
- **Algoritmi (SJF) e con priorit ** possono essere
 - Con prelazione
 - Senza prelazione
- **Algoritmo Round Robin (RR)** : e' sempre con prelazione

Siano i seguenti processi nella ready queue, arrivati nell'ordine specificato e tutti presenti ad un dato istante 0

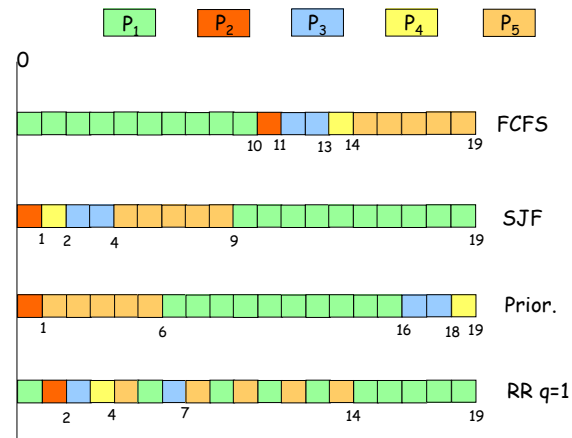
Proc.	Esecuz.	prior.
P ₁	10	3
P ₂	1	1
P ₃	2	3
P ₄	1	4
P ₅	5	2

si considerino i seguenti algoritmi di scheduling:

- FCFS
- SJF senza prelazione
- priorit  senza prelazione (p=0 pr. Max)
- Round robin con q=1

1. Si disegnano i 4 diagrammi di Gantt
2. Calcolare i tempi di completamento nei 4 casi
3. Calcolare il tempo medio di completamento nei 4 casi
4. Calcolare la deviazione standard nei casi SJF e RR

Diagrammi di Gantt



6-bis. Esercizi sullo scheduling

5

marco lapegna

Tempi di completamento e medi

- **FCFS**
 - P₁ → 10 P₂ → 11 P₃ → 13 P₄ → 14 P₅ → 19
 - T_m = 67/5 = **13.4**
- **SJF**
 - P₁ → 19 P₂ → 1 P₃ → 4 P₄ → 2 P₅ → 9
 - T_m = 35/5 = **7** s = **6.6**
- **Priorita'**
 - P₁ → 16 P₂ → 1 P₃ → 18 P₄ → 19 P₅ → 6
 - T_m = 60/5 = **12**
- **RR q=1**
 - P₁ → 19 P₂ → 2 P₃ → 7 P₄ → 4 P₅ → 14
 - T_m = 46/5 = **9.2** s = **6.3**

6-bis. Esercizi sullo scheduling

6

marco lapegna

Esercizio 2:

Siano i seguenti processi nella ready queue, arrivati nei tempi specificati

Proc.	arrivo	esecuz.	si considerino i seguenti algoritmi di scheduling:
P ₁	0.0	8	<ul style="list-style-type: none"> ▪ FCFS senza prelazione ▪ SJF senza prelazione
P ₂	0.5	4	
P ₃	1.0	1	

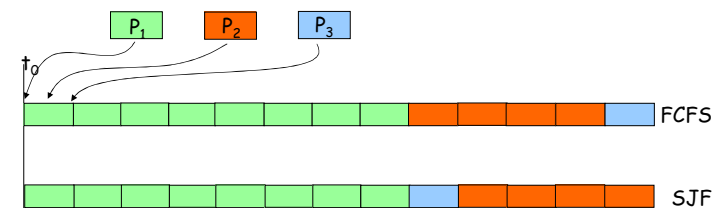
1. Si disegnano i **diagrammi di Gantt**
2. Calcolare i **tempi di completamento** nei 2 casi
3. Calcolare il **tempo medio di completamento** nei 2 casi

6-bis. Esercizi sullo scheduling

7

marco lapegna

soluzione



- **FCFS**
 - P₁ → 8 P₂ → 11.5 P₃ → 12
 - T_m = 31.5/3 = **10.5**
- **SJF**
 - P₁ → 8 P₂ → 12.5 P₃ → 8
 - T_m = 28.5/3 = **9.5**

6-bis. Esercizi sullo scheduling

8

marco lapegna

Esercizio 3:

Siano i seguenti processi nella ready queue, arrivati nell'ordine specificato e tutti presenti ad un dato istante 0

Proc. Esecuz.

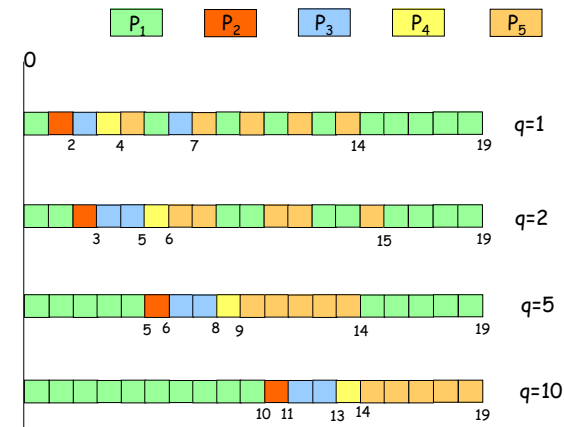
P_1	10
P_2	1
P_3	2
P_4	1
P_5	5

si considerino l'algoritmo RR con i seguenti quanti di tempo:

- $q=1$
- $q=2$
- $q=5$
- $q=10$

1. Si disegnano i 4 diagrammi di Gantt
2. Calcolare i tempi di completamento nei 4 casi
3. Calcolare il tempo medio di completamento nei 4 casi

Diagrammi di Gantt



Tempi di completamento e medi

- $q=1$
 - $P_1 \rightarrow 19$ $P_2 \rightarrow 2$ $P_3 \rightarrow 7$ $P_4 \rightarrow 4$ $P_5 \rightarrow 14$
 - $T_m = 46/5 = 9.2$
- $q=2$
 - $P_1 \rightarrow 19$ $P_2 \rightarrow 3$ $P_3 \rightarrow 5$ $P_4 \rightarrow 6$ $P_5 \rightarrow 15$
 - $T_m = 48/5 = 9.6$
- $q=5$
 - $P_1 \rightarrow 19$ $P_2 \rightarrow 6$ $P_3 \rightarrow 8$ $P_4 \rightarrow 9$ $P_5 \rightarrow 14$
 - $T_m = 56/5 = 11.2$
- $q=10$
 - $P_1 \rightarrow 10$ $P_2 \rightarrow 11$ $P_3 \rightarrow 13$ $P_4 \rightarrow 14$ $P_5 \rightarrow 19$
 - $T_m = 67/5 = 13.4$

quindi

- q piccolo \rightarrow tempi medi di esecuzione piccoli
- q grande \rightarrow tempi medi di esecuzione grandi

MA

Non si è considerato il tempo di cambio di contesto



Si ripeta l'esercizio considerando il tempo di cambio di contesto $\tau=1$

Tempi medi di completamento con c.s.

q=1

- P1 soffre di 14 cs, P2 di 1, P3 di 6, P4 di 3 e P5 di 13 cs
- Tempo medio di compl. $(46+37)/5 = 83/5 = 16.6$

q=2

- P1 soffre di 9 cs, P2 di 1, P3 di 2, P4 di 3 e P5 di 8 cs
- Tempo medio di compl. $(48+23)/5 = 71/5 = 14.2$

q=5

- P1 soffre di 5 cs, P2 di 1, P3 di 2, P4 di 3 e P5 di 4 cs
- Tempo medio di compl. $(56+15)/5 = 71/5 = 14.2$

q=10

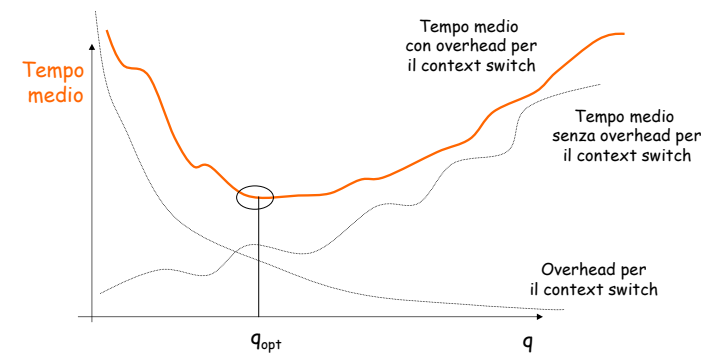
- P1 soffre di 0 cs, P2 di 1, P3 di 2, P4 di 3 e P5 di 4 cs
- Tempo medio di compl. $(67+10)/5 = 77/5 = 15.4$

6-bis. Esercizi sullo scheduling

13

marco lapegna

In generale



6-bis. Esercizi sullo scheduling

14

marco lapegna

Esercizio 4:

Siano i seguenti processi nella ready queue, arrivati nell'ordine specificato

Proc. arrivo. esec.

P ₁	0	7
P ₂	2	4
P ₃	3	4
P ₄	5	2
P ₅	7	3
P ₆	10	2

si consideri l'algoritmo SJF

- senza prelazione
- con prelazione

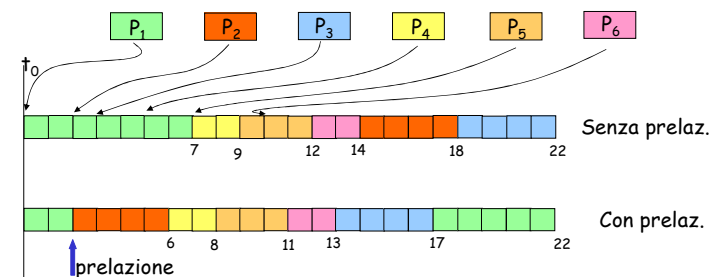
- Si disegnano i diagrammi di Gantt
- Calcolare i tempi di completamento nei 2 casi
- Calcolare il tempo medio di completamento nei 2 casi

6-bis. Esercizi sullo scheduling

15

marco lapegna

soluzione



- Senza prelazione
 - $P_1 \rightarrow 7$ $P_2 \rightarrow 16$ $P_3 \rightarrow 19$ $P_4 \rightarrow 4$ $P_5 \rightarrow 5$ $P_6 \rightarrow 4$
 - $T_m = 55/6 = 9.16$
- Con prelazione
 - $P_1 \rightarrow 22$ $P_2 \rightarrow 4$ $P_3 \rightarrow 14$ $P_4 \rightarrow 3$ $P_5 \rightarrow 4$ $P_6 \rightarrow 3$
 - $T_m = 50/6 = 8.33$

6-bis. Esercizi sullo scheduling

16

marco lapegna

Esercizio 5:

Siano i seguenti processi presenti al tempo 0

Proc. esec.

P ₁	10
P ₂	50
P ₃	40
P ₄	50
P ₅	30

Si usi scheduling a code multiple

- coda Q1 → P₁ P₂ P₃ con scheduling SJF s.p.
- coda Q2 → P₄ P₅ con scheduling RR (q1=20)
- scheduling tra le code RR (q2=30)

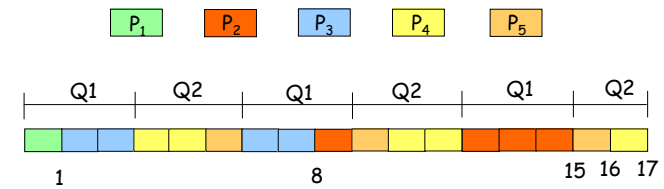
1. Si disegni il **diagramma di Gantt**
2. Calcolare i **tempi di completamento**
3. Calcolare il **tempo medio di completamento**

6-bis. Esercizi sullo scheduling

17

marco lapegna

soluzione



- P₁ → 1 P₂ → 15 P₃ → 8 P₄ → 17 P₅ → 16
- T_m = 57/5 = **11.4**

6-bis. Esercizi sullo scheduling

18

marco lapegna

Esercizio 6:

Siano i seguenti processi

Proc. esec. arrivo

P ₁	9	0
P ₂	6	2
P ₃	5	4
P ₄	2	5
P ₅	1	8

Si usi scheduling

- FCFS
- SJF con prelazione

Fornire i relativi diagrammi di Gantt

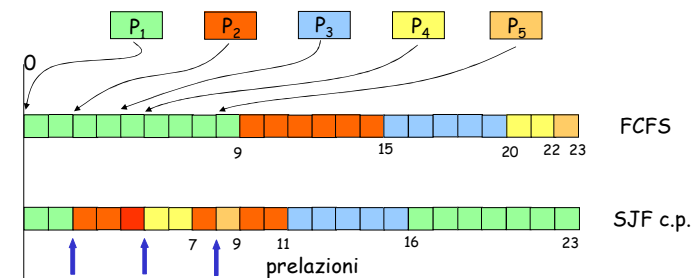
Nel caso di scheduling RR con q=10 ms, quanto deve durare il context switch per far terminare tutti i processi in 45 ms?

6-bis. Esercizi sullo scheduling

19

marco lapegna

soluzione



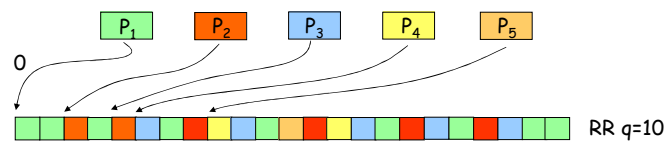
- **FCFS**
 - P₁ → 9 P₂ → 13 P₃ → 16 P₄ → 16 P₅ → 15
 - T_m = 69/5 = **13.8**
- **SJF Con prelazione**
 - P₁ → 23 P₂ → 9 P₃ → 12 P₄ → 2 P₅ → 1
 - T_m = 47/5 = **9.4**

6-bis. Esercizi sullo scheduling

20

marco lapegna

soluzione



Sia x il tempo del context switch

20 context switches \rightarrow tempo = $20x$ ms

Tempo esec dei processi = 23 ms

Tempo totale di esecuzione = $23 + 20x$

$23 + 20x < 45 \rightarrow x < 22/20 = 1.1$

note

- **SJF con prelazione**
 - T=2 arriva P2 con 6 u.t. mentre a P1 restano 7 u.t. \rightarrow **Prelazione**
 - T=4 arriva P3 con 5 u.t. mentre a P2 restano 4 u.t. \rightarrow No prelazione
 - T=5 arriva P4 con 2 u.t. mentre a P2 restano 3 u.t. \rightarrow **Prelazione**
 - T=7 finisce P4 va in esecuzione P2 con 3 u.t.
 - T=8 arriva P5 con 1 u.t. mentre a P2 restano 2 u.t. \rightarrow **Prelazione**
 - T=9 finisce P5 e vanno in esecuzione nell'ordine P2, P3 e P1 fino al loro completamento
- RR con q=10
 - T=2 supponiamo che P2 arriva un attimo prima della scadenza del quanto ed entra nella coda
 - T=4 supponiamo che P3 arriva un attimo prima della scadenza del quanto ed entra nella coda. Allo scadere del quanto, P2 va in esecuzione ed esce dalla coda mentre entra P1

Scheduler di Linux



- Vengono schedulati i **threads**
- Tre classi: **real time FCFS**, **real time Round Rodin**, **Time sharing**
- Ogni thread ha:
 - **Priorita' variabile p** nell'intervallo $[0, 40]$, dove $p=0$ e la **priorita' minima**. Generalmente all'inizio $p=20$
 - **Un credito c** (quanti di 10 ms da spendere)
- Lo scheduler calcola la **goodness (gdn)** di ogni thread come
 - Se (class == realtime) $gdn=1000 + p$
 - Se (class == timesharing & $c > 0$) $gdn = c+p$
 - Se (class == timesharing & $c == 0$) $gdn = 0$

Scheduler di Linux

- Ogni volta viene selezionato il **thread con gdn maggiore**
- Ogni 10 ms **c viene decrementato**
- Il **thread viene prelazonato** quando:
 - c diventa 0
 - Il thread si blocca
 - Nella ready queue entra un thread con gdn maggiore
- Quando $c = 0$, per tutti i processi nella ready queue viene ricalcolato c come segue:
 - $c = c/2 + p$

Esercizio

- Sia un sistema Linux dove nella ready queue sono presenti i seguenti threads:

- A: $c=5$, $p=8$, class = timesharing
- B: $c=15$, $p=4$, class = timesharing
- C: $c=9$, $p=X$, class = Y
- D: $c=7$, $p=15$, class = timesharing


Se B e' stato l'ultimo thread ad andare in esecuzione, si discuta in funzione di X e Y quale thread sara' il prossimo ad andare in esecuzione

Soluzione

- Se $Y == \text{realtime}$ allora

- A: $\text{gdn} = 13$
- B: $\text{gdn} = 19$
- C: $\text{gdn} = 1000 + X$ ← goodness maggiore qualunque sia X
- D: $\text{gdn} = 22$

- Se $Y == \text{timesharing}$ allora

- A: $\text{gdn} = 13$
 - B: $\text{gdn} = 19$
 - C: $\text{gdn} = 9 + X$
 - D: $\text{gdn} = 22$
-  $\left\{ \begin{array}{l} X > 13 \text{ il prossimo thread e' C} \\ X = 13 \text{ il prossimo thread e' C} \\ X < 13 \text{ il prossimo thread e' D} \end{array} \right.$