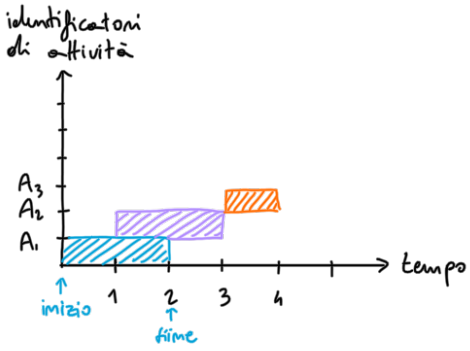
Coda Ready

POLITICHE DI SCHEDULING - servono a minimizzare il tempo medio di attesa

1. First Come First Served (FCFS)
2. Shortest Jobs First (SJF)
3. A Priorità
4. Round Robin
5. A Code Multilivelli
6. A Code Multilivelli con Feedback

Il throughput dipende dal tempo di attesa della coda ready

DIAGRAMMA DI GANTT 1917



Diagrammi utilizzati per lo scheduling / progettazione / monitoraggio di progetti (A_1, A_2, A_3)

Se vediamo i progetti come se fossero processi dobbiamo ricordarci può che i processi non possono sovrapporsi e sono SEQUENZIALIZZATI

esempio:

P	D
P_1	10
P_2	3
P_3	8
P_4	4

→ durata del prossimo CPU-BURST

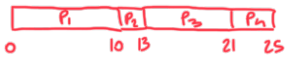
FCFS 

diagramma di Gantt dell'algoritmo FCFS

tempo medio di attesa dato dalla media dei tempi d'attesa dei processi

$P_1 = 0$
$P_2 = 10$
$P_3 = 13$
$P_4 = 21$
$\rightarrow \frac{44}{4} = 11$

se cambiamo l'ordine di esecuzione dei processi la media dei tempi d'attesa cambia

SJF permette un'ottimizzazione del tempo medio di attesa dei processi (POLITICA OTTIMA)
si ha una coda ordinata di processi ma ciò il rischio che processi molto lunghi non vengano mai eseguiti (fenomeno di STARVATION)

FORMULA PER CALCOLARE IL CPU-BURST

$$\tau_{m+1} = \alpha t_m + (1-\alpha)\tau_m \qquad \text{con} \quad \alpha \in [0,1]$$

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

Con τ che rappresenta la stima della durata del processo mentre t la sua durata effettiva mentre α deve essere scelto empiricamente attraverso simulazioni in fase di design

PRIORITÀ definita dagli utenti o in base a caratteristiche del processo

ROUND ROBIN prevede l'utilizzo del meccanismo di preemption

PRELAZIONE (PREEMPTION) → sottrazione delle risorse della CPU del processo running da parte dell'OS

è una variazione di FCFS

P	D
P_1	10
P_2	3
P_3	8
P_4	4

serve un parametro chiamato QUANTO DI TEMPO
se il QUANTO vale 4 l'OS fornisce ai processi 4 unità di tempo prima di interrompere il processo e metterlo in coda nuovamente (waiting)

$P_1 = 0 + (15-4) + (23-19) = 15$
$P_2 = 4$
$P_3 = 7 + (19-11) = 15$
$P_4 = 11$

TMA = 11,25

Algoritmo pensato per dispositivi che sfruttano il TIME SHARING, le risorse di calcolo della CPU sono distribuite in maniera equa e non si rischia starvation

CODE MULTILIVELLO — si distinguono i processi per tipo in code ready separate con priorità diverse

Le code multilivello con feedback presuppongono, oltre a un meccanismo di priorità delle code, un meccanismo di AGING per far scalare di priorità i processi in maniera tale da prevenire il fenomeno di starvation