Sistemi Operativi
AA 2019/20
Esercitazione

Esercizio 1

Data i processi `P1`, `P2` e `P3`, si assuma che vengano schedulati tramite algoritmo Round Robin con quanto $T = 5$. Avvalendosi della traccia di esecuzione illustrata in Figura 1, calcolare:

- Il tempo di attesa medio T_w (aka *Mean Waiting Time*)
- Il tempo di risposta medio T_r (aka *Mean Response Time*)
- Il tempo di evasione medio T_t (aka *Mean Turnaround Time*)

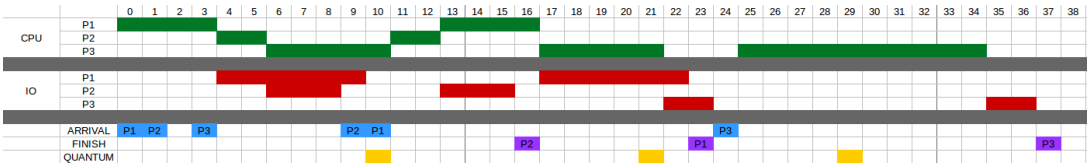

Figure 1: Traccia di esecuzione dei processi tramite RR preemptive con quanto $T = 5$.

Soluzione Le quantita' da calcolare fanno parte degli indicatori utilizzati per monitorare il comportamento di uno scheduler e sono definiti come segue:

1. T_w: il tempo che un processo attende nella coda di *ready* prima che venga messo in running
2. T_r: il tempo che intercorre tra la fine di un IO burst e la messa in esecuzione di un processo
3. T_t: il tempo necessario per la conclusione di un processo.

Un buon algoritmo di scheduling deve *minimizzare* ognuna di queste quantita'. In base alla traccia in Figura 1 avremo:

- $\star$ $T_w = \frac{(0+2)+(2+1)+(2+0)}{3} = 1.4$
- $\star$ $T_r = \frac{3+2+1}{3} = 2$
- $\star$ $T_t = \frac{23+15+34}{3} = 24$

Esercizio 2

Sia data la seguente tabella che descrive il comportamento di un insieme di processi periodici

Processo	T_{start}	**CPU burst**	**IO burst**
P1	0	4	6
P2	1	2	3
P3	3	10	2

Si assuma di disporre di uno *scheduler preemptive Round Robin* (RR) con quanto di tempo $T = 5$. Si assuma inoltre che:

- l'operazione di avvio di un processo lo porti nella coda di ready, ma **non** necessariamente in esecuzione
- il termine di un I/O porti il processo che termina nella coda di ready, ma **non** necessariamente in esecuzione.

Si illustri il comportamento dello scheduler in questione nel periodo indicato.

Soluzione In base alle specifiche, avremo la traccia sara' quella illustrata in Figura 2.

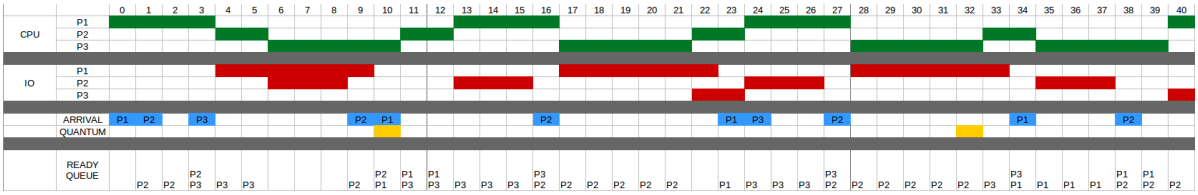

Figure 2: Traccia di esecuzione dei processi tramite RR preemptive con quanto $T = 5$.

Esercizio 3

Si consideri un Sistema Operativo batch, avente una tabella di processi di dimensione 20. Si assuma che i job durino in media 10s. Con queste premesse, ogni quanto tempo il sistema puo' accettare un nuovo job senza eccedere il numero di PCB disponibili?

Soluzione La Legge di Little permette di mettere in relazione dimensione della coda dei processi, tempo medio di esecuzione e frequenza media di arrivo degli stessi; tale formula e' descritta dalla relazione:

$$n = \lambda \cdot W \tag{1}$$

dove n indica la dimensione della coda, λ la frequenza di arrivo media e W il tempo di esecuzione medio. La soluzione e' facilmente ottenibile per sostituzione dalla (1), ottenendo una frequenza media pari a $\lambda = 2Hz$ - ovvero 2 processi al secondo.

Esercizio 4

Sia data la seguente tabella che descrive il comportamento di un insieme di processi

Processo	T_{start}	CPU burst 1	IO burst 1	CPU burst 2	IO burst 2
P1	0	5	5	3	1
P2	1	2	5	2	2
P3	5	8	1	8	1
P4	7	1	9	1	9

Si assuma di disporre di uno scheduler preemptive con quanto di tempo 5, e politica di selezione dei processi *Shortest Remaining Job First* (SRJF). Si assuma che i processi in entrata alla CPU "dichiarino" il numero di quanti necessari all' esecuzione di un CPU burst. Si assuma inoltre che:

- l'operazione di avvio di un processo lo porti nella coda di ready, ma **non** necessariamente in esecuzione
- il termine di un I/O porti il processo che termina nella coda di ready, ma **non** necessariamente in esecuzione.

Soluzione Date queste premesse, la traccia di esecuzione dei processi e' riportata nella Figura 3. Si e' assunto che nel caso vi siano piu' processi con caratteristiche uguali vada in esecuzione quello con identificativo piu' basso.

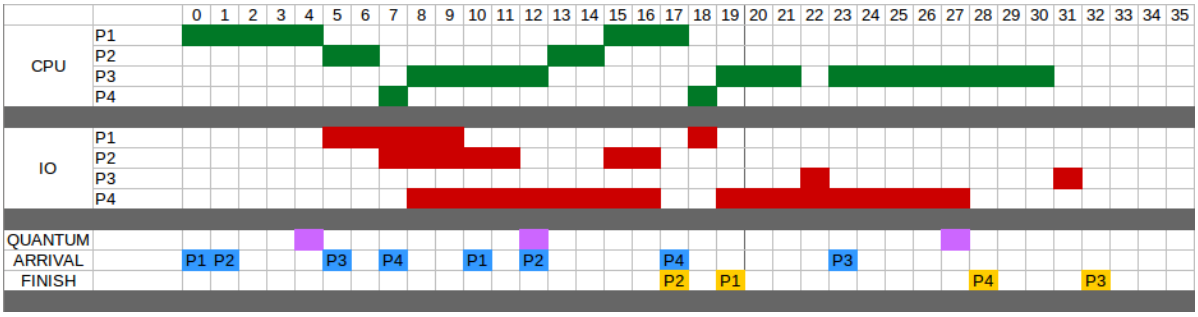

Figure 3: Traccia di esecuzione dei processi schedulati tramite *quantized SRJF*. In verde sono indicati i cicli di CPU burst e in rosso quelli di I/O; il time quantum di 5 cicli e' scandito in colore viola, l'arrivo dei processi in azzurro e la conclusione di un processo in giallo.

Esercizio 5

Sia data la seguente tabella che descrive il comportamento di un insieme di processi periodici **real-time**.

Process	T_{start}	**Period**	**CPU Burst**
P1	0	3	1
P2	0	5	2
P3	0	6	1

Si assuma di disporre di uno scheduler preemptive *Earliest Deadline First* (EDF). Si assuma inoltre che:

- la deadline di ogni processo **coincida** con il suo periodo;
- nessuno dei processi debba attendere il rilascio di una risorsa posseduta da un altro processo;
- i processi in entrata alla CPU dichiarino il numero di burst necessari al proprio completamento;
- l'operazione di avvio di un processo lo porti nella coda di ready, ma **non** necessariamente in esecuzione.

Si illustri il comportamento dello scheduler in questione nel periodo indicato, avvalendosi degli schemi di seguito riportati.

Soluzione Dato il requisito di esecuzione dei processi in *real-time* per prima cosa bisogna verificare che lo scheduler in questione possa garantire l'esecuzione di ogni ciclo di CPU burst entro la deadline specificata. Ricordando che in questo caso la deadline coincide con il periodo, calcoliamo quindi la

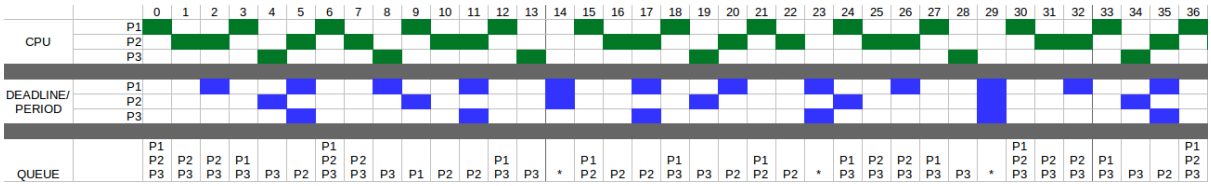

Figure 4: Traccia di esecuzione dei processi secondo l'algoritmo EDF. In blu sono scanditi i periodi dei processi, mentre in verde i cicli di CPU per ogni burst.

percentuale di utilizzo della CPU come

$$U_{CPU} = \sum_{p} \frac{t_p}{d_p} = \frac{1}{3} + \frac{2}{5} + \frac{1}{6} = 0.9. \tag{2}$$

Come si nota dalla Equazione 2, $U_{CPU} \leq 1$, quindi e' possibile effettuare lo scheduling tramite EDF rispettando il vincolo di esecuzione *real-time*. In base alle specifiche dell'algoritmo EDF, la traccia di esecuzione dei processi e' riportata in Figura 4. In questa traccia, quando due processi si trovano nelle stesse condizioni, si e' scelto di privilegiare il processo con `pid` minore.