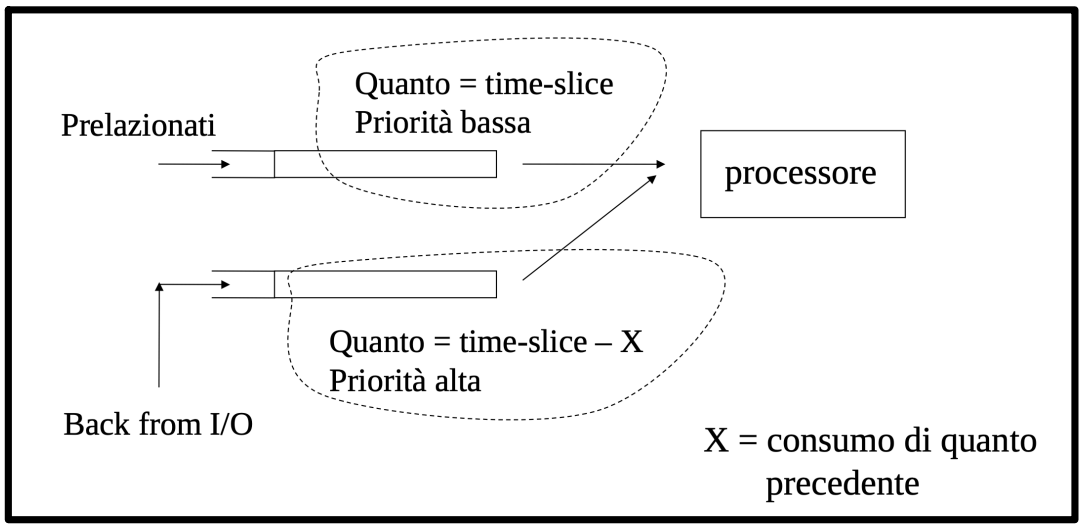
Round Robin Virtuale

martedì 11 aprile 2023 14:47

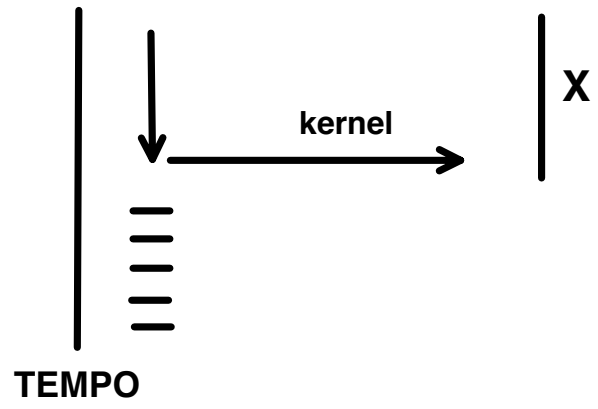
Scheduling Round-Robin Virtuale

Abbiamo una "CODA" di processi "Ready", e un'altra coda di processi "Ready". Noi siamo nello stato ready quando siamo prelazionati, ossia ho usato la CPU e poi non c'era più tempo e quindi da running sono stato riportato ready e la CPU è stata assegnata a qualche altro processo.
Il Process Control Block di questo processo che è stato riportato ready dopo aver utilizzato la CPU è collocato nella prima coda, perché è prelazionato.
Nella seconda coda quali PCB abbiamo?
Mettiamo processi che hanno subito un transito all'interno del loro diagramma erano RUNNING, hanno chiamato il kernel per andare in I/O, sono andati in blocked per un po', quando l'I/O è stato completato sono stati riportati "READY".
Quindi abbiamo due code.

Separazione tra processi prelazionati e non



La priorità è di coloro che sono nella coda "Back From I/O", perché questi hanno un comportamento che è stato in qualche modo notato essere I/O BOUND, quindi se li rimando in esercizio in CPU una volta che sono diventati pronti, PROBABILMENTE andranno di nuovo in I/O, E QUINDI MI PERMETTERANNO DI RIATTIVARE UN DISPOSITIVO DI I/O.
Questi processi ad un certo istante di tempo prendono la CPU e gliela assegno per DELTA unità di tempo, loro la rilasciano e passano il controllo al kernel chiamando il servizio di I/O. Ma per quanto tempo gli CEDO l'utilizzo della CPU? Se io voglio riparare al problema che avevo nel round robin ossia al fatto che su un processo, del DELTA di tempo assegnato, avevo usato solo una frazione di tempo e l'altra parte di DELTA pianificato per lui non era stato utilizzato, devo fargli



Non gli assegno tutto il TIME-SLICE, come invece accade per i PRELAZIONATI, ma il TIME-SLICE - X: X è quanto era già stato utilizzato DALL'ULTIMO TIME-SLICE che avevo pianificato per quel processo che era entrato in stato "READY".

Non sfavoriamo più i processi I/O bound, come invece nella pagina precedente succedeva.
Ma tra tutti i processi BACK FROM I/O, ce n'è uno più importante? È anche quello un problema da gestire.
Ma tra tutti i prelazionati, c'è qualcosa di più importante? Anche quello dobbiamo sapere.
Devo distinguerle.

Il RR è particolarmente adatto per il Time Sharing. Il comportamento del RR dipende molto dal valore del *quanto di tempo q* scelto: se q tendente a infinito rende RR uguale a FCFS, se q tendente a zero produce un maggior effetto di "parallelismo virtuale" tra i processi però aumenta il numero di context switch, e quindi l'overhead. L' efficenza non migliora necessariamente se si aumenta il quanto q, ma può migliorare se la maggior parte dei processi ha un burst minore di q.