# Sistemi 8

Abbiamo visto il nuovo Laboratorio. Il file da completare si chiama microbash.c e deve diventare una versione “giocattolo” della shell. Non dovrebbero servire librerie di terze parti, escluse quelle già presenti.

Lagorio è un cazzo di Chad.

Ritorniamo ai nostri algoritmi di scheduling. Vediamo il Multi-Level FeedBack Queue, questo algoritmo tenta di ottimizzare sia il turnaround time (facendo eseguire prima i job corti) sia il response time. Tuttavia uno scheduler non può sapere in anticipo quale sarà la durata complessiva di un processo, quindi come fa ad imparare le sue caratteristiche e predire il futuro?  
I processi veri in alcuni momenti si comportano CPU-Bound e in altri si comportano usando principalmente periferiche di I/O. Per questo motivo il MFLQ ha diverse code a differente priorità, su una stessa coda viene effettuato il Round Robin dei processi presenti, e la priorità assegnata a ciascuna coda dipende dal comportamento dei processi in essa presenti.

Se p(a) > p(b), gira A e non B (p = priorità).

Se p(a) = p(b), A e B girano in Round Robin.

Con solo queste due regole c’è starvation per tutti i processi a priorità più bassa di A e B.

Vengono quindi aggiunte le regole:

Un nuovo job arriva con la priorità massima (ciò abbassa il tempo di risposta; inoltre, se questo è corto, termina subito, dandoci un turnaround fantastico).

Se il nuovo job si usa tutto il suo quanto di tempo (senza fare syscall di I/O) il processo diventa “CPU-bound” e la sua priorità viene ridotta.

Un problema è che un job potrebbe “barare”: basterebbe che continuasse a usare la cpu per il 99% del suo quanto di tempo e alla fine facesse input output, per poi tornare alla CPU, rimanendo così ad alta priorità. Inoltre, jobs che finiscono a priorità bassa potrebbero rimanere ingiustamente declassati per troppo tempo anche se vorrebbero fare operazioni di I/O.

Aggiungiamo quindi la regola che:

Ogni s secondi, spostiamo tutti i job alla priorità più alta; questo dovrebbe prevenire la starvation, ma non impedisce a un processo di continuare a barare. Per ovviare a questo basta tenere conto del tempo TOTALE t in cui un processo ha usato la CPU (quindi non si dà un quanto resettabile se si fanno operazioni di I/O, quando l’I/O termina il tempo “nuovo” di utilizzo della CPU viene sommato a quello precedente), al termine del quale la sua priorità viene ridotta.

L’unica problematica è che i parametri finora definiti non possono essere facilmente impostati automaticamente (quante code, quanto vale t, quanto s?). Una soluzione possibile sarebbe lavarcene le mani e lasciare la responsabilità in mano all’amministratore di sistema, ma ci sono altri approcci che vedremo successivamente.