

Scheduling UNIX tradizionale (SVR3 – 4.3 BSD)

Questa è una schedulazione che è tutt'ora presente nelle release di tipo UNIX.

Dobbiamo gestire code multiple con feedback. Con feedback significa che a seconda di quello che sta succedendo noi processi possiamo cambiare coda. E quindi i processi possono scendere nei livelli di priorità ma eventualmente anche risalire.

Ovviamente abbiamo un livello di priorità distinto per ciascuna coda. Per assegnare la CPU all'interno di ciascuna coda vi è una gestione di tipo Round Robin (RR). Ovviamente il carosello è basato sul quanto di tempo.

Passaggio da una coda all'altra (feedback)

Se noi rientriamo nello stato “ready” dopo un passaggio nello stato “sleep” questo ci provoca un cambio del livello di priorità.

QUEUE 0

Veniamo prelevati dalla coda, ci viene assegnata la CPU, veniamo eseguiti, poi veniamo passati in uno stato di WAIT e poi ritorniamo in stato READY. Magari avevamo chiesto al sistema operativo di attivare una richiesta di I/O avevamo aspettato che il dispositivo completasse le sue operazioni e quando veniamo messi ready non rientriamo nella coda dove originariamente eravamo, ma rientriamo in un'altra.

ESECUZIONE

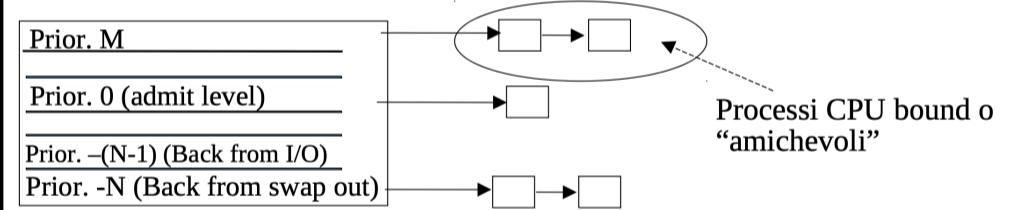
QUEUE 1

Quando nello stato di blocco poi vado a transitare in uno stato ready, mi viene assegnata una priorità differente. Questo è un feedback.

Qui abbiamo l'insieme delle possibili priorità che abbiamo all'interno dello scheduler Unix tradizionale.

Queste priorità vanno da +M fino a -n. Lo zero è un livello intermedio e abbiamo poi dei livelli superiori allo zero e dei livelli inferiori allo zero.

Quando abbiamo un livello di priorità negativo abbiamo un livello di priorità migliore dello zero. Quando abbiamo un livello di priorità positivo abbiamo un livello di priorità peggiore dello zero.



Supponiamo che avevamo un processo che aveva preso la CPU, aveva chiamato il kernel per andare in I/O, e quindi era andato in blocco, e poi era stato riportato ready, dove rientriamo come livello di priorità? Al livello -(N-1)! Queste sono tutte le applicazioni che sono "back from I/O".

I livelli sono 40.

Addirittura c'è una coda ancora più prioritaria, al livello -N, che è destinata ad applicazioni che non erano in memoria, erano state migrate fuori dalla memoria, vengono ora riportate all'interno della memoria e giustamente a queste applicazioni dovremmo riassegnare la CPU il prima possibile.

Ma se ero nello stato running, finisce il quanto di tempo, e vengo riassegnato all'interno dello stato “ready”. In questo momento la CPU è assegnata a qualche d'un altro. Qual'è la priorità che mi viene data qui? Si assegna la cosiddetta priorità di riferimento per quel processo. E c'è una formula per calcolarla.

$$P = \text{base} + f(\text{CPU usage}) + \text{nice}$$

La base tipicamente è zero.

Se il nice è 0, la priorità è quella a livello 0.

Se nice è M noi abbiamo una priorità che è il primo livello, e quindi implica dire che se noi da "running->read", veniamo messi al livello M, cioè abbiamo una probabilità inferiore di riuscire a riprendere la CPU rispetto ad altri processi che eventualmente sono "ready".

Il valore del nice può anche essere negativo.

Tutto questo ipotizzando che la CPU usage è tendente a zero.

Tutto questo dipende dal valore della "NICE".

Il valore di *nice* è un attributo numerico di ciascun processo dei sistemi Unix e Unix-like che è usato dallo **scheduler** per stabilire quanto tempo di CPU dedicare all'esecuzione del processo.
A parità di priorità e di politica di **schedulazione**, i processi che hanno valori di *nice* maggiori ottengono in proporzione meno tempo di CPU rispetto a processi che hanno valori di *nice* minori, e quindi la loro esecuzione procede più lentamente, favorendo gli altri processi.

nice è anche il nome di una **chiamata di sistema** definita dallo standard **POSIX⁽¹⁾** che modifica il valore di *nice* del **processo** che la invoca. Di fatto il comando *nice* opera invocando l'omonima chiamata di sistema.

Per diminuire il valore di *nice* è necessario disporre dei privilegi dell'amministratore (**root**), mentre ciò non è necessario per aumentarlo. Tipicamente è possibile diminuire il valore di *nice* fino a 20 unità rispetto al valore predefinito, o aumentarlo fino a 19 unità: è possibile specificare scostamenti più ampi, ma essi sono automaticamente ricondotti entro i limiti sopra citati.

F(CPU Usage) è il tempo di CPU per quel processo. Quindi il valore della priorità assegnata può cambiare se questo processo sta utilizzando la CPU di più o di meno.

Se il tempo di CPU che un processo ha utilizzato è molto alto, chiaramente questo addendo nell'equazione, contribuisce in maniera significativa, e quindi la priorità ha un fattore positivo che fa sì che la priorità stessa di quel processo sale all'interno dello schema.

Ma salendo all'interno dello schema, la sua priorità VERA è peggiore.

Il nice può anch'esso cambiare: non viene cambiata autonomamente dal software del sistema, ma viene cambiata perché abbiamo le System Call per cambiarla.

Ogni processo può chiedere al sistema di cambiare la sua stessa NICE: SE NOI ABBIAMO UN PROCESSO ATTIVO CON NICE PARI A X, UN ALTRO PROCESSO ATTIVO P' PUÒ CHIEDERE AL KERNEL DI ANDARE A CAMBIARE QUESTA X IN Y.

Se andiamo I/O bound ed andiamo ad essere bloccati la nostra priorità va sotto fino a che prendo la CPU , poi quando mi viene assegnata la CPU se uso tutto il quanto di tempo che mi è stato dato, verrò riportato al mio livello originario. Però se non lo faccio e vado di nuovo in blocco, la prossima volta che mi sblocco sono ancora qui a -(N-1). Invece se divento CPU Bound e quindi cambio fase viene usata la priorità di riferimento e, in questa, sono in grado di assegnare un valore NICE, e oltretutto il sistema fa da solo una rappresentazione a grana fine basata anche sull'uso della CPU, ossia F(Cpu Usage).

Periodicamente si va ad osservare, nell'ultimo periodo, la CPU utilizzata.

... tale scheduler di CPU e' tutt'ora riadattato per sistemi orientati ai threads ...

