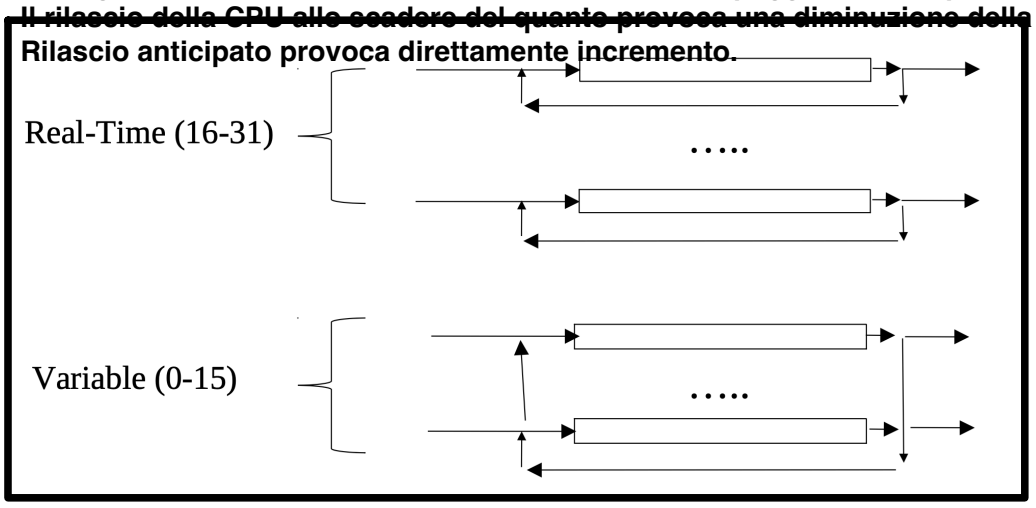
Scheduling in sistemi Windows

Nello scheduling windows abbiamo 32 code multiple di priorità distinte in due fasce: Fascia Real-Time e la fascia Variable.
Il livello di priorità è distinto per ciascuna coda. Da 0 a 15 siamo nella fascia variable, da 16 a 31 siamo nella fascia real-time.
All'interno di ciascuna di queste code c'è una gestione di tipo round robin.
Ogni processo ha una sua priorità base e i thread hanno una priorità dinamica (entro certi vincoli) che è in funzione della priorità base del processo di cui fanno parte.

Il passaggio da una coda all'altra (feedback) non è ammesso nella fascia real-time, un thread real time non cambierà mai il suo livello di priorità indipendentemente da quello che ha fatto.
Ogni volta che rientra ready a prendere il controllo della CPU, entrerà sempre in quello specifico livello di priorità.
Solo tramite system call possiamo spostare un thread nelle code della fascia real-time, oppure spostarlo in una coda nella fascia variable e una volta spostato lì quel thread sarà autonomo.

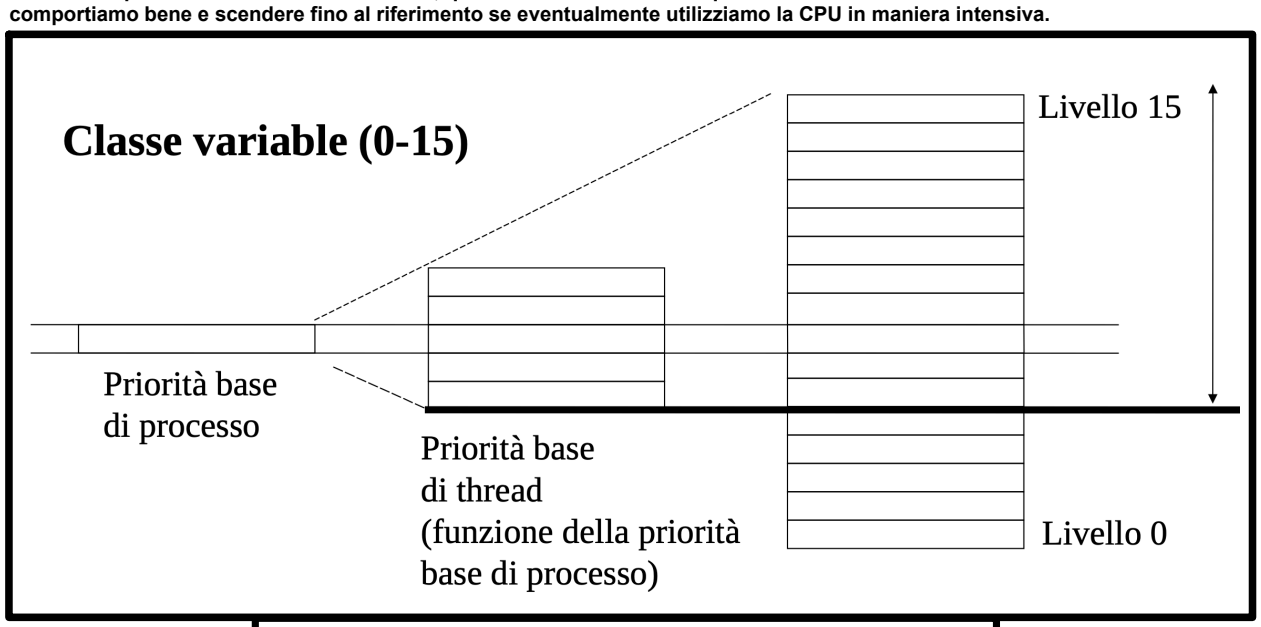
Il passaggio da una coda all'altra (feedback) è ammesso all'interno della fascia variable, un thread variable può cambiare il suo livello di priorità, migliorarla o peggiorarla, ma sempre rimanendo dentro la fascia variable.
La migliorerà se lascerà la CPU e andrà in blocco, la peggiorerà se tipicamente utilizzerà tutto il quanto.
Il rilascio della CPU allo scadere del quanto provoca una diminuzione della priorità.
Rilascio anticipato provoca direttamente incremento.



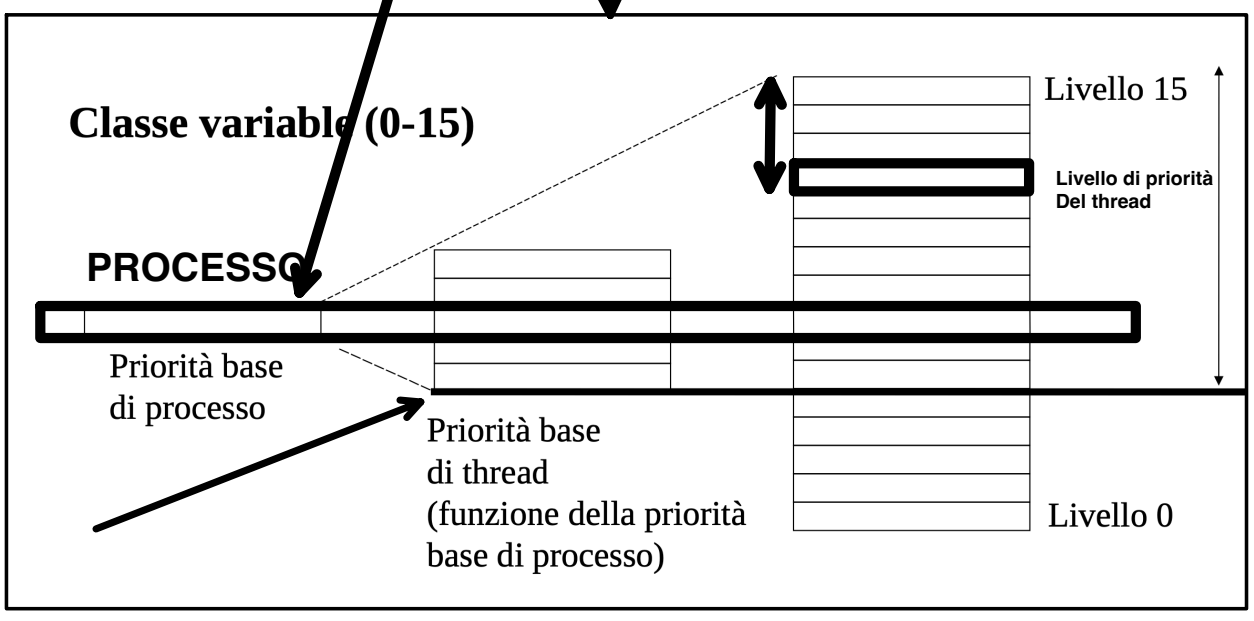
Se un thread è nella fascia real-time, ad esempio nel livello di priorità 16 (ossia il livello più basso), quando rientra ready rientra sempre nello stesso livello di priorità.

Mentre invece nella fascia variable, se siamo originariamente al livello 0 e rientriamo nello stato ready, possiamo salire piano piano all'interno di questa fascia variable.
Il time-sharing tradizionale è rappresentato all'interno della parte variable.

Qui c'è una differenza in base alla priorità che noi associamo ad un processo e alla priorità che noi associamo al thread di quel processo.
In CPU non ci vanno i processi, anzi, ci vanno i thread in un sistema Multi-Threading.
Però quando lanciamo un processo possiamo assegnargli una "Priorità Base", essa serve poi per andare a definire quali sono le priorità dei thread di quel processo.
Se al processo abbiamo assegnato una certa priorità, al thread possiamo assegnare una priorità di riferimento che è due livelli sotto a quella del processo (non necessariamente deve essere due).
Quando attribuiamo la priorità al thread ci riferiamo ad un riferimento rispetto alla priorità del processo.
Questa è la priorità di riferimento del thread, questo vuol dire che sotto quel livello non ci scendiamo ed eventualmente se siamo nella classe variable possiamo salire se ci comportiamo bene e scendere fino al riferimento se eventualmente utilizziamo la CPU in maniera intensiva.



Chiaramente quando noi lanciamo i vari thread delle applicazioni windows questo livello di priorità di riferimento di livello dei thread, può essere differente: possiamo avere un processo qui ed un livello di priorità che specifica che siamo più su rispetto alla priorità di riferimento.



Andiamo a vedere come si fa a settare la priorità di un processo. Abbiamo una system call che si chiama "SetPriorityClass", e possiamo specificare il livello di priorità che assegniamo a quel processo: se cambiamo il processo da una priorità all'altra chiamando quella system call, stiamo cambiando anche la priorità dei vari thread di quel processo, perché la priorità dei vari thread è riferita alla priorità del processo. Quindi ora siamo +2 rispetto alla nuova priorità.
Ovviamente però ai vari thread possiamo cambiare la priorità con "SetThreadPriority": Un processo ha una priorità L, il thread ha un certo livello di priorità relativo ad L e vogliamo cambiare questo livello relativo ad L, e quindi portarci più su o più giù rispetto rispetto al riferimento L che stiamo utilizzando.

Ovviamente, è necessario tenere in considerazione che Windows nasce come sistema multi-threading, e ad essere schedulati ed eseguiti dalle CPU sono, ovviamente, i threads. Troviamo delle code di priorità; all'interno della singola coda, la gestione è di tipo Round Robin. Ciò che cambia è la gestione delle priorità. In Windows troviamo infatti due fasce di priorità, la prima è detta variable [0;15], mentre la seconda è detta real-time [16,31]. Ovviamente, il livello più prioritario è il 31, ed il meno prioritario è lo 0. È possibile spostarsi all'interno delle varie code di priorità, ma sotto determinate condizioni:

1) I threads nella fascia real-time sono destinati a rimanere per sempre nella coda cui erano stati originariamente destinati.

2) Al processo generatore è associata una priorità di base. Tale priorità non è volta alla schedulazione del processo (Windows schedula ed esegue solamente threads), quanto fornire un vincolo di movimento per i threads generati dal processo; infatti, i processi nella fascia variable possono muoversi da 0 a 15, ma questo è possibile solamente in un range deciso dalla priorità del processo padre. Tale priorità del processo padre può essere modificata mediante la syscall SetPriorityClass;

3) La Priorità di un thread può cambiare (sullo spunto di ciò che accadeva in Multi-level-feedback-queuing) in base al tempo di utilizzo della CPU. Se il thread non utilizza tutto il quanto di tempo concessogli (ricordiamo che la gestione all'interno della singola coda di priorità è di tipo Round Robin, quindi è previsto un quanto di tempo) allora la sua priorità si alza, altrimenti si abbassa.
In sistemi multithreading vi sono altre facility interessanti che permettono di settare quella che viene chiama "affinità" per ogni singolo thread. Tramite la syscall SetThreadAffinity si può infatti fornire una maschera di bit per indicare al thread su quali CPU può entrare e quali invece gli sono precluse, controllo capillare necessario quando si sta parlando di sistemi multiprocessore.