

# Multilevel Feedback Queue

Algoritmo alla base dello scheduling della maggior parte dei sistemi operativi moderni, non è ottimo per nessuna metrica ma ha buona reattività, overhead, fairness e evita starvation.

È un'estensione del RR con più code, ciascuna con un livello di priorità e quanto di tempo diverso – maggiore la priorità, minore il quanto. Se un task termina il suo quanto scende di priorità, se si interrompe prima (e.g. per richiesta di I/O) sale. Questo fa sì che i task CPU-bound abbiano priorità bassa, viceversa quelli I/O-bound. Il task da eseguire viene estratto dalla coda non vuota con priorità maggiore.

Il sistema monitora i processi per assicurarsi che ricevano una quantità equa di risorse (per impedire che i task CPU-bound non vengano eseguiti se ci sono troppi task I/O-bound), e che i processi non aumentino la propria priorità effettuando sempre operazioni di I/O subito prima della scadenza del quanto.

## Windows

- 32 livelli di priorità, più -1 per i processi idle (sleep della CPU). I processi di sistema hanno sempre priorità più alta (16-31) di quelli utente (0-15). I processi utente svolgono diverse operazioni tramite quelli di sistema, quindi è conveniente per tutti che questi siano più reattivi;
- l'incremento della priorità a seguito di operazioni di I/O dipende dal tipo di dispositivo, per esempio:
  - dischi: +1, lavora a blocchi e non è urgente;
  - dispositivi seriali: +6;
  - schede audio: +8, richiede bassa latenza;
- se un thread non è in esecuzione da un po' di tempo, +15 per 2 quanti (senza comunque superare la barriera). Serve per limitare situazioni di *inversione di priorità*: A>B>C, A è in attesa che C rilasci una risorsa, ma lo scheduling favorisce B, che quindi sta impedendo l'esecuzione di un processo con priorità maggiore;
- ogni volta che si consuma il quanto di tempo, la priorità scende di 1;
- i processi associati a finestre visibili hanno priorità maggiore.