Gestione dei process

Il processo

In un moderno sistema sono caricati in memoria diversi programmi che sono eseguiti contemporaneamente. I programmi in esecuzione devono essere descritti e la loro evoluzione deve essere controllata.

Il processo è lo strumento usato per descrivere un programma in esecuzione.

Per descrivere i programmi in esecuzione si usano diversi termini:

* job, task degli utenti, task del sistema operativo (sinonimi di processo).

Un intero sistema è descritto come un insieme di processi che possono essere eseguiti in parallelo sui processori e il sistema operativo ne alterna l’esecuzione.

L'alternarsi delle esecuzioni avviene per effetto:

* di azioni generate dagli stessi processi: un programma in esecuzione si ferma in attesa di ricevere dati da un'unità periferica.Un processo che si è fermato, perché aspetta i dati da un'unità periferica, riprenderà l’esecuzione dopo che i dati sono stati trasferiti in memoria.
* di azioni esterne al processo: un segnale inviato dal timer per interrompere un processo una volta scaduto il quanto di tempo assegnato.

Le interruzioni

Il processore riesce a eseguire molti programmi contemporaneamente alternandone l'esecuzione nel tempo. L'alternarsi delle esecuzioni avviene secondo diverse modalità e per differenti cause:

* il sistema operativo interrompe l'esecuzione di un programma a bassa priorità per assegnare il processore a un programma a priorità più elevata;
* nel caso dei sistemi Urne sharing, allo scadere di una quantità prefissata di tempo, il programma in esecuzione è interrotto per cedere il controllo della CPU (processore) ad un altro programma;
* un programma in esecuzione che deve leggere dei dati da una periferica richiede l'intervento del sistema operativo che avvia l’operazione di lettura di un altro programma;
* un evento esterno al sistema causa l’interruzione del programma in esecuzione, permettendo al sistema operativo di intervenire per attuare le azioni necessarie. Per esempio, un sensore collegato alla porta di una camera d’albergo rileva l'apertura della stanza, il televisore della camera si deve accendere e sullo schermo appare un messaggio personalizzato di benvenuto al nuovo ospite.

L'esecuzione delle istruzioni macchina può essere descritta con il seguente schema:



Il processore, dopo avere eseguito un'istruzione, preleva la prossima istruzione, la esegue e poi ripete il ciclo. Lo schema è corretto solo se le interruzioni sono disabilitate. Quando le interruzioni sono abilitate dopo avere eseguito un'istruzione, il processore controlli se ci sono segnali di interruzione pendenti. In caso affermativo, il processore:

* prima gestisce l’interruzione trasferendo il controllo a una routine di gestione dell'interruzione;
* successivamente, passa a eseguire la prossima istruzione.

L’abilitazione è gestita dal sistema operativo che modifica il valore di un apposito flag memorizzato nella PSW (Program Status Word).

I computer gestiscono l'interruzione in modo analogo alle persone quando devono interrompere la propria attività per rispondere a un segnale di allerta.

Un programma per la gestione dell’interruzione, indicato con la sigla ISR (Interrupt Service Routine), deve:

1. salvare lo stato del programma in esecuzione per farlo ripartire al termine dell'interruzione;
2. identificare chi ha inviato la richiesta di interruzione e quindi il tipo di servizio richiesto;
3. trasferire il controllo alla routine specializzata nella gestione del servizio richiesto.

Le interruzioni possibili sono in numero predefinito. Per velocizzare le operazioni di gestione dell’interruzione si usa un vettore delle interruzioni (interrupt vector).

Il vettore delle interruzioni è in un insieme di celle di memoria contigue прилежащие, ognuna delle quali contiene l'entry point di una routine specializzata in un servizio.

Ad ogni dispositivo è associato un valore numerico che viene usato come indice per accedere al vettore delle interruzioni e far partire la routine in grado di servire le interruzioni di quel dispositivo (ISR del dispositivo). Questa tecnica è usata da molti sistemi operativi tra i quali Dos e Unix/Linux.

Un esempio il caso di un CD-ROM, identificato come dispositivo 32, che invia un segnale di interruzione per informare di essere pronto a trasmettere un blocco di dati:

1. Il CD-ROM memorizza il blocco di dati nel buffer del controllore del dispositivo e invia un segnale di interruzione al processore.
2. Il processore completa l'esecuzione dell'istruzione corrente e poi esamina il registro delle interruzioni per identificare il dispositivo che ha causato l’interruzione. Invia un segnale di risposta al dispositivo per informarlo che l'interruzione è stata riconosciuta.
3. Il processore usa il numero del dispositivo per accedere alla posizione 32 del vettore delle interruzioni.
4. Il processore, prima di trasferire il controllo alla Routine 32, deve salvare lo stato del programma in esecuzione.
5. Il processore carica nel Program Counter l'indirizzo che ha prelevato dal vettore delle interruzioni avviando così l'esecuzione della Routine 32.
6. Il processore esegue la ISR del CD che contiene le istruzioni necessarie per elaborare l'interruzione. Nel caso specifico il processore trasferisce i dati dal controllore del CD alla memoria centrale e poi avverte il controllore di avere svuotato il buffer, che potrà quindi essere riempito con altri dati.
7. Completata l'elaborazione dell'interrupt, il programma interrotto deve ripartire.

Il punto 7 è corretto solo se c’è un solo programma in esecuzione. In un sistema multi programmato dopo avere ricevuto i dati inviati dal CD, il programma che li attende è pronto a ripartire. La decisione se far ripartire il programma appena interrotto dipende dalle strategie del sistema operativo quale programma far ripartire.

Programmi e processi

Un programma in linguaggio macchina pronto per l'esecuzione (un insieme di bit collocati in un file su disco) è un'entità statica:

* non cambia nel tempo,
* non modifica lo stato della memoria,
* non modifica lo stato del Program Counter
* non modifica lo stato dei registri;

Un processo è un'entità dinamica che viene creata, passa attraverso diversi stati e, alla fine, è eliminata dal sistema:

* l'esecuzione di un processo modifica lo stato della memoria e dei registri.
* Un processo ha una traccia di esecuzione, che può essere definita come la sequenza di stati assunti dal processore durante l’esecuzione del processo.
* A un processo sono associati il codice, un’area dati, la memoria, i processori, i file, i dispositivi di I/O.

La traccia di esecuzione è la sequenza delle istruzioni eseguite e quindi alla sequenza di valori assunti dal Program Counter.

Un processo è un programma in esecuzione. Allo stesso programma possono essere associati più processi: in un sistema utenti mandano in esecuzione contemporaneamente la stessa applicazione (l’automobile sia il programma e il viaggio in automobile sia il processo).

Stati di un processo e transizioni di stato

La sequenza di stati che assume di un processo:

1. Facendo clic su un'icona si richiede di mandare in esecuzione un programma. Il sistema operativo crea la struttura dati che rappresenta il processo: il processo si trova nello stato new (nuovo).
2. Il sistema operativo assegna al processo le risorse richieste e lo colloca in memoria: si dice che il processo si trova nello stato ready (pronto). Un processo nello stato ready è un processo che può essere eseguito e attende l'assegnazione di un processore.
3. Quando il sistema operativo decide di eseguire il processo assegnando ad esso il processore, il processo passa nello stato running (esecuzione).

Un processo creato, messo in memoria e mandato in esecuzione, si trova nello stato running.

In un sistema con un solo processore ci può essere un solo processo running, ma ci possono essere molti processi ready in memoria.

L'evoluzione del processo può seguire tre strade diverse:

* Il processo completa la sua esecuzione e cede il controllo al sistema operativo che lo mette nello stato terminated (terminazione): il processo abbandona il sistema.
* Il tempo assegnato al processo scade e l'orologio di sistema invia un segnale di interruzione. Il sistema operativo mette il processo interrotto nello stato ready e seleziona uno dei processi ready per l'esecuzione.
* Il processo esegue un'operazione di Input/Output (I/O) e deve attendere il completamento. Il sistema operativo attiva l'operazione di I/O, mette il processo nello stato waiting (attesa) e seleziona uno dei processi ready per l'esecuzione. Un processo nello stato waiting è un processo bloccato in attesa di un evento. Al completamento dell'operazione di I/O, il dispositivo invia un segnale di interruzione che provoca l'intervento del sistema operativo. Il sistema operativo, poiché i dati richiesti sono ora disponibili, sposta il processo allo stato ready.

Più programmi sono presenti in memoria, minore è la probabilità che siano tutti bloccati in operazioni di I/O.

Un processo passa diverse volte negli stati running, ready e waiting. In particolare si possono individuare due cicli:

* un ciclo causato dallo scadere dei quantum temporali: running -> ready -> running
* un ciclo dal processo di lettura e scrittura: running —> waiting -> ready -> running

Le transizioni di stato di un processo sono:

* Ammissione è la transizione da new a ready ed è provocata da un utente (o da un altro processo).
* Assegnazione è la transizione da ready a running, assegna un processore a un processo ed è eseguita dal sistema operativo.
* Interruzione è la transizione da running a ready ed è causata da un’interruzione. Il processo potrebbe proseguire, ma viene interrotto da un evento esterno al processo: un segnale inviato dall'orologio.
* Attesa evento è la transizione da running a waiting ed è causata dal processo stesso che, per eseguire un’operazione di I/O, provoca l’intervento del sistema operativo.
* Avviene evento è la transizione da waiting a ready ed è causata da un evento esterno al processo, per esempio il completamento di un'operazione di Input/Output.

L'insieme dei processi ready, running e waiting è l'insieme dei processi attivi.

Le strutture per gestire i processi

I processi sono gestiti dal sistema operativo hanno l'obiettivo di ottimizzare l’uso delle risorse del sistema. Quando un processo entra nel sistema nello stato new, il sistema operativo crea la struttura dati.

Nella struttura dati devono essere tutte le informazioni necessarie:

* del suo stato di esecuzione
* delle risorse.

A un processo sono associati:

* il codice eseguibile,
* lo spazio di memoria con i dati e ulteriore spazio in memoria per lo stack
* il PCB (Process Control Block, blocco di controllo del processo).

Il PCB è la struttura dati con le informazioni necessarie alla gestione del processo.

In un PCB ci sono:

* gli identificatori del processo,
* alcuni puntatori per collegare i processi nel sistema,
* la PSW con le informazioni sullo stato del processore
* le informazioni per descrivere lo stato del processo.

Puntatori sono collegati tra di loro per costruire:

* la lista dei processi in un certo stato (ready, waiting, ecc.),
* la lista dei processi in attesa di uno specifico evento,
* la lista dei processi con la medesima priorità,
* le liste per collegare processi che stanno tra di loro in una relazione gerarchica di genitore-figlio.

Identificatori: il sistema assegna ad un processo un identificatore, in genere un numero e specifica l'utente associato al processo.

La PSW contiene le informazioni necessarie per far ripartire un processo dopo averlo interrotto:

* il contenuto dei registri utente,
* il valore del Program Counter
* degli altri registri del processore (come il registro EFLAGS del processore Pentium nei sistemi operativi Windows e Unix/Linux.

Stato del processo contiene le principali informazioni(new, ready, waiting, ecc.) necessarie all’esecuzione di un processo. Tra queste:

* il valore del quantum assegnato
* l'ora di attivazione.
* le informazioni sulle risorse collegate al processo: i file aperti, la memoria assegnata.
1. Un processo ready è quando un processo è messo in memoria e gli sono state assegnate le risorse di cui necessita.
2. Lo stato running quando il processo ottiene un processore.

Un processo nel corso della sua vita passa diverse volte attraverso gli stati waiting, ready, running prima di arrivare al suo completamento.

Il sistema operativo, per sapere lo stato dei processi e attuare le opportune transizioni, i processi inserisce in code distinte in base al loro stato. La figura seguente mostra alcuni processi inseriti in tre differenti code: una per i processi pronti e due code per i processi bloccati.

Le code sono state realizzate collegando mediante puntatori.

La quantità di code e la loro struttura è del tutto indicativa. Per esempio, ci potrebbe essere, per esempio, una coda di processi in attesa di dati dal disco, una seconda coda per le operazioni di I/O su nastro magnetico e così via.

Ci possono essere anche altre code. Per esempio la coda dei processi sospesi e, nel caso di un sistema multiprocessore, la coda dei processi running.

Anche la struttura delle code può essere differente: i processi in una coda possono essere collegati con una doppia catena di puntatori per ragioni di sicurezza ed efficienza.

Un processo in esecuzione lascia il processore (CPU) per diverse ragioni e ha comportamenti differenti:

* se il processo rilascia il processore per la fine del time che gli è stato assegnato, viene immediatamente messo nella coda dei processi pronti,
* se il processo lascia il processore in attesa di un evento, è inserito in una coda di processi in attesa di quello specifico evento ed è trasferito nella coda dei processi pronti quando accade l’evento.

Il sistema operativo deve gestire le risorse del computer. Le risorse gestite sono:

* i processori,
* la memoria centrale,
* i file
* i dispositivi periferici.

Il sistema operativo per ragioni di sicurezza ha bisogno di :

* conoscere lo stato di tutte le risorse per poterle allocare ai processi.
* Limitare e controlare la comunicazione tra processi: i registri limite servono a controllare che un processo acceda solamente al proprio spazio di memoria.
* in certi casi la communicazione tra i processi: il sistema operativo mette a disposizione sistemi per lo scambio di informazioni tra processi (si usano i servizi di IPC (Inter Process Communication,) che permettono lo scambio di informazioni tra processi per mezzo di messaggi).

I thread

Un processo fosse associato un solo flusso di esecuzione.

Il flusso di esecuzione, in inglese thread è dato dalla sequenza di valori assunti dal Program Counter durante l'esecuzione del processo.

L’esistenza di un solo thread in un processo significa che al processo può essere assegnato un solo processore. I moderni sistemi operativi sono in grado di gestire processi con più flussi di esecuzione e, se il sistema ha più di un processore, i thread possono essere eseguiti in parallelo su differenti processori, rendendo più veloce l'esecuzione delle applicazioni.

Con un solo thread, in memoria ci sono:

* il codice del programma,
* i dati,
* lo stack,
* il PCB,
* le informazioni contenute nella PSW dell'unico thread di esecuzione. In particolate nella PSW ci sono i registri del processore e il Prneram Counter.

Un processo che incapsula tre thread:

* I thread condividono spazio di memoria e file.
* in comune ai tre thread, ci sono il codice, i dati, il PCB e lo spazio per la gestione dei file, le informazioni necessarie per gestire i tre flussi di esecuzione.
* I tre thread hanno ciascuno il proprio: Program Counter, propri registri e stack. Le informazioni contenute nel Program Counter, nei registri e nello stack permettono di tenere conto dello stato di esecuzione del singolo flusso.

Ci sono diversi vantaggi nell'approccio multithreading:

* Se un'applicazione è sviluppata usando il multithreading può essere eseguita più rapidamente: in un programma di word processor:
1. un thread legge l'input dell'utente,
2. un secondo thread si preoccupa di formattare e visualizzare il testo introdotto.
3. Un terzo thread potrebbe poi gestire i salvataggi periodici del testo introdotto.
* L'esecuzione dei tre flussi è più veloce anche nel caso di sistemi a singolo processore, dove l'esecuzione dei thread è ottenuto grazie al parallelismo. Il passaggio dall'esecuzione di un thread a un altro è più veloce del cambiamento di contesto tra processi.
* I thread possono collaborare facilmente nell'esecuzione dei compiti richiesti dall'applicazione. I thread condividono lo spazio di memoria. Differenti thread possono scambiarsi informazioni facilmente, a differenza dei processi che operano ciascuno nel proprio spazio di memoria.
* Con i thread la progettazione del software può essere migliorata per la semplificazione concettuale dei compiti del sistema complessivo.

Per passare dall'esecuzione di un thread di un dato processo a quella di un altro thread del medesimo processo, le informazioni da salvare sono di meno e si limitano a quelle riguardanti lo stato di esecuzione del singolo thread e non di tutte quelle che riguardano l’intero processo. Cosi non si ha la necessità di spostare i PCB dei processi da una coda a un'altra. Cosi il cambiamento di contesto tra thread sia dieci volte più veloce del cambiamento di contesto tra processi.

I thread sono spesso indicati con il nome di processi leggeri. I thread sono processi fidati perché, appartenendo allo stesso processo, sono autorizzati a condividere informazioni. Questo fatto è un elemento di forza dei thread, è anche elemento di debolezza a causa dei problemi nella reciproca interazione. Si pensi al caso del programma word processor: Consideriamo il thread L che legge l’input dell'utente e il thread F che formatta e visualizza il testo introdotto. Il progettista del software deve riuscire a dare risposta a domande del tipo:

* Che cosa succede se L ed F cercano di accedere contemporaneamente ai medesimi dati?
* Come si possono sincronizzare le attività L ed F per evitare confusione con i dati?

Schedulazione dei processi

Il sistema operativo ottimizza l'esecuzione: I processi passano da uno stato all'altro e in alcuni casi il sistema operativo decide quali processi cambiano di stato e quali no. Questo avviene di solito durante la transizione:

* da ready a running, quando viene assegnarmento il processore.
* da sospeso ad attivo.

Lo schedulatore è la componente del sistema operativo che implementa oсуществляет gestione dei processi eseguendo le scelte dei passagi dei processi da un stato all'altro.

Ci sono diversi tipi di schedulazione caratterizzate dalla frequenza con cui le scelte di schedulazione sono attuate:

* La schedulazione a lungo termine è quella eseguita meno frequentemente.
* la schedulazione a medio termine è quella eseguita più frequentemente
* La schedulazione a breve termine è quella eseguita la più frequentemente

La schedulazione a lungo termine riguarda la transizione dallo stato new allo stato ready nei sistemi batch (controlla il grado di multiprogrammazione del sistema) È assente nei sistemi come Unix/Linux. In questi sistemi un parte di lavori è messo su disco nello stato sospeso.

I processi sospesi (i processi dormienti), sono quei processi che sono stati rimossi dalla memoria centrale e salvati sul disco. Lo schedulatore decide quali di questi lavori possono essere caricati in memoria e diventare ready.

La schedulazione a medio termine riguarda invece le transizioni tra pronto e sospeso e tra waiting e sospeso. Lo schedulatore decide quali processi scaricare su disco (swap out) liberando la memoria e, viceversa, quali processi riportare in memoria (swap in). Anche lo schedulatore a medio termine influisce sul grado di multiprogrammazione del sistema.

La schedulazione a breve termine riguarda le transizioni tra ready e running. Questo tipo di schedulazione è eseguita molto frequentemente perché lo schedulatore deve intervenire ogni volta che un processo rilascia il processore. Questo succede per diversi motivi:

* quando un processo termina l’esecuzione,
* allo scadere del quanto di tempo assegnato al processo in esecuzione,
* quando un processo attende il completamento di un’operazione di I/O.

I diversi tipi di schedulazione sono accomunati dalla necessità di completare la loro esecuzione in breve tempo per limitare il sovraccarico del sistema operativo.

I processi si possono suddividere in:

* processi CPU bound, che impegnano molto la CPU e richiedono poche operazioni di I/O,
* processi I/O bound che richiedono molte operazioni di I/0 e utilizzano poco la CPU.

Se l'insieme dei processi attivi è composto prevalentemente da processi CPU bound:

* può peggiorare il tempo di risposta dei processi interattivi lasciando inattivi i dispositivi periferici.

Se un insieme di processi è composto prevalentemente da processi I/O bound:

* può causare l'inattività del processore, perché aumenta la probabilità che tutti i processi attivi sono in attesa.

Per un corretto funzionamento del sistema, è necessario che l'insieme dei processi attivi sia equilibrato di entrambi i tipi di processi.

Lo scheduling si propone di ottimizzare differenti parametri che misurano l'efficienza del sistema:

* Massimizzare la percentuale di utilizzo della CPU; questa è un'importante misura di efficienza del sistema.
* Massimizzare il throughput (capacità di lavorazione) del sistema. Il throughputè è il numero di processi completati nell'unità di tempo.
* Minimizzare il sovraccarico del processore: l'obiettivo del sistema operativo è di massimizzare l'utilizzo del processore nello svolgere lavoro utile, eseguendo i processi utente.
* Minimizzare il tempo di turnaround. Il tempo di turnaround (tempo di circolazione) è il tempo che passa tra la sottomissione di un processo e la sua fine. Per l'utente è la durata di un job.
* Minimizzare il tempo di risposta dei processi interattivi. È il tempo impiegato dal processo a fornire il primo output. È un parametro importante per gli utenti interattivi.
* Prevenire la situazione di starvation. Inedia(starvation) голод тоска ☹ è la situazione quando un processo non viene mai eseguito. Anche i processi a bassa priorità prima o poi devono essere completati.

Algoritmi di schedulazione

L’algoritmi di schedulazione sono le tecniche usate dai sistemi operativi per operare la scelta quale processo a cui è assegnato la CPU deve essere mandato alla coda dei processi pronti.

Gli algoritmi di schedulazione possono essere essere classificati secondo due categorie:

* Schedulazione senza prerilascio (non preemptive scheduling), cioè un processo running non può essere interrotto durante l'esecuzione.
* Schedulazione con prerilascio (preemptive scheduling), cioè un processo running può essere interrotto e portato nello stato ready per lasciare il posto ad altri processi. Questo avviene sia per l'arrivo di un nuovo processo ad elevata priorità, sia per l'intervento dell'orologio di sistema (timer).

Schedulazione FCFS

FCFS (First Come First Served, primo arrivato primo servito) è una politica di schedulazione senza prerilascio, secondo la quale i processi sono eseguiti nell'ordine di arrivo.

Un nuovo processo è inserito in una coda di processi ready gestita in modalità FIFO (First In First Out). Quando un processo ottiene la CPU continua nell’esecuzione sino alla fine o finché si deve fermare in attesa di un evento. L'evento atteso, quando si presenta, causa il trasferimento di un processo dalla coda dei processi waiting nella coda dei processi ready.

Questo tipo di schedulazione usa l'ordine di arrivo come unico criterio per scegliere il processo da eseguire. Le conseguenze sono:

* Se un processo molto breve arriva subito dopo una serie di processi lunghi, attenderà molto tempo prima di essere eseguito.
* In un insieme misto di processi I/O bound e CPU bound, risultano favoriti i processi del secondo tipo che monopolizzano il processore, mentre sarebbe meglio privilegiare i processi I/O bound che, rilasciando frequentemente la CPU, permettono anche ai secondi di essere eseguiti.

Schedulazione SJF

 SJF (Shortest Job First, il lavoro più corto per primo) è un algoritmo di schedulazione senza prerilascio quando il primo processo ad essere eseguito è il più breve tra quelli in attesa:

* Con la schedulazione SJF i processi brevi sopravanzano quelli lunghi, che rischiano la starvation e hanno tempi di completamento meno prevedibili.
* I processi I/O bound rilasciano spesso la CPU, permettendo anche agli altri di essere serviti. Ma un processo di questo tipo può essere visto come una serie di brevi processi che rilasciano la CPU per eseguire operazioni di I/O.

L'algoritmo SJF è il migliore possibile per ridurre il tempo di attesa, ma non dà prevedere la durata di un job. È usato nella schedulazione di lavori batch: si pensi alle operazioni periodiche di calcolo e stampa degli stipendi, o alla preparazione degli estratti conto bancari.

Schedulazione usando la priorità

La priorità è un parametro che determina l’importanza o l'urgenza di un processo. In un sistema basato sulle priorità, il processo in esecuzione è sempre quello con la priorità più elevata tra i processi pronti per l’esecuzione.

I processi ready sono ordinati per priorità. Quando lo schedulatore deve scegliere tra i processi ready quello da mandare in esecuzione, prende sempre quello con la priorità più elevata. Il processo in esecuzione può essere forzatamente rimosso dalla CPU nel caso quano un processo a priorità più elevata pronto per l'esecuzione.

Ci sono due situazioni che possono causare il prerilascio del processo in esecuzione:

* quando arriva un nuovo processo a priorità più elevata,
* quando un evento causa lo spostamento di un processo a priorità più elevata dalla coda dei processi waiting a quella dei processi ready.

Ogni volta che il sistema operativo aggiorna la coda dei processi ready, controlla la priorità dei processi ready e quella del processo running per decidere quale deve proseguire.

Schedulazione Round Robin

La schedulazione Round Robin considera tutti i processi egualmente importanti.

II Round Robin è un algoritmo di schedulazione primo arrivato - primo servito con prerilascio, tipico dei sistemi time sharing.

Il quantum o time slice è un intervallo temporale che rappresenta la quantità massima di tempo di CPU concessa a un processo. Se il processo non rilascia la CPU entro questo tempo, è interrotto e posto in coda fra i processi ready.

Il Round Robin è sensibile al variare del time slice:

* se il time slice è breve, le interruzioni sono frequenti e l’efficienza del sistema diminuisce.
* Se il time slice è eccessivamente lungo, il Round Robin degenera in un sistema primo arrivato primo servito, e gli utenti interattivi hanno un peggioramento dei tempi di riposta.

I valori del time slice sono di diversi millisecondi, per esempio 20 msec.

Feedback scheduling

Un processo in esecuzione alterna periodi di utilizzo del processore (burst di CPU) a periodi nei quali esegue operazioni di lettura e scrittura (burst di I/O).

La schedulazione a retroazione (Il feedback scheduling) è una modalità di schedulazione che tiene conto contemporaneamente di diverse caratteristiche per privilegiare i processi con brevi burst di CPU:

* priorità dei processi,
* Round Robin
* storia passata del processo (per adeguare priorità e quantum)

Il feedback scheduling privilegia i processi con brevi burst di CPU evidenziando burst di CPU lunghi.

Per passare a una coda a priorità superiore bisogna che il processo rilasci la CPU senza consumare tutto il suo quantum. Una tecnica di scheduling con caratteristiche simili è usata, per esempio, dallo schedulatore di Windows.