

## ***Inversione della priorità ed ereditarietà della priorità***

Introduciamo il fenomeno dell'inversione della priorità attraverso un esempio.

Si consideri un sistema sul quale si trova in esecuzione 3 processi, ciascuno con un diverso livello di priorità. Per semplicità supponiamo che siano  $P_A$ ,  $P_B$ , e  $P_C$  i processi, e che i loro livelli di priorità siano rispettivamente 1 (minima), 2, e 3 (massimo). Supponiamo che seguente configurazione iniziale della nostra osservazione sia la seguente:

$t_0$ : coda processi pronti a priorità 1:	<u><math>P_A</math></u> *
coda processi pronti a priorità 2:	<i>vuota</i>
coda processi pronti a priorità 3:	<i>vuota</i>
coda processi bloccati:	$P_B, P_C$

Il carattere “\*” indica che il processo  $P_A$  detiene una particolare risorsa (ad esempio una stampante, ma anche l'accesso in mutua esclusione ad una determinata sezione critica). Il processo in esecuzione è quindi  $P_A$ , essendo l'unico processo pronto. Supponendo che successivamente il processo  $P_B$  si sblocchi, la configurazione del sistema diviene quindi:

$t_1$ : coda processi pronti a priorità 1:	$P_A$ *
coda processi pronti a priorità 2:	<u><math>P_B</math></u>
coda processi pronti a priorità 3:	<i>vuota</i>
coda processi bloccati:	$P_C$

ed il processo in esecuzione diviene  $P_B$ , avendo questo priorità maggiore di  $P_A$ . Supponiamo quindi che si sblocchi anche il processo  $P_C$ . Essendo questo il processo pronto a più alta priorità otterrà l'uso del processore:

$t_2$ : coda processi pronti a priorità 1:	$P_A$ *
coda processi pronti a priorità 2:	$P_B$
coda processi pronti a priorità 3:	<u><math>P_C</math></u>
coda processi bloccati:	<i>vuota</i>

Se durante la sua esecuzione il processo  $P_C$  cerca di ottenere l'accesso alla risorsa detenuta dal processo  $P_A$ , esso si blocca nuovamente, ed il processore viene nuovamente assegnato al processo  $P_B$ :

$t_3$ : coda processi pronti a priorità 1:	$P_A$ *
coda processi pronti a priorità 2:	<u><math>P_B</math></u>

coda processi pronti a priorità 3:	<i>vuota</i>
coda processi bloccati:	$P_C$ (in attesa della risorsa *)

Il fenomeno dell'**inversione di priorità** (o *priority inversion*) si verifica quando si ha un processo ad alta priorità—al limite un processo real-time (nel caso dell'esempio il processo  $P_C$ )—che si trova bloccato in attesa di una risorsa detenuta da un processo a bassa priorità (nell'esempio si tratta del processo  $P_A$ ), il quale potrebbe rilasciare la risorsa con notevole ritardo proprio in virtù del fatto che per la sua bassa priorità altri processi possono andare in esecuzione (il processo  $P_C$  dell'esempio). In una situazione del genere il processo ad elevata priorità risulta accodato ad un processo a bassa priorità, e quindi procede tutt'altro che speditamente (contrariamente a quanto ci si aspetterebbe da un processo cui è stata assegnata un'elevata priorità).

La soluzione al problema appena descritto prende il nome di **ereditarietà della priorità** (o *priority inheritance*), e consiste nell'assegnare temporaneamente al processo che detiene la risorsa la priorità del processo a maggiore priorità che la richiede (il processo bassa priorità viene “promosso”); non appena il processo rilascia la risorsa, ad esso viene riassegnata la priorità originaria (viene cioè “retrocesso”). In questo modo si “accelera” il processo che detiene la risorsa, e quindi si riduce il periodo di tempo durante il quale il processo a maggiore priorità risulta essere bloccato.

Vediamo la soluzione applicata all'esempio.

Non appena il processo  $P_C$  si blocca in attesa della risorsa detenuta da  $P_A$ , a quest'ultimo viene temporaneamente assegnata una priorità pari a 3, per cui gli viene assegnato il processore:

$t_4$ : coda processi pronti a priorità 1:	<i>vuota</i>
coda processi pronti a priorità 2:	$P_B$
coda processi pronti a priorità 3:	<u><math>P_A</math></u> *
coda processi bloccati:	$P_C$ (in attesa della risorsa *)

Quando  $P_A$  rilascia la risorsa, il sistema gli riassegna la priorità originaria, ed inoltre il processo  $P_C$  viene sbloccato e riprende l'esecuzione:

$t_5$ : coda processi pronti a priorità 1:	$P_A$
coda processi pronti a priorità 2:	$P_B$
coda processi pronti a priorità 3:	$\underline{P_C^*}$
coda processi bloccati:	<i>vuota</i>

Si noti che in assenza del meccanismo di ereditarietà della priorità, nella peggiore delle ipotesi la situazione si sarebbe sbloccata solo una volta che il processo  $P_B$  avesse completato la propria esecuzione; infatti, nell'ipotesi che il processo  $P_B$  non si bloccasse mai in attesa di una risorsa, anche la presenza di un meccanismo tipo round-robin: fino al suo completamento il processo  $P_B$  sarebbe sempre risultato il processo pronto a maggiore priorità.

La soluzione descritta è di un certo interesse per la realizzazione di sistemi operativi tempo-reale, ed in particolare di quelli *soft real-time*: essa consente ad un processo ad elevata priorità di entrare velocemente in possesso di una risorsa di cui necessita (in questo modo, ad esempio, un programma per la riproduzione di filmati può consentire una fruizione sufficientemente “fluida” del filmato).

La tecnica può comunque essere utilizzata anche in altri tipi di S.O., in quanto consente di realizzare una gestione della priorità che più si avvicina a quella attesa dall'utenza (ad esempio, l'ereditarietà della priorità è presente su molte VM Java).