

Il (soft) real-time con Linux

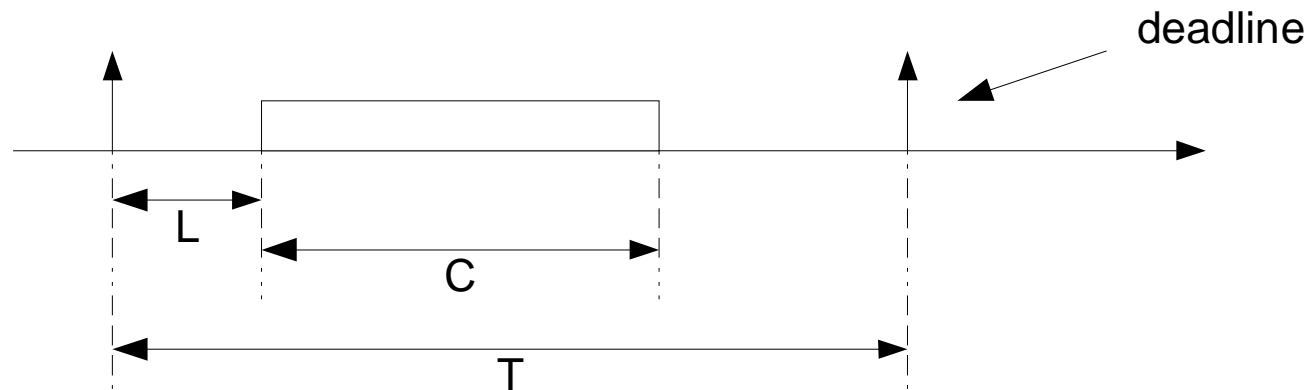
Rodolfo Giometti
r.giometti@ascensit.com

Cosa significa «real-time»

La definizione:

Un sistema real-time è un sistema che, dato un insieme di «task» schedulabile, è in grado di verificare/garantire che se arriva un nuovo «task» il nuovo insieme è ancora schedulabile.

Un task si dice schedulabile se può essere mandato in esecuzione senza che corra il rischio di sfondare la propria «deadline».



Un insieme di task è schedulabile quanto tutti i task sono schedulabili.

La definizione data è più generale di quella che «comunemente» si intende: gestione veloce delle interruzioni e veloce cambio di contesto.

Un sistema real-time non è necessariamente un sistema veloce. Un sistema può essere veloce quanto si vuole ma se non garantisce la **fattibilità di schedulazione** il sistema di controllo che implementa non ha la garanzia di funzionare correttamente!!!

In un sistema, per quanto veloce possa essere, si può sempre definire un task per cui la sua schedulazione non è garantita e quindi uno o più task sfondano la propria «deadline».

La teoria dei sistemi «real-time»

La teoria dei sistemi «real-time» ci dice chiaramente che per garantire la fattibilità di schedulazione un sistema **deve essere prevedibile**.

Nota: Molti sedicenti «Real-Time System» non garantiscono la fattibilità di schedulazione ma offrono semplicemente una gestione veloce delle interruzioni e un veloce cambio di contesto.

Hard & Soft «real-time»

Alcune volte lo sfondamento della deadline da parte di uno o più processi è accettabile e quindi si parla della differenza tra task di tipo «hard real-time» (sfondamento della deadline non accettabile) e task di tipo «soft real-time» (sfondamento della deadline accettabile anche se non desiderabile).

Se nel nostro sistema abbiamo solo task di tipo «soft real-time» allora è forse possibile risolvere i nostri problemi di controllo di processo utilizzando Linux così com'è, senza utilizzare quindi patch come RT-Linux/RTAI.

Linux ed il real-time

Il nucleo di Linux permette già nella sua versione standard (vanilla) di avere un «certo» controllo sulle politiche di schedulazione nonché sulla gestione dei processi in generale e della gestione del tempo.

Le varie tecniche disponibili sono più o meno efficaci se implementate nello spazio di nucleo o nello spazio utente.

Nello spazio utente

Nello spazio utente possiamo agire direttamente su:

- ◆ La politica di schedulazione dei processi.
- ◆ La gestione dell'immagine di un processo.
- ◆ La gestione del tempo.

Quello che alcune volte ci interessa è essere sicuri che al verificarsi di un evento il sistema sia il più «reattivo possibile» mandando in esecuzione una routine di gestione nello spazio utente.

Il fatto di avere un sistema «time sharing» ci porta inevitabilmente a delle «incertezze» sulla schedulazione dei processi.

La schedulazione di tipo «time sharing» è ottima per la «virtualizzazione» della CPU (dando cioè l'impressione all'utente di avere più CPU) ma questo non è accettabile quando si deve minimizzare la latenza di esecuzione di un processo.

La schedulazione FIFO e FIFO-RR

È possibile impostare la politica di schedulazione dei processi con la chiamata di sistema:

```
int sched_setscheduler(  
    pid_t pid,  
    int policy,  
    const struct sched_param *p);
```

a schedulazioni di tipo FIFO o FIFO-Round-Robin.

In questo modo è possibile definire direttamente quali processi debbano andare in esecuzione prima di altri e, addirittura, di decidere quando rilasciare la CPU.

Questi due tipi di schedulazione sono di tipo «preemptive» e la CPU viene quindi tolta al processo in esecuzione se un processo a più alta priorità è pronto per l'esecuzione.

La priorità per questi tipi di schedulazione è di tipo «statico» e quindi siamo sempre sicuri che il processo che abbiamo definito a priorità più alta andrà sempre in esecuzione per primo ogni volta che tale processo è pronto.

Si noti che tale comportamento può essere «simulato» anche utilizzando la normale priorità di tipo «dinamico» (*nice level*) dello schedulatore standard time sharing (OTHER) ma in questo caso non abbiamo la garanzia che invece gli altri schedulatori ci offrono.

La differenza tra la schedulazione FIFO e FIFO-RR è solo per il fatto che nella seconda è possibile definire un «quanto temporale» terminato il quale la CPU viene comunque rilasciata

Nel caso di schedulazione FIFO la CPU non viene rilasciata dal processo che la detiene fintanto che il processo in questione non la rilascia utilizzando la chiamata di sistema `sched_yield()`.

Utilizzando un programmino che genera eventi periodici si ha:

```
giometti@zaigor:~$ sudo ./rt 100000 8192
iterations = 100000
setting frequency 8192
min/avg/max = 1/1.036680/56
```

```
giometti@zaigor:~$ sudo nice --20 ./rt 100000 8192
iterations = 100000
setting frequency 8192
min/avg/max = 1/1.002260/6
```

```
giometti@zaigor:~$ sudo ./rt 100000 8192 10
iterations = 100000
setting priority 10
setting frequency 8192
min/avg/max = 1/1.002080/5
```

La gestione del «paging»

In alcune applicazioni (e soprattutto in quelle in cui ci interessa diminuire la latenza di esecuzione) il fatto di avere una tecnica di paging può non essere accettabile a causa dei possibili ritardi che questa tecnica può introdurre.

Se, per una qualche ragione, una pagina di un processo viene spostata sul disco nel momento in cui c'è da gestire l'arrivo di un evento notevole, si hanno dei ritardi indefiniti dovuti al fatto di dover riportare la pagina mancante in memoria per l'esecuzione.

In applicazioni di tipo real-time si preferisce rinunciare al paging proprio per evitare questi ritardi indeterminati.

Per disabilitare il paging di una o più pagine di memoria che compongono l'immagine di un processo è possibile utilizzare le chiamate di sistema:

```
int mlock(const void *addr, size_t len);  
int mlockall(int flags);
```

La gestione del tempo

La gestione del tempo può essere implementata con diverse tecniche:

- ◆ Usando le tecniche UNIX standard.
- ◆ Usando alcune estensioni proprie di Linux.

Le tecniche di UNIX risentono della risoluzione temporale del sistema (che si può aumentare fino a qualche decina di KHz) mentre alcune estensioni proprie di Linux possono aggirare questo problema.

È interessante a questo proposito il dispositivo `/dev/rtc` che si basa sul *real time clock* presente su quasi tutte le architetture.

Questo dispositivo permette di programmare l'invio di una interruzione hardware con frequenze regolabili fino a 8192Hz.

Un processo può programmare questo dispositivo e poi mettersi in attesa dell'interruzione che il timer invia quando scade il *timeout*.

Nota: Tutte le tecniche che si possono implementare nello spazio utente sono comunque sottoposte allo schedatore e ai tempi relativi alla gestione del cambio di contesto.

Nello spazio di nucleo

Nello spazio di nucleo è possibile ottenere delle latenze ancora minori per quanto riguarda la gestione di eventi poiché non si hanno cambi di contesto e non si ha il paging. Lo svantaggio è che il codice è più «rigido» e «delicato».

Tutti i task che girano nello spazio di nucleo operano nello spazio di indirizzamento del nucleo.

Un errore di programmazione può essere catastrofico!!!

La gestione degli eventi

Le interruzioni vengono gestite con gli «interrupt handler» che vengono mandati in esecuzione direttamente dall'hardware del sistema non passando per lo schedulatore.

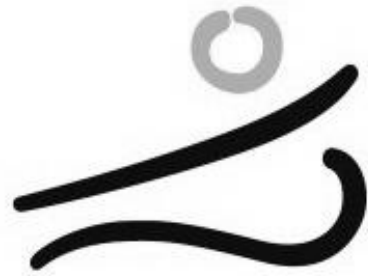
L'uso degli interrupt handler per la gestione degli eventi è comunque sconsigliato a vantaggio delle interruzioni software che hanno comunque «priorità» più alta dei normali processi ma girano ad interruzioni abilitate.

La gestione del tempo

Il tempo può essere facilmente gestibile utilizzando i «kernel timer». Si può cioè programmare l'esecuzione di un certo task più avanti nel tempo ad un istante prefissato.

Queste «entità» sono implementate con delle interruzioni di tipo software e dipendono dalla risoluzione del sistema.

Sulle piattaforme x86 la risoluzione temporale del sistema è per default 100Hz ma può essere incrementata (anche con un Pentium) a qualche migliaia di Hz modificando il valore della `define HZ` in `linux/include/asm/param.h`.



www.ascensit.com