SISTEMI REAL-TIME
Di Marco Giancola

I sistemi real-time sono quei sistemi informatici atti a reagire ad eventi esterni entro precisi vincoli temporali. Essi, a differenza degli altri sistemi informatici, non devono limitarsi ad eseguire correttamente le loro funzioni, ma devono anche completarle entro chiari limiti temporali. Il termine real-time (tempo reale) può indurre facilmente a ritenere che tali sistemi siano più veloci degli altri, ma in realtà non è così; infatti, i vincoli temporali a cui essi sono sottoposti possono essere anche intervalli di tempo della durata di molti minuti.
Tipici esempi di sistemi real-time sono:

- sistemi militari per la difesa (es.: sistemi di difesa missilistici);
- sistemi di controllo (es.: il pilota automatico di un aereo, il controllo di impianti nucleari e chimici, sistemi di guida di missili e satelliti);
- sistemi di monitoraggio/rilevamento (es.: rilevazione dati ambientali, sistemi di sorveglianza, controllo del traffico);
- robot;
- macchine di distribuzione automatiche;
- sistemi di telecomunicazioni;
- sistemi multimediali;
- piccoli dispositivi come cellulari, videogiochi, giocattoli computerizzati, elettrodomestici e simili.

È da notare che molti di essi sono *sistemi embedded*, ossia hanno incorporato il computer che li controlla.
Un sistema si definisce *hard real-time* o *soft real-time* a seconda che il limite temporale entro cui deve espletare la sua funzione sia specificato come valore assoluto o come valore medio. In parole povere, per i sistemi soft real-time, a differenza dei sistemi hard real-time, è ammissibile un certo margine di ritardo nel fornire la risposta. Un esempio di sistema hard real-time è il sistema di controllo della temperatura del nucleo di una centrale nucleare, dove il mancato rispetto dei vincoli temporali può provocare un evidente disastro. Un'altro esempio è il controllo di processo di un laminatoio per l'acciaio: la laminazione va eseguita entro un certo intervallo di tempo, dopo la formazione del materiale ferroso fuso, prima che le lastre si induriscano (questo è un esempio di sistema hard real-time tale che il non rispetto dei vincoli temporali produce effetti inaccettabili ma non catastrofici). Un esempio, invece, di sistema soft real-time può essere un riproduttore DVD, in cui il mancato rispetto dei vincoli temporali può determinare un degrado della qualità del filmato, ma non pregiudica il proseguimento della riproduzione. Un'altro esempio di sistema soft real-time è rappresentato dagli sportelli automatici per la distribuzione di banconote: tali sistemi, mediamente, devono avere un tempo di risposta accettabile per l'utente, altrimenti potrebbero causare l'insoddisfazione della clientela, ma niente di più.
Rispetto alla velocità della risposta, possiamo suddividere i sistemi real-time in veloci e lenti, anche se non c'è una netta separazione tra queste 2 classi. I sistemi con tempi di risposta inferiori al secondo si possono certamente considerare veloci, mentre quelli con tempi di risposta misurabili in minuti si possono considerare lenti (lasciando, in tal modo, un intervallo d'incertezza dell'ordine dei secondi).

	⇐ VELOCE (μ) (mS)	⇔ (sec)	LENTO (min) (ore) ⇒
Hard real-time	*hard-veloce*		*hard-lento*
Soft/Near real-time	*soft-veloce*		*soft-lento*

I sistemi real-time si possono inoltre suddividere in altre 2 categorie: quelli che modificano l'ambiente e quelli che non lo modificano. I sistemi appartenenti alla prima categoria si dividono in:

1. *Sistemi ad anello aperto*, le cui azioni sono preprogrammate. Esempi:
    - robot per la verniciatura;
    - robot per la movimentazione di pezzi.
2. *Sistemi con feedback*, le cui azioni sono in funzione delle informazioni sensoriali che ricevono. Esempi:
    - controllori per impianti chimici o nucleari;
    - sistemi di difesa militari;
    - sistemi di regolazione di volo;
    - robot evoluti.

Il componente hardware di un sistema real-time deve essere in grado di gestire e registrare gli eventi esterni (per esempio, di ricevere dati dai sensori e inviare ordini agli attuatori), mentre il componente software deve essere in grado di reagire su più processi, ovvero deve essere un software *multitasking*. Infatti, le attività del sistema sono concorrenti, in quanto esso deve simultaneamente controllare diverse parti dell'ambiente e reagire ai suoi cambiamenti. Per comprendere meglio questo concetto, è sufficiente considerare, ad esempio, il sistema di controllo di volo di un aereo o di una centrale nucleare: è impensabile che un sistema del genere non reagisca ad un evento in real-time perché è impegnato a svolgere altre operazioni.
Tra i linguaggi di programmazione utilizzati per sviluppare applicazioni real-time, citiamo:

- *Assembly*, che, tra tutti i linguaggi informatici, è il più vicino al linguaggio macchina. Ha avuto una certa diffusione in passato, poiché spesso utilizzava al meglio le risorse hardware, ma è stato ormai praticamente abbandonato il suo uso per scopi real time, a causa dei suoi forti vincoli con l'hardware e della difficoltà di programmazione, di manutenzione del software e di verifica dei vincoli temporali.
- *Ada*, un linguaggio sviluppato verso la fine degli anni 70, su iniziativa del Dipartimento della Difesa degli Stati Uniti, principalmente per lo sviluppo di applicazioni militari. Si tratta di un linguaggio estremamente efficiente, dotato di molte peculiarità (tra cui quella di consentire lo sviluppo di applicazioni multitasking) che lo rendono ottimale per lo sviluppo di varie tipologie di software, ma soprattutto per i software real-time.
- *C*, creato da Dennis Ritchie nei Bell Laboratories della AT&T nel 1972 come evoluzione del linguaggio B usato per lo sviluppo dei primi sistemi operativi UNIX. È stato per lungo tempo il linguaggio dominante nei domini applicativi caratterizzati da forte enfasi sull'efficienza, tra cui le applicazioni real-time.
- *C++*, un linguaggio di programmazione orientato agli oggetti, sviluppato da Bjarne Stroustrup nel 1983 presso i Bell Laboratories. Si tratta di un perfezionamento del linguaggio C, ottenuto aggiungendo al C alcune delle peculiarità del linguaggio Simula e di altri linguaggi come Ada e l'Algol 68. Insieme al C, è attualmente il linguaggio più utilizzato per lo sviluppo di software real-time.
- *J2ME (Java 2 Micro Edition)*, che è la tecnologia più diffusa per lo sviluppo di software dedicato a dispositivi come cellulari, palmari e simili, che costituiscono un esempio di sitemi real-time.

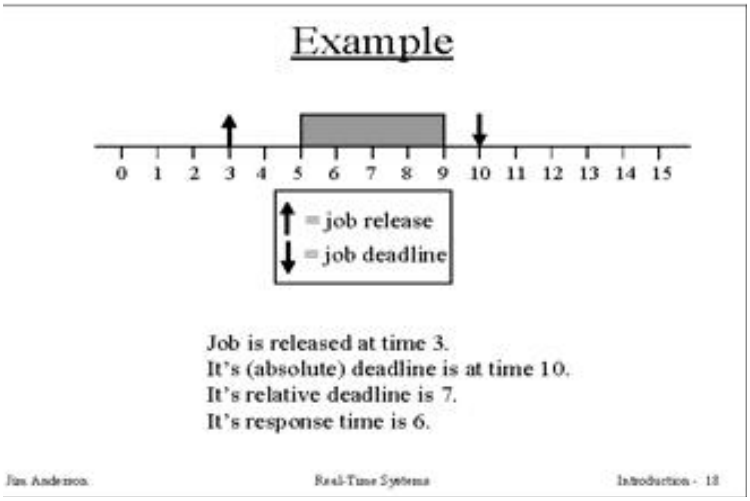
Un *processo* (o *task* o *job*) è una sequenza di istruzioni che, in assenza di altre attività, viene eseguita dal processore in modo continuativo fino al suo completamento. Un sistema real-time deve garantire che ciascun task termini entro un dato vincolo temporale detto *deadline*.
Un task di un sistema real-time può essere:

1. *periodico*: quando viene eseguito periodicamente;
2. *aperiodico*: quando viene attivato ad intervalli di tempo irregolari (è attivato al verificarsi di un evento).

Può essere anche:

1. *soft*: se può non rispettare la sua scadenza (in gergo si dice *sfondare la deadline*), provocando un danno non irreparabile al sistema;
2. *hard*: quando, se sfonda la sua deadline, provoca un danno irreparabile al sistema.

Un sistema hard real-time è in grado di gestire hard task.
Gli stati di un task sono: *ready* (se è in attesa della CPU), *running* (se sta usando la CPU) e *blocked* (se è in attesa di un evento). Il *release time* è l'istante in cui un task diventa pronto per l'esecuzione, mentre il corrispondente stimolo viene chiamato *release event*. Può esserci un ritardo tra l'arrivo del release event e l'istante in cui il corrispondente task diventa ready. Il *response time* è l'intervallo temporale che intercorre tra il release time e l'effettivo completamento del task. Il massimo response time ammissibile per un task viene detto *relative deadline*. Viene denominata invece *absolute deadline* di un task la somma algebrica di response time e relative deadline.



Nel funzionamento di un sistema real-time, ricopre un ruolo fondamentale lo *schedulatore* (o *scheduler*). Lo schedulatore è quel componente dei sistemi operativi multitasking in grado di far eseguire al processore più processi parallelamente, attraverso un'operazione chiamata *schedulazione* (*scheduling*). Nel caso dei sistemi real-time, nel momento in cui viene rilevato un evento esterno, lo schedulatore ha il compito di determinare un'opportuna sequenza di esecuzione dei processi tale da garantire il rispetto di tutti i vincoli temporali.

Consideriamo un insieme di n task, ognuno con una propria deadline d_i. Definiamo la *funzione di costo* del task i-esimo come:

$$C_i(t) = \begin{cases} 0 & t \le d_i \\ \infty & t > d_i \end{cases}$$

se il task *i*-esimo è di tipo hard, oppure come:

$$C_i(t) = \begin{cases} 0 & t \le d_i \\ f(t) & t > d_i \end{cases}$$

se il task *i*-esimo è di tipo soft, dove f è una funzione monotona crescente. La funzione di costo totale sarà:

$$C(t) = \sum_{i=1}^{n} C_i(t)$$

I task di tipo hard dovranno quindi essere schedulati in modo da terminare entro un lasso di tempo minore della deadline, in modo da far valere la funzione $C_i(t)$ 0 e non ∞; mentre i task soft dovranno anch'essi far valere 0 $C_i(t)$, ma, in questo caso, un eventuale sfondamento della deadline non manderà il costo globale all'infinito. La schedulazione migliore è quella che minimizza la funzione di costo totale C(t).

Si definisce *utilizzazione* di un task periodico il rapporto tra il suo tempo d'esecuzione e il suo periodo: $U_i = E_i/P_i$. Se abbiamo n task periodici, l'utilizzazione totale sarà la somma delle n utilizzazioni Ui:

$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} E_i / P_i$$

U misura il carico del processore e deve essere ≤ 1, altrimenti il processore risulterà sovraccaricato e non sarà possibile schedulare gli n task periodici.

Bibliografia

http://feanor.sssup.it/~giorgio/slides/rts/intro1w.pdf
http://feanor.sssup.it/~giorgio/slides/rts/intro2w.pdf
http://it.wikipedia.org/wiki/Ada_(linguaggio)
http://it.wikipedia.org/wiki/Assembly
http://it.wikipedia.org/wiki/C%2B%2B
http://it.wikipedia.org/wiki/J2ME
http://it.wikipedia.org/wiki/Linguaggio_C
http://it.wikipedia.org/wiki/Linguaggio_real-time
http://it.wikipedia.org/wiki/Real-time
http://it.wikipedia.org/wiki/Scheduler
http://it.wikipedia.org/wiki/Sistema_embedded
http://it.wikipedia.org/wiki/Sistema_operativo_real-time
http://www.artisansw.it/RtPprocess/RtP_embedded.asp
http://www.dii.unisi.it/~rigutini/teaching/talks/Sistemi%20Real-Time.pdf
http://www.embedded.it/?q=content/definizioni
http://www.math.unipd.it/~tullio/RTS/2009/L01.pdf