Lezione R13

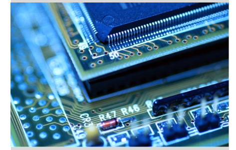
Sistemi operativi real-time

Sistemi operativi open-source, embedded e real-time

7 dicembre 2017

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma Tor Vergata

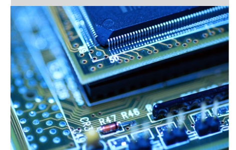


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Di cosa parliamo in questa lezione?

In questa lezione richiamiamo alcune nozioni di base sui sistemi operativi con particolare riferimento a quelli real-time

1. Sistemi operativi real-time
2. Interruzioni hardware
3. Gestione del tempo
4. Schedulazione
5. Standard di riferimento



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Sistemi operativi general-purpose

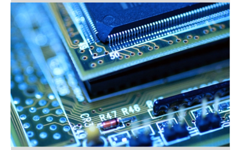
Quali sono gli obiettivi di un sistema operativo?

Interagire con l'hardware:

- Fa funzionare i dispositivi (programmi *driver*)
- Controlla l'uso delle risorse H/W (consumo di energia, salva-schermo, . . .)

Fornire un ambiente di esecuzione alle applicazioni:

- Costruisce una astrazione dell'architettura fisica
- Offre interfaccie di accesso ai dispositivi H/W
- Distribuisce le risorse H/W alle applicazioni
- Implementa i servizi di comunicazione tra applicazioni locali e remote
- Permette l'esecuzione contemporanea di più applicazioni (*multitasking*)
- Consente l'accesso di più utenti (*multiuser*)



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Sistemi operativi real-time

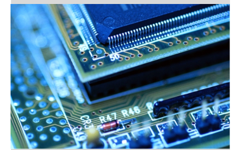
In cosa differisce un sistema operativo real-time da un sistema operativo general purpose? ***Le applicazioni!***

Le applicazioni real-time sono profondamente differenti rispetto a quelle di un sistema operativo general-purpose

Le caratteristiche distintive delle applicazioni real-time:

- Predicibilità
- Affidabilità

Talvolta le applicazioni real-time debbono avere *tempi di risposta rapidi*, ma questa proprietà non è caratterizzante



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

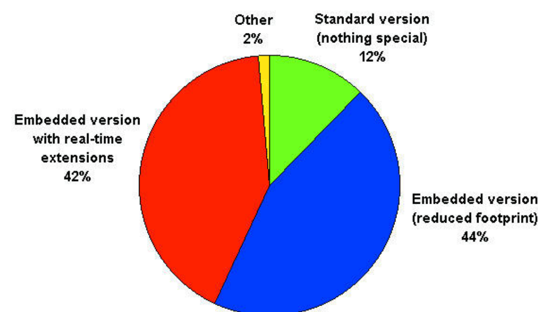
Sistemi operativi embedded e real-time

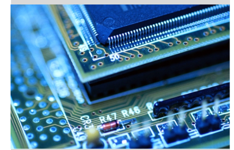
Spesso le applicazioni real-time sono anche embedded

Devono quindi essere:

- Compatte
- Scalabili
- Con ridotto consumo di risorse

Quali necessità determinano la scelta di un sistema operativo?





Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

RTOS a microkernel

Molti SO per sistemi embedded e RT sono basati su un modello detto a *microkernel*

Il microkernel è un piccolo programma che realizza pochi servizi essenziali:

- driver dei circuiti H/W di base
- schedulazione dei processi
- comunicazione di base tra processi

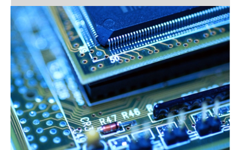
Tutti gli altri servizi offerti alle applicazioni (driver delle periferiche, stack di rete, file system, . . .) sono realizzati da altre applicazioni di sistema

Quali vantaggi offre l'approccio microkernel agli RTOS?

Il microkernel è costituito da poche linee di codice, quindi è possibile verificarlo con accuratezza

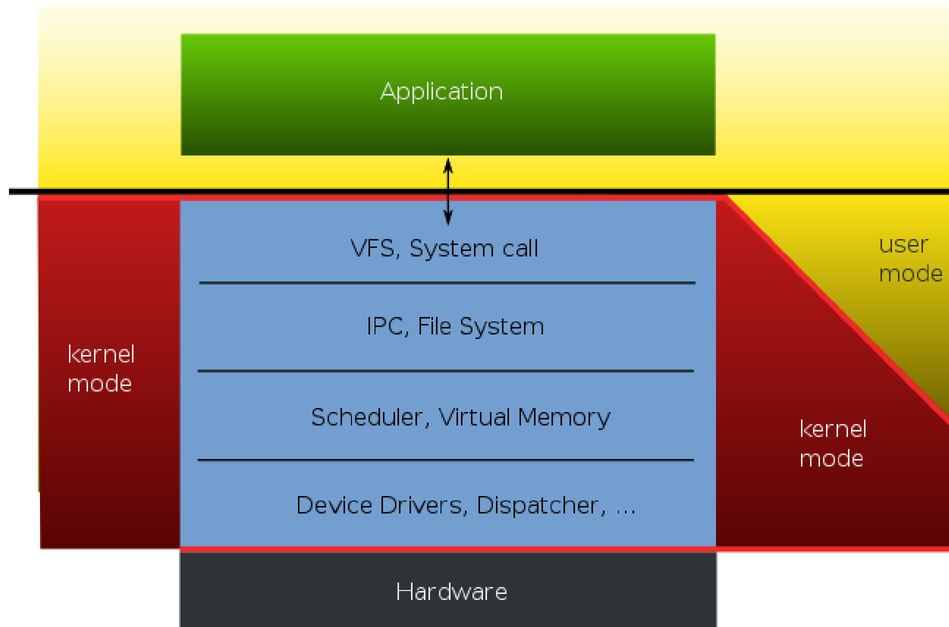
Sistemi operativi real-time

Marco Cesati

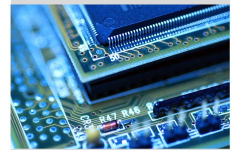


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Struttura di un SO monolitico

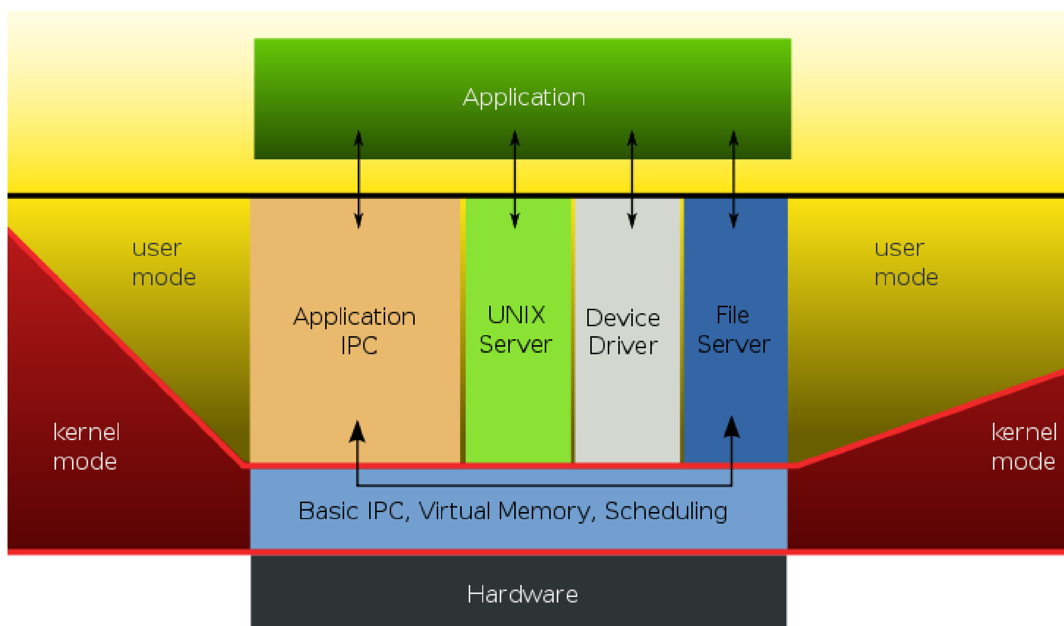


Esempi: Linux, FreeBSD, SunOS Solaris, . . .

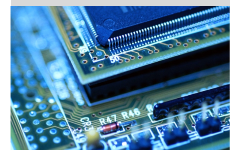


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

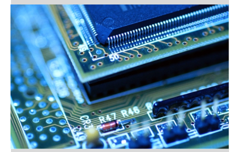
Struttura di un SO a microkernel



Esempi: QNX, GNU Hurd (Mach & Ell Four), BeOS, . . .

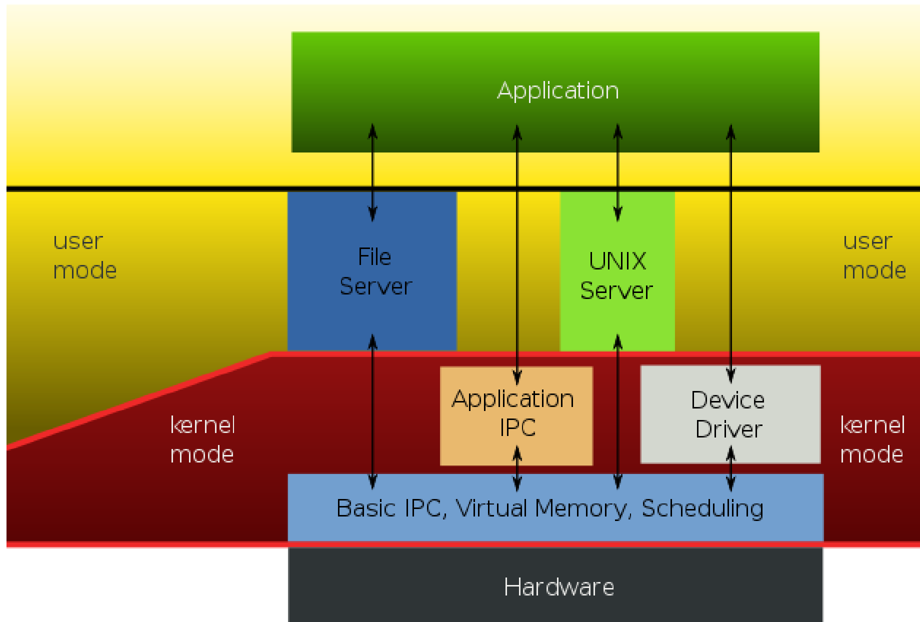


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

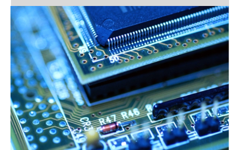
Struttura di un SO ibrido



Fonte: Golftheman – public domain

Esempi: Windows NT, Mac OS X (XNU), . . .

Sistemi operativi real-time
Marco Cesati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Caratteristiche chiave di un RTOS

Un sistema operativo che opera in un contesto real-time ed embedded deve quindi offrire:

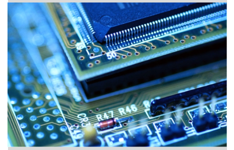
- Risposta ad eventi esterni predicibile: cura nella gestione delle interruzioni hardware
- Risposta ad eventi esterni efficiente: bassa latenza nei tempi di risposta
- Gestione affidabile e precisa di eventi temporali:
  - gestione dei timer e clock hardware con le relative interruzioni
  - gestione del tempo di sistema
  - gestione di allarmi e timer software per le applicazioni

Caratteristiche chiave di un RTOS (2)

- Schedulazione predicibile ed efficace dei task
  - implementazione di algoritmi deterministici (non euristici)
  - supporto al partizionamento dei task in sistemi multiprocessore
- Gestione della comunicazione e sincronizzazione dei task
  - memoria condivisa, code, mailbox, segnali
  - primitive di sincronizzazione
- Gestione della memoria
  - memoria virtuale
  - protezione dello spazio di indirizzamento

In un RTOS tutte queste caratteristiche devono sempre essere presenti? ***No!***

Ad es., in un sistema embedded le applicazioni RT potrebbero non utilizzare memoria virtuale e spazi di indirizzi separati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

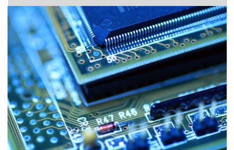
Gestione delle interruzioni hardware

Ogni occorrenza di una interruzione hardware deve essere gestita tramite l'esecuzione di una opportuna procedura del SO o di una applicazione

Spesso la procedura deve essere attivata e conclusa rapidamente dopo la generazione dell'interruzione

Il tempo richiesto per la completa gestione di un interruzione (*latenza*) dipende comunque dalla periferica H/W:

- la lettura dei dati da un sensore è tipicamente un operazione breve
- il trasferimento di grandi blocchi di dati, ad esempio in una operazione su memoria di massa o su rete, è tipicamente una operazione lunga

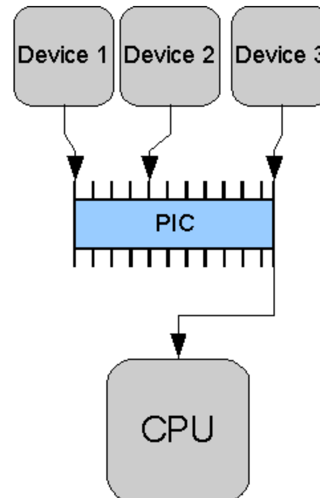


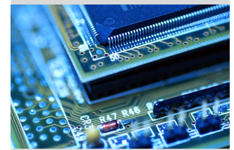
Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Gestione delle interruzioni hardware (2)

In ogni caso, il SO deve dare risposte rapide alle interruzioni, altrimenti le periferiche H/W potrebbero funzionare male

- 1 linea fisica di interruzione da ogni dispositivo
- 1 sola linea verso il processore
- l'interruzione arriva al PIC (Programmable Interrupt Controller)
- il PIC asserisce la linea verso la CPU e aspetta conferma di ricezione (*acknowledge*)
- la CPU deve confermare velocemente la ricezione per poter ricevere ulteriori interruzioni dagli altri dispositivi





Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Gestore delle interruzioni e ISR

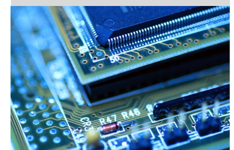
Problema: le interruzioni hardware devono essere confermate nel minor tempo possibile

Soluzione: i SO gestiscono le interruzioni hardware in più fasi distinte

Almeno due fasi implementate da due procedure distinte:

- *interrupt handler*
  - priorità elevata
  - dà conferma della ricezione dell'interruzione al PIC
  - salva e recupera il contesto di esecuzione del processore
- *interrupt service routine* (ISR):
  - priorità più bassa (ma generalmente superiore ai processi nel sistema)
  - esegue le operazioni specifiche per l'interruzione ed il dispositivo che l'ha generata

Sistemi operativi real-time
Marco Cesati



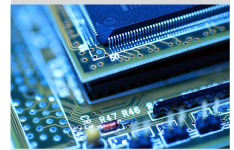
Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

La schedulazione

- La schedulazione di applicazioni real-time è uno dei punti fondamentali degli RTOS
- L'obiettivo dello scheduler è di determinare il successivo task da porre in esecuzione
- La scelta è effettuata utilizzando politiche di schedulazione semplici e dal comportamento facilmente predicibile

Gli RTOS oggi più adottati non supportano alcun tipo di analisi di schedulabilità o test di accettazione *on-line* per nuovi task

L'analisi di schedulabilità è generalmente realizzata durante la fase di progetto ed è normalmente affiancata da un test approfondito del comportamento del sistema



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Schedulazione a priorità fissa

- Tutti gli RTOS implementano un qualche tipo di politica di schedulazione preemptive con priorità fissa
- In molti RTOS lo scheduler è esclusivamente clock-driven

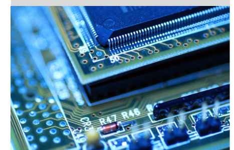
Va considerato nell'analisi di schedulabilità!

- Generalmente gli RTOS offrono un numero adeguato di livelli di priorità, ad esempio:

  Windows CE: 256 Linux: 100 VxWorks: 256

- Tuttavia l'analisi teorica relativa ai test di schedulabilità per algoritmi preemptive a priorità fissa prevede un numero infinito di livelli di priorità differenti

Va considerato anche il numero di livelli di priorità nell'analisi di schedulabilità!



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Perdita di schedulabilità

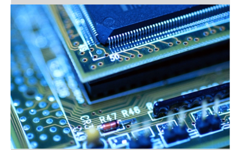
In generale un numero finito di livelli di priorità causa una *perdita di schedulabilità*

- Siano $1, 2, \ldots, \Omega_n$ le priorità assegnate dall'algoritmo di schedulazione
- Sia Ω_s ($< \Omega_n$) il numero di livelli di priorità del sistema
- L'associazione tra le priorità assegnate e le priorità di sistema è rappresentato dai valori scelti per le priorità di sistema $\pi_1, \pi_2, \ldots, \pi_s$, ove:
  - $\pi_i \in \{1, 2, \ldots, \Omega_n\}$, per ogni i
  - $\pi_i < \pi_j$ se $i < j$
  - tutte le priorità assegnate numericamente minori o uguali a π_1 sono mappate su π_1
  - tutte le priorità assegnate tra $\pi_{k-1}+1$ e π_k sono mappate su π_k, per $1 < k \leq \Omega_s$

Esempio: se $\Omega_n = 10$ e $\Omega_s = 3$, l'associazione lineare tra le priorità assegnate e quelle di sistema mappa 1, 2 e 3 su π_1; 4, 5 e 6 su π_2; 7, 8, 9 e 10 su π_3 $\quad \Rightarrow \quad \pi_1 = 3$, $\pi_2 = 6$, $\pi_3 = 10$

Sistemi operativi real-time

Marco Cesati



Schema della lezione

RTOS

Interruzioni H/W

Schedulazione

Standard per RTOS

Caratteristiche comuni

Esempi di RTOS

Perdita di schedulabilità (2)

Come tenere conto nell'analisi di schedulabilità che alcuni task possono avere identica priorità?

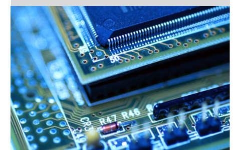
Considerato un task T_i, siano $\mathcal{T}_E(i)$ e $\mathcal{T}_H(i)$ gli insiemi dei task di priorità uguale e superiore a T_i, rispettivamente

$$w_i(t) = e_i + b_i + \sum_{T_k \in \mathcal{T}_E(i)} e_k + \sum_{T_k \in \mathcal{T}_H(i)} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

$$w_{i,j}(t) = j \cdot e_i + b_i + \sum_{T_k \in \mathcal{T}_E(i)} \left(\left\lceil \frac{(j-1)p_i}{p_k} \right\rceil + 1 \right) e_k + \sum_{T_k \in \mathcal{T}_H(i)} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

Sistemi operativi real-time

Marco Cesati



Schema della lezione

RTOS

Interruzioni H/W

Schedulazione

Standard per RTOS

Caratteristiche comuni

Esempi di RTOS

Associazione a rapporto costante

È conveniente utilizzare una associazione lineare tra priorità assegnate e priorità di sistema? ***In generale no!***

Lo scheduler non potrebbe distinguere i job di priorità più alta

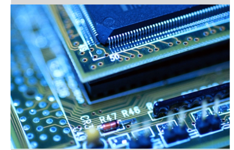
Si utilizza invece una associazione che riserva più livelli di priorità di sistema ai livelli di priorità assegnata più alti, accorpando insieme le priorità assegnate di livello più basso

In pratica si può cercare di mantenere approssimativamente costanti i rapporti

$$g_k = \frac{\pi_{k-1} + 1}{\pi_k} \quad (1 < k \le \Omega_s)$$

Nell'esempio precedente considerando $\pi_1 = 1$, $\pi_2 = 4$, $\pi_3 = 10$ si ottengono rapporti uguali a 1/2

Quindi 1 è mappato su π_1; 2, 3 e 4 su π_2; da 5 a 10 su π_3



Schema della lezione

RTOS

Interruzioni H/W

Schedulazione

Standard per RTOS

Caratteristiche comuni

Esempi di RTOS

Associazione a rapporto costante (2)

Teorema (Lechoczky & Sha, 1986)

Per l'algoritmo di scheduling RM, con scadenze relative pari al periodo e numero n di task elevato, usando l'associazione a rapporto costante con $g = \min_{1 < k \le \Omega_s} g_k$ allora:

$$U_{\text{RM}}(g) = \begin{cases} \ln(2g) + 1 - g & \text{se } g > 1/2 \\ g & \text{se } g \le 1/2 \end{cases}$$

La *schedulabilità relativa* indica la perdita di schedulabilità dovuta ad un numero limitato di livelli di priorità nel sistema:

$$U_{\text{RM}}(g) / \ln 2$$

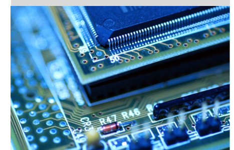
Casi limite:

$g = 1$: $U_{RM}(g)/\ln 2 = 1$ $\Rightarrow$ nessuna perdita

$g = 1/2$: $U_{RM}(g)/\ln 2 = 1/(2\ln 2)$ $\Rightarrow$ perdita del 28%

Sistemi operativi real-time

Marco Cesati



Schema della lezione

RTOS

Interruzioni H/W

Schedulazione

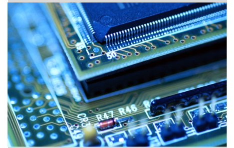
Standard per RTOS

Caratteristiche comuni

Esempi di RTOS

Schedulazione a priorità dinamica

- Tutti gli RTOS offrono delle API che consentono di impostare le priorità di un task direttamente sulla base della deadline relativa
- Generalmente lo scheduler implementa liste di task ordinate in base alla deadline relativa dei task
- Pochi RTOS (RTAI, RTLinux) supportano nativamente una politica di scheduling EDF
- Il reinserimento automatico dei task in una coda a priorità nella corretta posizione in base alla deadline assoluta è inefficiente
  - Si utilizzano strutture di dati dinamiche più sofisticate, ad esempi alberi di ricerca bilanciati
  - Si utilizzano strutture di dati molto semplici (ad es., vettore statico), limitando però drasticamente il numero di task periodici supportati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

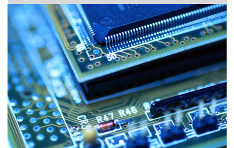
Standard per RTOS

Gli standard per RTOS hanno un ruolo molto importante perché consentono

- la portabilità delle applicazioni
- l'interoperabilità dei sistemi
- la possibilità di implementazioni alternative dei RTOS (maggiore concorrenza e migliore qualità)

I principali standard per RTOS esistenti sono:

- POSIX (estensioni real-time dello standard POSIX)
- OSEK/VDX (sistemi automobilistici)
- ARINC 653/APEX (sistemi avionici)
- ITRON/μITRON (sistemi embedded)



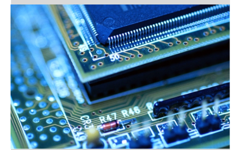
Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

POSIX

Lo standard POSIX (IEEE Std 1003.1-2014) include una estensione dedicata alla realizzazione dei sistemi real-time

Definisce API per ottenere

- mutua esclusione con priority inheritance
- attesa e sincronizzazione tramite variabili condizione
- memoria condivisa
- code di messaggi a priorità
- schedulazione preemptive a priorità fissata
- server sporadici
- gestione del tempo con alta risoluzione (sia timer che ritardi)
- possibilità di misurare e limitare il tempo d'esecuzione dei task



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

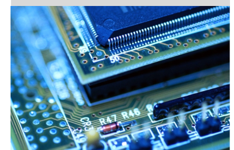
POSIX (2)

Lo standard definisce 4 profili differenti:

- Minimal Real-Time System (PSE51), per piccoli sistemi embedded: solo thread, non processi; I/O tramite device file; nessun file system.
- Real-Time Controller (PSE52), per sistemi robotici: come PSE51, ed in più un file system con cui gestire i file regolari.
- Dedicated Real-Time System (PSE53), per grandi sistemi embedded tipo avionici: come PSE52, ed in più processi multipli con meccanismi di protezione degli accessi alle risorse
- Multi-purpose Real-Time System (PSE54), per sistemi general-purpose con applicazioni sia real-time che non real-time: come PSE53, ed in più tutti i servizi POSIX per i sistemi operativi general-purpose

Sistemi operativi real-time

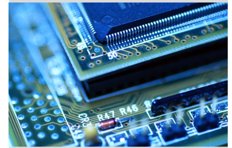
Marco Cesati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

OSEK/VDX

- Progetto congiunto di diverse industrie automobilistiche (BMW, Bosch, DaimlerChrysler, Opel, Siemens, Volkswagen, Renault, Pegeout Citröen) e l'università di Karlsruhe
- Definisce un insieme di API per un sistema real-time (OSEK) integrato in un sistema di gestione di rete (VDX)
- Orientato a sistemi di controllo con vincoli real-time stringenti, alta criticità, e grandi volumi di produzione
- Tiene in grande considerazione l'ottimizzazione del codice, la riduzione dell'occupazione di memoria, ed il miglioramento delle prestazioni del RTOS
- Propone interfacce e protocolli ad alto livello per le comunicazioni interne del veicolo (OSEK COM)
- Propone interfacce e protocolli per l'interoperabilità dei vari sistemi embedded all'interno dell'autoveicolo, definendo
  - politiche di accesso
  - meccanismi per la resistenza ai guasti
  - procedure di diagnostica della rete edi comunicazione

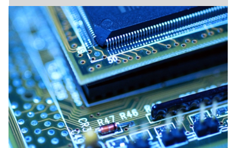


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

OSEK/VDX (2)

Un tipico sistema OSEK ha le seguenti caratteristiche:

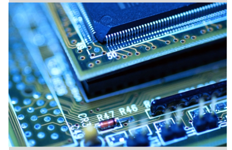
- Scalabità: dai microcontrolleri a 8 bit ai processori più potenti. Vengono definite quattro classi di conformità, ma in ogni caso non sono previsti meccanismi di protezione della memoria
- Portabilità software: Tra l'applicazione ed il SO è presente uno strato di interfaccia scritto in ISO/ANSI-C. In ogni caso non vengono specificate interfacce verso i sistemi di I/O.
- Configurabilità: il progettista può utilizzare degli strumenti di configurazione standard per definire i servizi e l'uso delle risorse. Viene proposto un linguaggio chiamato OIL (OSEK Implementation Language) per codificare i dati di configurazione.
- Allocazione statica: tutte le strutture di dati del kernel e delle applicazioni sono allocate staticamente.
- Supporto per architetture "time triggered": fornisce la specifica di un sistema operativo basato sul tempo (OSEKTime OS) che può essere completamente integrato nel framework OSEK/VDX.



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

ARINC 653/APEX

- Lo standard ARINC 653 (Avionics Application Software Standard Interface) fa parte di una famiglia di standard utilizzati per la progettazione e la certificazione dei sistemi avionici
- Obiettivo: consentire la progettazione, implementazione, certificazione ed esecuzione di sistemi safety-critical e real-time
- Lo standard consente di definire sistemi IMA (Integrated Modular Avionics): diversi componenti software con diversi requisiti di robustezza coesistono ed utilizzano le stesse risorse hardware
- Le applicazioni real-time interagiscono con i servizi offerti dalla piattaforma ARINC 653 utilizzando un insieme di API chiamato APEX (Application Executive)
- La memoria fisica è suddivisa in *partizioni*: ciascun sotto-sistema software utilizza una propria partizione

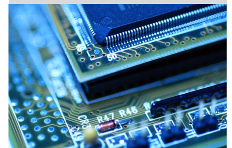


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

ARINC 653/APEX (2)

- Uno scheduler cyclic-executive è utilizzato per schedulare l'esecuzione dei programmi nelle varie partizioni
  - Non è in alcun modo possibile per un sotto-sistema in una partizione consumare più tempo di quanto allocato dallo scheduler per la partizione
- Il sotto-sistema in una partizione può definire una o più applicazioni, eventualmente costituite da più processi periodici schedulati tramite priorità fissa
- Comunicazioni tra processi in partizioni differenti sono gestite tramite APEX come scambio di messaggi
  - sampling messages: hanno sempre la stessa struttura, ed ogni occorrenza di un certo messaggio sovrascrive la precedente occorrenza dello stesso tipo
    - le API non bloccano mai
  - queuing messages: possono avere diversa struttura
    - le API possono bloccare se la coda di messaggi è piena (TX) o vuota (RX)
    - Il tempo massimo di blocco può essere definito in fase di invio o ricezione del messaggio

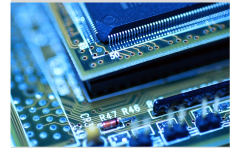
Sistemi operativi real-time
Marco Cesati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

ITRON/μITRON

- TRON (The Real-time Operating System Nucleus) è un progetto giapponese nato nel 1984
  - Prof. Ken Sakamura, Università di Tokyo
  - Definisce le API e le linee di progetto di un RTOS
- Da TRON sono scaturiti una lunga serie di prodotti industriali, che sono diventati standard de-facto nell'industria giapponese
  - ITRON (Industrial TRON): per sistemi embedded
  - μITRON: variante di ITRON per sistemi embedded a 8 e 16 bit
  - JTRON (Java TRON): variante di ITRON per sistemi basati su Java
  - BTRON (Business TRON): per computer con interfaccia utente come desktop computer e tablet
  - CTRON (Central and Communication TRON): per mainframe ed apparati di rete

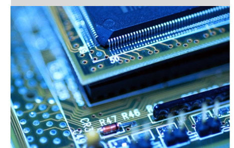


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

ITRON/μITRON (2)

- ITRON e μITRON hanno una enorme diffusione in Asia
  - Più di 50 RTOS industriali
  - Più di 35 architetture di microprocessori supportate
  - Presenti in tutti i settori dell'industria orientale
- Caratteristica dello standard è la *loose standardization*
  - API specificate a livello sorgente, non esistono requisiti di compatibilità a livello del codice eseguibile
  - Parametri delle API passati separatamente o come unico pacchetto, a discrezione del RTOS
- Le ultime versioni includono uno Standard Profile con requisiti più stringenti per migliorare la portabilità
  - Processori a 16 o 32 bit
  - Intero sistema collegato come un modulo
  - Supporta priorità dei task, semafori, code di messaggi, primitive di sincronizzazione con priority inheritance e priority ceiling
  - Nessun meccanismo di protezione della memoria!

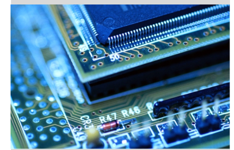
Sistemi operativi real-time
Marco Cesati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Caratteristiche comuni dei principali RTOS

- Corrispondenza agli standard: generalmente le API sono proprietarie, ma gli RTOS offrono anche compatibilità (*compliancy*) o conformità (*conformancy*) allo standard Real-Time POSIX
- Modularità e Scalabilità: il kernel ha una dimensione (*footprint*) ridotta e le sue funzionalità sono configurabili
- Dimensione del codice: spesso basati su microkernel
- Velocità e Efficienza: basso overhead per cambi di contesto, latenza delle interruzioni e primitive di sincronizzazione
- Porzioni di codice non interrompibile: generalmente molto corte e di durata predicibile
- Gestione delle interruzioni “separata”: interrupt handler corto e predicibile, ISR lunga e di durata variabile
- Gestione della memoria: possibilità di utilizzare memoria virtuale e protezione dello spazio di indirizzi kernel (disabilitabili); non esiste in genere la paginazione



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

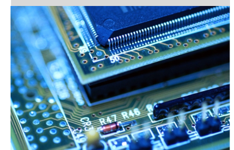
La schedulazione nei principali RTOS

- Almeno 32 livelli di priorità differenti
- Possibilità di scelta fra FIFO e Round Robin per la gestione dei task nello stesso livello di priorità
- Possibilità di cambiare la priorità a run-time
- Generalmente non supportano in maniera nativa politiche di scheduling a priorità dinamica (es. EDF) o la possibilità di integrare server a conservazione di banda, ecc.
- Offrono meccanismi di controllo dell’inversione di priorità: tipicamente priority inheritance, alcuni RTOS anche priority ceiling
- Risoluzione nominale di timer e orologi di un nanosecondo, ma l’accuratezza non è mai sotto a 100 ns a causa della latenza di gestione delle interruzioni

Alcuni meccanismi, come quelli di controllo dell’inversione di priorità, possono essere abilitati o disabilitati in fase di progetto

Sistemi operativi real-time

Marco Cesati



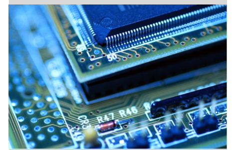
Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Comunicazione e sincronizzazione

Gli RTOS mettono a disposizione vari meccanismi per garantire la comunicazione e la sincronizzazione dei task del sistema:

- memoria condivisa
- code di messaggi, pipe, FIFO
- segnali
- mutex, semafori

Lo standard POSIX.1-2001 descrive in maniera dettagliata l'utilizzo di questi meccanismi in ambito real-time



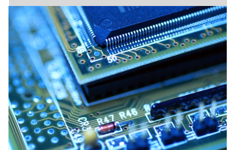
Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Gestione della Memoria

- I principali meccanismi di gestione della memoria (memoria virtuale, protezione dello spazio di indirizzamento) sono supportati da tutti i sistemi operativi moderni
- Non tutti i sistemi operativi operanti in ambito embedded e real-time offrono un supporto a questi meccanismi
- Molti di quelli che li supportano permettono di disabilitarne l'utilizzo
- Infatti non tutte le applicazioni real-time (ed embedded) necessitano di questi meccanismi che possono penalizzare il tempo di esecuzione (o la dimensione)

Sistemi operativi real-time

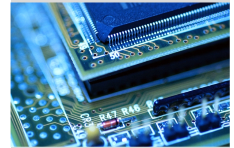
Marco Cesati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

La memoria virtuale

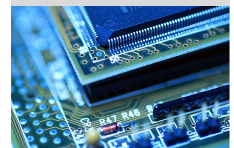
- Le applicazioni embedded e real-time sono generalmente di dimensione contenute
- Hanno un numero limitato di modalità operative (complessità limitata) e processano generalmente dati di dimensione prefissata
- La mancanza del meccanismo di memoria virtuale non è quindi un problema fondamentale
- L'overhead introdotto dal meccanismo di traduzione degli indirizzi può essere eccessivamente oneroso per un sistema embedded
- Il RTOS alloca alle applicazioni blocchi contigui di memoria
- Tutte le applicazioni progettate per i sistemi embedded sono generalmente sviluppate e testate allo stesso tempo
- I problemi di frammentazione della memoria sono minimi



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Protezione della memoria

- Molti RTOS non forniscono meccanismi di protezione dello spazio di indirizzamento per i processi del kernel e per le applicazioni
- Alcuni RTOS (RTAI, RTLinux) prevedono l'esecuzione dei task direttamente in modalità "kernel"
- Un unico spazio di indirizzamento implica una maggiore *leggerezza* e *semplicità* nel task switching $\Rightarrow$ maggiore predicibilità
- Ovviamente lo sforzo di sviluppo degli applicativi è maggiore, ma spesso è necessario



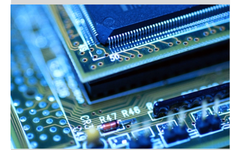
Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Quanti sono gli RTOS?

Esistono molti sistemi operativi RTOS, in particolare per il segmento embedded

Ad esempio:

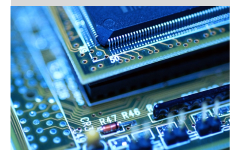
- VxWorks
- LynxOS
- QNX
- Windows CE
- Linux
- eCos
- FreeRTOS
- PikeOS
- ThreadX
- RTEMS
- Operating System Embedded
- OS-9
- Nucleus OS



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Quanti sono gli RTOS? (2)

Abassi, AMOS, AMX RTOS, ARTOS (Locamation), ARTOS (Robotu), Atomthreads, AVIX, BeRTOS, BRTOS, CapROS, ChibiOS/RT, ChorusOS, ChronOS, CMX RTOS, cocoOS, Concurrent CP/M, Concurrent DOS, Contiki, COS, CooCox CoOS, Deos, DioneOS, DNIX, GEC DOS, DrRtos, DSPnano RTOS, DSOS, eCos, eCosPro, embOS, Embox, ERIKA Enterprise, EROS, Femto OS, FlexOS, FreeRTOS, FunkOS, Fusion RTOS, Helium, HP-1000/RTE, Hybridthreads, IBM 4680 OS, IBM 4690 OS, INTEGRITY, IntervalZero RTX, ITRON, μITRON, ioRTOS, iRTOS, KolibriOS, LynxOS, MaRTE OS, MAX II,IV, MenuetOS, Micrium μC/OS-II, Micrium μC/OS-III, Milos, Microsoft Invisible Computing (MMLite), MP/M, MERT, Multiuser DOS, Nano-RK, Neutrino, Nokia OS, Nucleus OS, NuttX, On Time RTOS-32, OS20, OS21, OS4000, OPENRTOS, OSA, OSE, OS-9, OSEK, Phar Lap ETS, PaulOS, PICOS18, picoOS, Phoenix-RTOS, PikeOS, Portos,POK, PowerTV, Prex, Protothreads, pSOS, QNX, Q-Kernel, QP, RDOS, REAL/32, Real-time Linux (CONFIG_RT_PREEMPT), REX OS, RSX-11, RT-11, RTAI, RTEMS, rt-kernel, RTLinux, RT-Thread, RTXC Quadros, SafeRTOS, Salvo, SCIOPTA, scmRTOS, SDPOS, SHaRK, SimpleAVROS, SINTRAN III, Sirius RTOS, SMX RTOS, SOOS Project, Symbian OS, SYS/BIOS, Talon DSP RTOS, TargetOS, T-Kernel, THEOS, ThreadX, Trampoline Operating System (OSEK and AUTOSAR), TNKernel, Transaction Processing Facility, TRON project, TUD:OS, Unison RTOS, UNIX-RTR, μTasker, u-velOSity, VRTX, Windows CE, Xenomai, xPC Target, Y@SOS, MontaVista Linux, μnOS, uOS

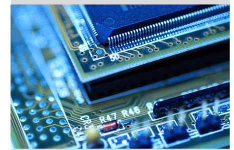


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

VxWorks

VxWorks è un RTOS commerciale prodotto dalla WindRiver (`www.windriver.com`)

- Installato su oltre 500 milioni di dispositivi
- Basato sul microkernel Wind
- È utilizzato in sistemi ad elevata affidabilità con certificazione DO-178B livello A, Arinc 653, IEC 61508
- Ha un ambiente di sviluppo basato su Eclipse (WorkBench)
- Supporta lo sviluppo cross-platform e semplifica la creazione delle toolchain di compilazione e collegamento
- Adatto sia per sistemi embedded che per sistemi con molte risorse
- Le API (interfaccie per le applicazioni) sono proprietarie
- Fornisce anche un'interfaccia di compatibilità per le API POSIX Real-Time



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

LynxOS

LynxOS è una linea di RTOS commerciali prodotti dalla LynuxWorks (`www.lynuxworks.com`)

Esistono diverse versioni, tra le quali:

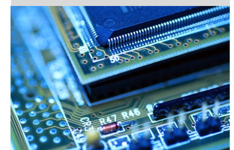
- LynxOS RTOS: per sistemi embedded
- LynxOS-SE RTOS: per sistemi partizionati con applicazioni POSIX, Linux e conformi a Arinc 653
- LynxOS-178 RTOS: per sistemi *safety-critical*, certificabili DO-178B livello A e Arinc 653

Il cuore di tutti questi SO è LynxOS RTOS:

- Utilizzato in milioni di dispositivi
- Compatibile con l'ABI (Application Binary Interface) di Linux
- Utilizza un kernel monolitico
- L'ambiente di sviluppo è tipicamente "cross-platform" e basato su Eclipse

Sistemi operativi real-time

Marco Cesati

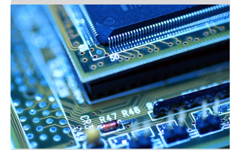


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

QNX Neutrino

QNX Neutrino è un RTOS commerciale prodotto da QNX Software Systems (`www.qnx.com`)

- Basato su microkernel di dimensione estremamente ridotta
- Molto utilizzato soprattutto nel settore automotive
- Conforme allo standard POSIX
- È progettato attorno ad un meccanismo di scambio di messaggi e segnali, implementato nel microkernel
- Tutti i servizi non essenziali sono forniti da applicazioni (server) in spazio utente
- Gestione delle interruzioni di tipo "separata"
- Le interruzioni possono essere annidate
- Le interruzioni possono essere abilitate o disabilitate

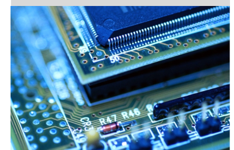


Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

eCos

eCos (`ecos.sourceware.org`) è un RTOS *open-source*

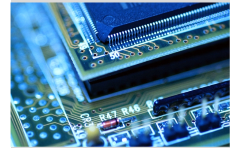
- Progettato per sistemi embedded
- È distribuito con una licenza GPL compatibile
- È fortemente modulare ed altamente configurabile
- Offre sia API proprietarie che funzioni compatibili con POSIX e μITRON
- Il kernel è solo uno dei package del sistema, e non è indispensabile per lo sviluppo di applicazioni in eCos
- Il kernel offre il supporto alla gestione delle interruzioni, alla schedulazione ed alla sincronizzazione fra thread
- È scritto in C++ ed utilizza il compilatore GNU
- I dettagli legati all'architettura sono incapsulati in un programma di interfaccia (HAL – Hardware Abstraction Layer) che supporta numerose architetture ed è scritto in C ed assembler



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

FreeRTOS

- FreeRTOS è un sistema operativo open-source con licenza GPL modificata
- Oggi molto diffuso e utilizzato sui sistemi embedded
- Supporta oltre 30 diverse architetture hardware
- Filosofia di progetto del kernel:
  - Piccolo e semplice (immagine 9 KiB ROM, 256 B RAM)
  - Scritto per lo più in C
  - Implementato in 3 file sorgente
  - Non include driver di periferiche
  - Non include lo stack di networking
  - Non include lo stack USB
  - Più simile ad una *libreria per thread* che a un SO completo
- Varianti: SafeRTOS (certificato), OPENRTOS (diversa licenza e documentazione)



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS

Windows Embedded

Microsoft offre una serie di OS embedded con footprint ridotto, ad esempio:

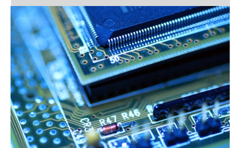
- Windows Embedded CE 6.0 e Windows Embedded Compact 7, Windows Embedded Compact 2013
- Windows Mobile: orientata ai telefoni cellulari, offre un sottoinsieme delle caratteristiche dell'Embedded CE 5.0
- Windows Phone 7: evoluzione di Windows Mobile, commercializzato nel 2011
- Windows Phone 8 (2012), 8.1 (2014), 10 Mobile (2015): basati su Windows NT

Riguardo a Windows Embedded Compact:

- **Non** è una versione "leggera" di Windows!
- Ha un kernel monolitico
- Non è compatibile con lo standard POSIX
- Il codice del kernel è distribuito con licenza "shared-source" (simile alla licenza BSD)
- Supporta solo architetture monoprocessore

Sistemi operativi real-time

Marco Cesati



Schema della lezione
RTOS
Interruzioni H/W
Schedulazione
Standard per RTOS
Caratteristiche comuni
Esempi di RTOS