

Linux e le sue applicazioni industriali, nei sistemi embedded e real-time

di Stefano Pozzi

Bologna, 16/11/04



Introduzione

Recenti indagini di mercato svolte da importanti analisti internazionali indicano come ormai Linux, il popolare sistema operativo open source tradizionalmente utilizzato in applicazioni desktop o enterprise, sia, insieme alle sue varianti real-time, il sistema operativo oggi più utilizzato per le applicazioni embedded e industriali. Questo grazie alle sue doti di robustezza, affidabilità, flessibilità, bassi costi e possibilità di personalizzazioni ed interventi a basso livello sul codice e alle recenti novità nel kernel 2.6, che lo rendono un sistema ancora più facilmente adattabile per questo tipo di sistemi.

Tutto ciò è favorito dalla crescita in termini di performance anche di piattaforme low-cost, dalla sempre più vasta disponibilità di librerie e comunità di sviluppatori, di piattaforme e strumenti di sviluppo, di linguaggi ed applicativi, dalla modularità del sistema e dalla sua estrema scalabilità su piattaforme diverse: tutto ciò rende certamente Linux un sistema operativo sempre più invitante e adatto per una grande varietà di soluzioni.

Il ruolo del sistema operativo nelle applicazioni embedded

Un sistema embedded è caratterizzato dal fatto di essere “incastonato” (in inglese appunto “embedded”) all’interno di un dispositivo o una apparecchiatura specifica, con una destinazione predefinita.

Non bisogna identificare, come spesso avviene, i sistemi embedded con quelli real-time: questi ultimi sono solo una sottocategoria, caratterizzata dal fatto di possedere specifiche stringenti in termini di tempi di risposta agli eventi.

Accanto all’applicativo, che permette al sistema di svolgere i compiti specifici per i quali è stato progettato, può o meno essere presente un sistema operativo (s.o.), inserito per avere a disposizione tutta una serie di servizi, ma a fronte di un certo impegno di risorse, anche computazionali. Se non è presente il sistema viene indicato tipicamente col termine “hard embedded”) e generalmente la scelta viene fatta a causa dei vincoli di specifica che il sistema deve rispettare.

Ciò avviene tipicamente nel caso di sistemi, non molto complessi, che non necessitano di particolare “concorrenza”, dalle risorse molto limitate (quali i sistemi ad 8 bit) o che devono essere particolarmente ottimizzati (DSP, ecc...). Esistono comunque anche s.o. ottimizzati per i DSP, come ad esempio VSPWorks di Wind River.

Tuttavia l’uso di un s.o. (sia esso nativo o open-source) comporta in genere notevoli vantaggi, tra cui il supporto per il multi-threading (spesso anche per il multi-processing), il supporto di protocolli di comunicazione, memorie di massa, librerie grafiche, strumenti di debugging, nonché, spesso, la maggiore facilità di ‘porting’ verso hardware differenti.

Questo ultimo punto è abbastanza importante per i sistemi embedded, poiché gli hardware utilizzabili per questo tipo di dispositivi (microcontrollori, memorie, periferiche) possono andare rapidamente in obsolescenza o venire superati da tecnologie meno costose, rendendo necessaria o opportuna la riprogettazione con componenti più facilmente reperibili o moderni.

A questo proposito si può far notare che le esigenze di porting sono state fortemente prese in considerazione nella progettazione di Linux, cercando di isolare in punti ben specifici le dipendenze dalla macchina.

Il real-time

Real-time non è sinonimo di veloce, bensì di deterministico, ossia prevedibile e garantibile a priori. Le specifiche più stringenti (*hard real-time*) consistono nell’assicurare un tempo massimo di risposta (*deadline*) prevedibile sempre inferiore ad un certo limite fra il presentarsi di un evento la generazione dell’effetto.

In un sistema “*soft real-time*” invece al software non è richiesto di rispondere in tempi certi pena la morte del sistema sotto controllo. Semplicemente si vuole che il software risponda in tempo nella maggior parte dei casi. Un esempio tipico è costituito dalla riproduzione audio o video: se un brano musicale si interrompe per un attimo la conseguenza è solo un po’ di fastidio per l’utente, nulla di irreparabile.

S.O. open source e proprietari

La scelta di sistemi operativi per il mondo embedded è molto varia, ma le analisi di mercato (fonti: [VDC], [EDC], [GART]) indicano chiaramente che la tendenza è verso l'affermazione di sistemi open-source, tra cui Linux in particolare (almeno per i nuovi progetti).

Un tempo Linux non era considerato adatto per applicazioni embedded e soprattutto per prestazioni in tempo reale, semplicemente poiché non era stato progettato per questo scopo, essendo derivato da Unix.

La scelta ricadeva solitamente sullo sviluppo in proprio del sistema operativo o sull'adozione di un sistema commerciale (quali ad esempio [VxWorks](#), [Neutrino](#), [LynxOS](#), [Integrity](#), [µC/OS-II](#), ecc...), questi ultimi specificamente progettati ed ottimizzati per sistemi dalle risorse limitate (memoria ridotta, assenza di memoria di massa, ecc...).

Esistono ancora oggi valide motivazioni per scegliere un s.o. proprietario, tra le quali la "reale" natura real-time di molti di questi sistemi che permette di ottenere prestazioni in tempo reale anche molto spinte, la loro migliore ottimizzazione per le applicazioni e gli hardware tipici del mondo embedded ed il fatto che in molti casi sono certificati per la conformità all'uso in ambiente aerospaziale, difesa o semplicemente per la conformità POSIX, spesso richiesta dai committenti appartenenti ad organizzazioni governative.

Recentemente tuttavia il Pinguino è riuscito a penetrare anche nelle mura del Pentagono. Infatti il comitato tecnico del progetto [Future Combat Systems](#) (www.darpa.mil) della US Army, che studia la creazione di nuovi sistemi di difesa intelligenti, ha scelto Linux come s.o. di riferimento (scartando ad esempio Windows, VxWorks e LynxOs) [PI].

La scelta tra un sistema proprietario o uno open source (e quindi privo di royalties) non si riduce solo ad un problema di costi: talvolta sono la necessità di innovazione degli impianti industriali, le richieste del mercato e le competenze tecniche esistenti in azienda a condizionare la scelta e a far pendere l'ago della bilancia da una parte o dall'altra. [EOE]

"Un caso frequente - dice Marco Cavallini, titolare e fondatore di Koan Software - sono ad esempio le aziende che provengono da realtà in cui sono utilizzati microcontrollori a 8 o 16 bit e che desiderano passare ad un sistema operativo embedded per fornire interconnettività ai propri prodotti, senza però dover affrontare l'onere di realizzare, per esempio, uno stack Tcp/Ip o un dispositivo Usb; in questo caso notiamo una maggior predisposizione ad avere il controllo totale del sistema, inteso come disponibilità completa del codice sorgente, e in tal senso la migliore offerta viene da Linux".

Buona parte dei sistemi operativi (in passato e ancora oggi) viene comunque sviluppata 'in casa'. Questo è senz'altro ancora più vero in ambito accademico e nel mondo della ricerca. Linux ha fatto il suo ingresso nei sistemi embedded proprio secondo questo approccio. Grazie alla libera disponibilità dei sorgenti e alla sua alta modularità, molte aziende, enti di ricerca e 'comunità open source', anziché sviluppare sistemi in proprio partendo da zero, hanno cominciato a 'ritagliare' Linux secondo le proprie esigenze per ottenerne versioni adatte anche ad applicazioni embedded. Questo rimane ancora oggi l'approccio preferito per l'adozione di Linux (fonte: [LD2]), nonostante la presenza sul mercato di numerose distribuzioni 'commerciali'. La maturazione di queste ultime e la sempre maggiore disponibilità di ambienti di sviluppo completi e sofisticati farà comunque probabilmente calare questa tendenza.

Certamente oltre a queste motivazioni 'basilari', la scelta può ricadere su Linux anche per le sue note doti di robustezza, affidabilità, flessibilità, bassi costi e possibilità di personalizzazioni ed interventi a basso livello sul codice, che sono probabilmente le caratteristiche più importanti richieste da una applicazione industriale ed, in particolare, in un sistema orientato alla robotica ed al controllo. Oppure più semplicemente a volte, al di là della modificabilità del codice, in alcune applicazioni, può essere importante semplicemente anche solo la possibilità di "vedere cosa fa esattamente" il codice.

Un fatto non trascurabile è che si tratta di un sistema operativo relativamente 'standard' (sebbene in mille varianti), cioè abbastanza diffuso e conosciuto da un grosso numero di sviluppatori 'Linux-literate' (grazie anche

Cos'è POSIX?

POSIX significa "Portable Operating System Interface" ed è uno standard prodotto dalla IEEE e riconosciuto al livello mondiale.

Il supporto di POSIX assicura la portabilità del codice tra i sistemi e molto spesso è richiesto nei contratti di fornitura, soprattutto verso le istituzioni.

Molte volte i sistemi sono compatibili solo con alcune parti dello standard POSIX (*POSIX compliance*) a differenza di quelli 100% compatibili (*POSIX conformance*).

In particolare vengono poste delle specifiche sul real-time, sull'implementazione processi e dei threads, sulla segnalazione tra i processi e tra i threads, sui timers, sulla separazione dei namespaces, ecc...

Nel kernel 2.6 di Linux è stata migliorata molto il supporto a POSIX, soprattutto per quanto riguarda threads, timers e segnali.

alla sua derivazione Unix ed alla sua diffusione in ambito universitario) e dotato di numerose comunità 'virtuali' (su Internet) spesso disponibili a fornire gratuitamente (secondo il modello del software libero) informazioni, documentazione e supporto tecnico.

Per le aziende è molto più facile trovare un ingegnere con almeno qualche conoscenza di sviluppo su Linux, che uno che conosca, ad esempio, Neutrino di QNX (se non altro per la presenza di numerosi testi sull'argomento). Un ultimo, ma importante, punto a favore di un sistema open source è l'indipendenza dal produttore: oltre alla possibilità di poterlo sviluppare e supportare in proprio a partire dai sorgenti (scelta da valutare attentamente dal punto di vista economico), sono presenti sul mercato molti produttori di distribuzioni Linux embedded e aziende di consulenza che offrono supporto, addestramento ed altri servizi, verso cui rivolgersi in qualsiasi momento e verso cui cambiare in caso di necessità o di offerte migliori.

L'affermarsi di Linux come s.o. embedded è stato favorito anche dalla crescita in termini di performance di piattaforme hardware a basso costo (per cui il vincolo dell'adozione di un sistema particolarmente ottimizzato viene in molti casi a cadere), dalla sempre più vasta ed costantemente aggiornata disponibilità di librerie, di protocolli e di applicativi di supporto (web server, database, ecc...), nonché di piattaforme ed ottimi strumenti di sviluppo (open source o commerciali), dalla possibilità di utilizzare numerosi linguaggi ed applicativi ed infine dalla estrema scalabilità di Linux su piattaforme diverse, anche dalle risorse limitate.

A proposito di questo ultimo punto già da qualche tempo Linux è stato adattato per essere inserito anche in sistemi embedded di modeste dimensioni e con risorse ridotte. La 'distribuzione' uCLinux (open source, www.uClinux.org), ad esempio, ha avuto grande diffusione grazie al fatto di essere particolarmente 'leggera' di poter funzionare anche su microcontrollori sprovvisti di MMU (questo, ovviamente, introduce delle limitazioni nel 'multi-process' in quanto non è possibile distinguere indirizzi logici da indirizzi fisici). Questa caratteristica è stata inserita anche nel nuovo kernel 2.6 (ereditandola direttamente da uCLinux), insieme a molti altri accorgimenti che sicuramente aprono decisamente le porte per l'affermazione del pinguino nel mondo embedded.

Le novità del kernel 2.6: "born to be embedded"

Il più grosso problema riguardo al real-time di Linux è sempre stato che i cambi di contesto (ad esempio il passaggio ad un processo ad elevata priorità) possono (o meglio *potevano*, fino all'avvento del kernel 2.6) avvenire solo al passaggio dallo spazio del kernel a quello utente.

Linux 2.6 introduce miglioramenti che lo rendono molto più valido che in passato in applicazioni dove il tempo di risposta è importante:

- sono stati inseriti dei punti di "preemption" nel kernel, cioè punti nei quali può girare lo scheduler. Fino ad ora ciò non era possibile, principalmente per motivi di semplificazione nel preservare l'integrità delle strutture dati del kernel stesso, ma ciò poteva provocare problemi in quanto un processo a priorità molto alta (e dalle specifiche temporali stringenti) poteva essere ritardato di molto in attesa che una system-call di un altro processo terminasse. Per risolvere questo problema uno dei metodi utilizzati erano le cosiddette "*preemption patch*" e "*low-latency patch*", ora introdotte ufficialmente nel kernel.
- è stato ottimizzato lo scheduler, implementando algoritmi più efficienti in modo da ridurre l'overhead introdotto dal s.o., soprattutto in presenza di molti "task".
- il sistema può girare anche su sistemi privi di memoria virtuale (allargando di molto la base di processori supportati, cioè tutti quelli sprovvisti di MMU). Infatti un software che deve rispondere entro certe "deadline" è in un certo senso incompatibile con la memoria virtuale, a causa dell'impredicibile ed eventuale lenta elaborazione dei page faults, che possono rovinare la prontezza di risposta del sistema.
- sono state introdotte nuove primitive di sincronizzazione, che permettono di usare i "mutex" senza ricorrere a chiamate di sistema (che introducono sempre un certo overhead).
- sono stati introdotti nel kernel il supporto per le segnalazioni e per i timer ad alta risoluzione POSIX (oltre ad una ottimizzazione del meccanismo dei thread POSIX).

Un altro miglioramento a favore dei sistemi embedded è l'introduzione del concetto di "sub-architettura": i componenti del sistema sono chiaramente separati, quindi se ad esempio deve cambiare il codice per gestire una interruzione poiché è cambiato l'hardware, il codice specifico può essere modificato con un minimo impatto sul resto del sistema. È inoltre possibile eliminare completamente il codice per gestire video, tastiera, ecc... ove questi non risultino necessari. Dal lato opposto è possibile inserire, in quanto supportati nativamente, sistemi di comunicazione quali USB 2.0 e Bluetooth.

Il real-time di Linux

Non bisogna comunque dimenticare che nonostante tutti questi miglioramenti Linux non è un vero sistema hard real-time. Per ottenere prestazioni real-time occorre ancora ricorrere a patch del kernel, delle quali una delle più utilizzate è RTAI, sviluppata dal Politecnico di Milano (www.rtai.org).

Si tratta di una delle soluzioni attualmente più efficaci per rendere Linux un sistema veramente real-time. Essa prevede che un secondo kernel, di tipo *hard real time*, prenda il controllo effettivo della macchina e faccia girare Linux come una propria applicazione a priorità più bassa.

Inoltre il kernel Linux non ha più il controllo diretto sull'abilitazione e la disabilitazione delle interruzioni: queste vengono intercettate da RTAI che sostituisce gli handler Linux con dei propri dispatcher.

Molte distribuzioni embedded commerciali includono RTAI: una di queste è ad esempio K-Linux della italiana KOAN Software (www.koansoftware.com), "royalty free" e rilasciata sotto licenza open source GPL, specifica per applicazioni industriali e completa di strumenti di sviluppo e supporto in italiano.

Linux nelle applicazioni industriali e nell'automazione

L'uso di Linux e delle sue varianti real-time è di grande interesse per applicazioni, anche significative, di automazione industriale, di supervisione, di robotica e nei sistemi di controllo distribuito.

La Yacme di Bologna (www.yacme.it) ha ad esempio realizzato un sistema di **supervisione** e gestione centralizzata di allarmi, di impianti e di grandezze campionate per una grossa azienda municipalizzata nel settore della distribuzione energetica. Questo sistema, chiamato WEST, sfrutta le capacità di comunicazione offerte dalle moderne reti Internet e risponde all'esigenza di gestire in modo standardizzato tipologie di impianti eterogenee, architetture differenti, tipologie di controlli eterogenee, differenti sistemi di rilevamento e attuazione con diverse interfacce, canali di comunicazione e trasmissione dati differenti. Il tutto basato su tecnologie open source (sistema operativo, web server, database, linguaggi e strumenti di sviluppo).

Nel campo dell'**automazione** si può sfruttare Linux, oltre che come sistema da inserire in un hardware dedicato, anche come piattaforma di sviluppo e prototipazione rapida su piattaforma PC.

Un *benchmark* in questo senso è stato sviluppato ad esempio dal Dipartimento di Elettronica, Informatica e Sistemistica (DEIS, www-lar.deis.unibo.it) dell'Università di Bologna, che ha voluto verificare le possibilità di Linux nelle sue varianti real time ed, in generale, del software Open Source, nell'ambito dell'automazione industriale.

Per questo è stata realizzata, tra le altre, una applicazione di controllo assi, che è stata implementata con Matlab/Simulink su una architettura PC basata su real time Linux (RT-Linux) e fornita di una scheda I/O capace di essere integrata facilmente all'interno dell'ambiente di simulazione. In questo modo, il personal computer usato nella fase di simulazione diventa anche il sistema che implementerà la versione definitiva del software o, almeno, un suo prototipo funzionante. Infatti una volta raggiunte prestazioni soddisfacenti in simulazione, il controllore viene testato sul sistema reale, senza abbandonare l'ambiente di simulazione. Attraverso il *real time Workshop* (RTW) di Matlab, infatti, è possibile generare del codice C (*hard*) real time a partire dallo schema di simulazione che può essere mandato in esecuzione sul kernel real time di Linux.

Tutta la procedura può essere portata avanti su un semplice PC dotato di opportune interfacce di I/O verso il sistema reale. La chiave risiede nell'utilizzo di hardware standard e di basso costo e di un sistema basato su real time Linux, capace di prestazioni di ottimo livello nei processi di automazione non troppo spinti.

Un'altra applicazione è stata realizzata, sempre dal DEIS, nel campo della robotica, per il pilotaggio di un robot industriale a 6 gradi di libertà ed una pinza a 3 gradi di libertà espressamente progettata per una possibile applicazione all'interno del PaT, il "Payload Tutor" proposto dall'A.S.I. (Agenzia Spaziale Italiana).

Nell'ambito dei **sistemi di controllo distribuiti**, a tutt'oggi si può riscontrare che non vi sono sistemi e macchine realmente funzionanti in modo distribuito. Soprattutto mancano strumenti metodologici generali e applicativi specifici in grado di aiutare i progettisti e i tecnici nel progetto di ambienti di controllo in tempo reale per sistemi distribuiti.

Il progetto di ricerca nazionale OASIS (software Open source per l'Automazione e i SISTemi distribuiti), coordinato dall'Università di Bologna, intende quindi affrontare queste tematiche (<http://www-lar.deis.unibo.it/oasis>).

Un seminario sull'argomento

Su queste tematiche è stato incentrato il seminario organizzato dalla Fondazione per l'Ingegneria e dall'Ordine degli Ingegneri di Bologna dal titolo "**Linux e le sue applicazioni industriali, nei sistemi embedded e real-time**" che si è tenuto a Bologna il 16 Novembre 2004, presso la sala Convegni dell'Ordine degli Ingegneri di Bologna. Per maggiori informazioni e materiale visitare il sito www.ingpozzi.it/semlinux.

A questo scopo sembra particolarmente interessante la possibilità di sviluppare nuovi strumenti di controllo in tempo reale basati su ambienti software "open source" (tipicamente Linux e le sue varianti real time, come RTAI e RT-Linux), che integrino notevoli capacità computazionali, possibilità di facili riconfigurazioni in ambienti distribuiti, capacità di simulazione di sistemi dinamici e algoritmi di controllo non banali e funzionanti su piattaforme hardware facilmente reperibili in commercio e di costo contenuto (PC).

Un futuro roseo: i giganti si schierano con Linux

Recenti indagini di mercato, tra cui quelle di [VDC](#) (Venture Development Corp.) [VDC][VDC2], [Gartner](#) [GART][GART][GART][GART] e [EDC](#) (Evans Data Corp.), indicano che gli sviluppatori stanno sempre più abbandonando sistemi operativi proprietari (tra cui Windows CE, VxWorks, ecc...) e sistemi operativi "fatti in casa" per abbracciare Linux embedded nei nuovi progetti. L'analista di VDC Steven Balacco spiega: "In passato l'elettronica di consumo e l'automazione industriale sono stati i maggiori mercati per Windows CE. Comunque adesso si intravede un trend verso l'uso di Linux in questo tipo di dispositivi".

Il supporto dei produttori di hardware

Anche i produttori di piattaforme hardware per l'integrazione nei sistemi embedded stanno fornendo sempre maggiore supporto a Linux, mettendo a disposizione kit di sviluppo che realmente fanno risparmiare tempo agli sviluppatori e rendono lo start-up di una soluzione che fa uso di Linux molto più veloce. Questi kit infatti includono, oltre a tutto l'hardware necessario, anche delle distribuzioni Linux già preconfezionate e pronte all'uso. È il caso ad esempio delle schede CPU della Diamond Systems e della Arcom. Entrambe le marche sono distribuite in Italia da Sistemi Avanzati Elettronici (www.sisav.it).

In effetti la costituzione nella scorsa estate del CELF (Consumer Electronics Linux Forum, www.celinuxforum.org) da parte di parecchi giganti del settore dell'elettronica di consumo (Hitachi, Philips, Sony, NEC, Sharp ed altri) indica chiaramente quale direzione essi intendono assumere, almeno in questo tipo di dispositivi e quale sia il destino (roseo) di Linux (perlomeno nel mercato consumer). Il CELF intende risolvere i vari problemi tecnici che fino ad ora hanno frenato l'avanzata di Linux in questo settore (tempo di avvio, real-time, richieste di memoria, gestione dei consumi, ecc...), promuovendo CE Linux (il proprio s.o. embedded) come uno standard per le piattaforme embedded destinate al mercato di massa.

Insomma: CE Linux si direbbe ideato per rivaleggiare con Windows CE di Microsoft (che aveva proprio in questo ambito la sua roccaforte) per girare su dispositivi quali palmari, telefoni cellulari, PDA, player MP3, televisioni digitali, ecc...

Al progetto contribuiscono, oltre ovviamente ai fondatori, anche diversi leader dell'elettronica di consumo e dei sistemi operativi, quali Nokia, Motorola, IBM, ARM, Wind River, Metrowerks e MontaVista.

Del resto anche molti dei maggiori produttori di sistemi operativi proprietari si sono mossi o si stanno muovendo verso Linux, fornendo accanto al loro s.o. "tradizionale", anche sistemi basati su Linux.

Recentemente ha suscitato grande interesse e scalpore il recente accordo [WR1] tra Wind River (www.windriver.com, produttrice del blasonato VxWorks) e Red Hat, per lo sviluppo comune di una distribuzione Linux (Red Hat Embedded Linux) e di un ambiente di sviluppo finalizzato all'ottimizzazione del software (DSO – Device Software Optimization) per il mercato dei dispositivi embedded. Man mano che aumenterà la complessità delle applicazioni embedded (e questo ormai è un dato di fatto poiché sono richiesti dispositivi via via più 'intelligenti' e user-friendly) saranno sempre più necessarie anche soluzioni e piattaforme per l'ottimizzazione del software. Il tutto rappresenterà la base (foundation) delle Wind River Platform che supportano Linux (per approfondimenti si può vedere l'articolo "[Wind River partners with Red Hat on embedded Linux](#)" su LinuxDevices.com [LD3][LD4]).

Un altro esempio è LynuxWorks che, accanto al suo storico LynxOS (reso peraltro compatibile a livello di binario con Linux), fornisce anche la distribuzione Blue Cat Linux (www.lynxworks.com).

Tutto ciò conferma che il fenomeno Linux Embedded non è solo una moda o qualcosa di cui si parla, ma è una realtà (una "unstoppable force", come sostiene MontaVista) da prendere seriamente in considerazione per nuovi progetti.

Un testa a testa

In questo scenario Microsoft non sta certo a guardare, poiché si tratta di un mercato che si stima arriverà presto, nel suo complesso, al miliardo di dollari (tra sistemi operativi, relativi strumenti di sviluppo e servizi).

Con il rilascio della [versione 5.0 di Windows CE](#) Microsoft ha varato un nuovo programma di licenze che (oltre a ridurre drasticamente le royalties) permetterà a tutti coloro che dispongono di licenza run-time di Windows CE, di

modificarne il codice sorgente nei propri dispositivi conservandone la proprietà, cioè senza l'obbligo di condividere le modifiche con Microsoft.

Non si tratta comunque di open source (è infatti detto "Shared Source"): non è possibile usare il codice di Windows CE per realizzare un nuovo sistema operativo o integrarne delle sue parti in altri prodotti software. C'è anche da dire che non tutto il codice viene reso pubblico, anche se lo è un buon 60-70% (tra cui kernel, file system, interfaccia grafica ed altri).

Sempre ispirandosi ai principali motivi di successo di Linux, oltre a ciò Microsoft ha cercato di rendere il sistema (tra l'altro dotato di prestazioni real-time) il più flessibile e modulare possibile ed ha introdotto una più ampia base di driver di periferica.

Tuttavia, se Linux embedded non richiede un processore particolare, ma è in grado di girare praticamente su qualsiasi CPU (ricompilandolo), per quanto riguarda Windows CE il discorso è un po' diverso. Infatti esso è meno indipendente dal processore ospite (solo un certo numero di processori sono supportati, di solito di casa Intel) ed è un sistema ottimizzato e particolarmente adatto per una specifica classe di dispositivi, cioè quella dei PC palmari o dei PDA (Personal Digital Assistant).

Stando alle indagini di mercato, spesso contrastanti, la gara per il primato nella diffusione dei sistemi Windows o di sistemi Linux nel mercato embedded rimane comunque un testa a testa, con risultati differenti a seconda dei segmenti verticali di mercato e dell'area geografica (ad esempio in Asia, patria dell'elettronica di consumo, sembra prevalere Linux, mentre in altre aree geografiche la potenza del marchio Microsoft sembra aiutare Windows).

Secondo una ricerca condotta da LinuxDevices.com Windows sembra prevalere in settori quali PDA, palmari e tablet PC, mentre Linux riscuote maggiore successo nella robotica, nei dispositivi audio/video e nelle network appliances (si veda l'interessante articolo "[Windows, Linux grapple in Great Gadget Smack-Down!](#)" su LinuxDevices.com) [LD1].

E i costi reali?

Quella dei costi totali di sviluppo, di 'ownership' e del 'time-to-market' di una soluzione basata su Linux piuttosto che su di un sistema proprietario (tra cui in particolare Windows embedded) è una questione molto dibattuta.

Spesso dovendo iniziare un nuovo progetto per lo sviluppatore si pone il dilemma di scegliere se usare Linux Embedded o Windows Embedded (le due soluzioni dominanti per varie tipologie di applicazioni, supponendo che entrambe siano tecnicamente adatte).

Da alcune parti si sente dire (anche in base ad indagini di mercato) che alla fine i costi totali (in termini di tempo e di denaro) di uno sviluppo con Linux sono maggiori rispetto ad uno sviluppo fatto con un sistema proprietario (Windows in particolare). A parte che le statistiche non sono facilmente generalizzabili, né facili da realizzare correttamente, poiché il tempo/costo di sviluppo va messo in relazione con la difficoltà del progetto, i fattori da considerare sono numerosi e l'analisi non è facile (poiché come abbiamo visto ci sono settori più o meno adatti per l'una o per l'altra soluzione), questo può essere senz'altro vero se si decide di sviluppare una distribuzione in proprio. Infatti, il fatto che Linux sia disponibile gratuitamente può trarre in inganno: l'assemblaggio e la manutenzione di una distribuzione e la realizzazione di personalizzazioni (peraltro impossibili con i sistemi proprietari) possono richiedere un grosso numero di 'ore-uomo', aumentando anche il time-to-market.

Tuttavia rimangono a favore di Linux altri fattori: è royalty-free (cosa che comunque spesso ha impatto solo sui grossi volumi), è meno costoso trovare sviluppatori e formarli (a parte forse il caso di Windows), sono disponibili tools di sviluppo gratuiti, sono disponibili distribuzioni off-the-shelf ('commerciali' o meno, comunque open source), da cui eventualmente partire come base nel caso si voglia sviluppare una distribuzione personalizzata.

Comunque, come si diceva più sopra, spesso il costo di sviluppo non è il fattore che fa la differenza (in entrambi i tipi di soluzioni). Molti altri fattori sono importanti (oltre a quelli già visti) tra cui la reale differenziazione del prodotto dalla concorrenza ed il suo costo finale (chiaramente il prodotto sarà più competitivo se il sistema operativo può girare su un hardware più economico).

Si vedano su questo argomento gli interessantissimi commenti all'analisi del 2003 condotta da Embedded Market Forecaster (finanziata da Microsoft), che compara i costi di sviluppo sulle piattaforme Windows Embedded rispetto alle piattaforme Linux Embedded:

- "[Windows device development faster, cheaper than Linux?](#)", di John Lettice
 - "[Concerns about embedded Linux](#)", di Tom Williams
 - "[Another perspective to the report on cost of development](#)", di Rick Lehrbaum
- tutti disponibili, insieme ad altri, su [LinuxDevices.com](#).

Conclusioni

Nella scelta del sistema operativo per un sistema embedded, Linux è sicuramente una alternativa da considerare, in quanto:

- possiede note doti di robustezza, sicurezza ed affidabilità;
- permette di vedere cosa fa esattamente il codice ed eventualmente è possibile modificarlo o integrarlo con le parti mancanti alle esigenze specifiche;
- grazie alla disponibilità dei sorgenti, assicura una completa manutenibilità, un completo controllo del codice da parte del progettista e permette di svincolarsi da produttori specifici;
- è privo di royalties sui prodotti finiti ed è eventualmente scaricabile gratuitamente da internet;
- è abbastanza diffuso e conosciuto da un'ampia comunità di sviluppatori, grazie anche all'ampio uso che ne viene fatto per la didattica e le ricerca;
- sono disponibili 'porting' su molte CPU differenti;
- è estremamente 'ritagliabile' e configurabile e quindi può essere reso relativamente piccolo, almeno rispetto alle caratteristiche e prestazioni che fornisce;
- è abbastanza rispondente agli standard POSIX, garantendo allo sviluppatore una certa continuità ed uniformità nel tempo delle API utilizzate e facilitando eventualmente il porting da altri sistemi conformi allo stesso standard;
- grazie al lavoro dell'ampia comunità di sviluppatori sono disponibili innumerevoli driver, utilities ed applicazioni pronti per l'uso o da cui partire per i propri sviluppi;
- l'ambiente di sviluppo è disponibile gratuitamente e sono presenti tools e librerie paragonabili in molti casi a quelli commerciali;
- sono comunque disponibili distribuzioni commerciali che forniscono piattaforme di sviluppo complete e molto potenti e possibilità di training e supporto diretto, al pari dei classici s.o. proprietari;
- rende disponibili un grosso numero di protocolli di comunicazione, rendendolo particolarmente adatto alle applicazioni nei settori delle reti di telecomunicazione (gateways, internet appliances, telefonia, firewalls, VPN, ecc...), nei quali è particolarmente diffuso, grazie anche alla presenza di una certa 'standardizzazione' fornita dalle specifiche "Carrier Grade Linux";
- molti produttori di hardware (ed anche grosse software house) si stanno muovendo per fornire il supporto diretto di Linux per i propri prodotti.

Fonti:

- [VDC] ["VDC data comparing Linux and Windows embedded development"](http://linuxdevices.com/articles/AT7047575756.html), LinuxDevices.com
<http://linuxdevices.com/articles/AT7047575756.html>
- [VDC2] ["Linux squeezing WinCE, VxWorks out of devices"](http://www.linuxdevices.com/news/NS4932714398.html), LinuxDevices.com
<http://www.linuxdevices.com/news/NS4932714398.html>
- [EDC] ["EDC: Embedded Linux remains #1 choice of developers"](http://linuxdevices.com/articles/AT4787985721.html), LinuxDevices.com
<http://linuxdevices.com/articles/AT4787985721.html>
- [GART] ["Gartner/Dataquest: Embedded Linux now number one in Asia"](http://linuxdevices.com/news/NS8173104789.html), LinuxDevices.com
<http://linuxdevices.com/news/NS8173104789.html>
- [PI] ["Il pinguino nelle armi del futuro"](http://punto-informatico.it/p.asp?i=43267), Punto-informatico.it
<http://punto-informatico.it/p.asp?i=43267>
- [LD1] ["Windows, Linux grapple in Great Gadget Smack-Down!"](http://www.linuxdevices.com/news/NS6877764952.html), LinuxDevices.com
<http://www.linuxdevices.com/news/NS6877764952.html>
- [LD2] ["Snapshot of the Embedded Linux market -- March, 2004"](http://linuxdevices.com/articles/AT8693703925.html), LinuxDevices.com
<http://linuxdevices.com/articles/AT8693703925.html>
- [LD3] ["Wind River partners with Red Hat on embedded Linux"](http://linuxdevices.com/news/NS6013277917.html), LinuxDevices.com
<http://linuxdevices.com/news/NS6013277917.html>
- [LD4] ["Embedded Linux promise helps boost Wind River earnings"](http://linuxdevices.com/news/NS2588884818.html), LinuxDevices.com
<http://linuxdevices.com/news/NS2588884818.html>
- [WR1] ["Partnership Wind River-Red Hat per sviluppare una soluzione basata su Linux in grado di ottimizzare il software dei dispositivi embedded"](http://cdn.windriver.com/it/press/pr.html?ID=118), WindRiver.it
<http://cdn.windriver.com/it/press/pr.html?ID=118>
- [EOE] "Linux 'si fa largo' nel mondo embedded", di G.Fusari, Elettronica Oggi Embedded, Luglio 2004
- [MS1] ["Windows CE .NET"](http://msdn.microsoft.com/embedded/prevver/ce.net/), Microsoft.com
<http://msdn.microsoft.com/embedded/prevver/ce.net/>