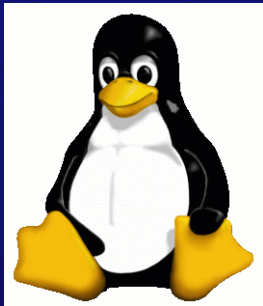


I sistemi *embedded* e il Software Libero

Rodolfo Giometti <giometti@gnudd.com>

Il sistema *GNU/Linux*

Parliamo di *GNU/Linux* e non solo di *Linux* perché *Linux* di per se è solo il nucleo del sistema.



+



=



Quando parliamo di «sistema operativo» intendiamo quindi l'insieme del nucleo (*Linux*) più tutta una serie di applicazioni di gestione, utilità, ecc. (*GNU*) del sistema stesso.

Nel caso dei sistemi *embedded* generalmente possiamo parlare di «distribuzione *embedded GNU/Linux*» poiché abbiamo non solo il sistema *target* di quel tipo ma anche i *tool di sviluppo* appartengono al progetto *GNU* (e molte volte anche il sistema *host*).



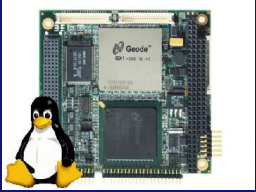
Distribuzione
del *target*



Toolchain



Host



Con il termine *target* si intende il sistema *embedded* sul quale gira la distribuzione *embedded*, ed è quindi il sistema finale da sviluppare.



Con il termine *host* si intende il sistema di sviluppo, cioè è la macchina che utilizziamo per *cross-compilare* e che ci aiuta nello sviluppo.

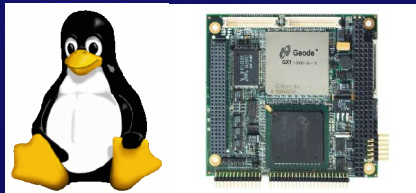


La *toolchain* è invece l'insieme delle *binutils*, delle librerie (*libc*, ecc.) e del *cross-compilatore*.

Il *cross-compilatore* è un compilatore che produce codice oggetto per una macchina diversa da quella su cui gira; risulta fondamentale quando il *target* è di un'architettura diversa dall'*host*.

La distribuzione GNU/Linux embedded

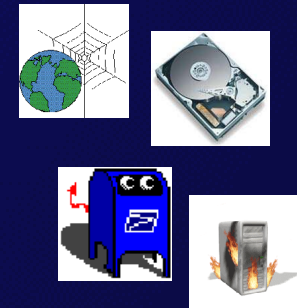
Quando scarichiamo o acquistiamo una distribuzione *embedded GNU/Linux* solitamente ci vengono fornite diverse componenti.



Versione di *Linux* adattato alla scheda



Toolchain per la cross-compilazione

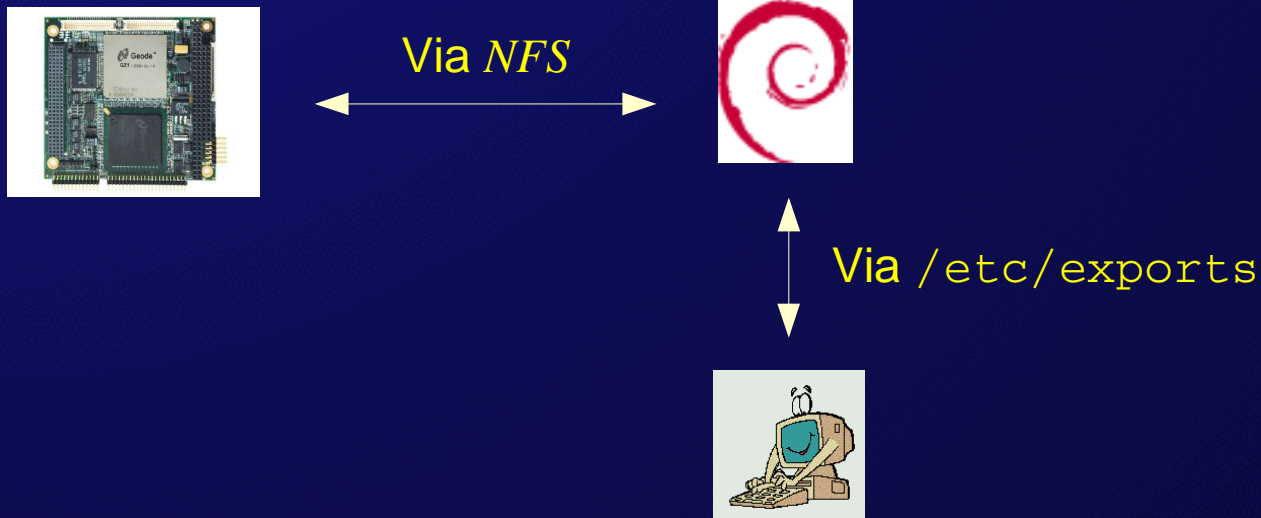


Applicativi

- ◆ La particolare versione di *Linux* deve (o dovrebbe) supportare i *driver* per tutte le periferiche del sistema. Nel caso che questo non avvenga occorre farsela (dal venditore) o farla (in proprio o meno) integrare in modo che si possa controllare completamente la macchina.
- ◆ Una buona *toolchain* deve (o dovrebbe) includere anche altre librerie di utilità oltre alla `libc`, ad esempio le librerie grafiche (`libX`, `libgtk`, ecc.).
- ◆ Un buon supporto per lo sviluppo è anche dato dalla disponibilità o meno di tool specifici come: `gdb`, `gdb server`, `strace`, `netcat`, `tcpdump`, ecc..

Se la nostra macchina *embedded* ha poi risorse sufficienti (una scheda di rete, 32MB ed una console su LCD/Monitor) potremmo anche pensare di installare una distribuzione completa magari attraverso *NFS*!

La distribuzione Debian, per esempio, essendo disponibile per molte architetture è un ottimo candidato.



Il nucleo (o *kernel space*)

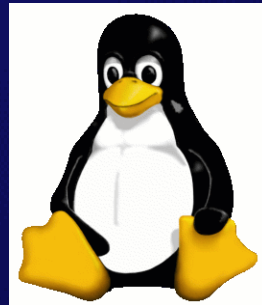
Linux è fondamentalmente un nucleo in stile *UNIX* e pertanto ne possiede tutte le caratteristiche rilevanti.



Gestione
dell'hardware



Gestione dei
processi



Meccanismi
di IPC



Gestione del
filesystem

Quando pensiamo ad un sistema *embedded* però alcune di queste caratteristiche potrebbero sembrare delle pecche, ma...

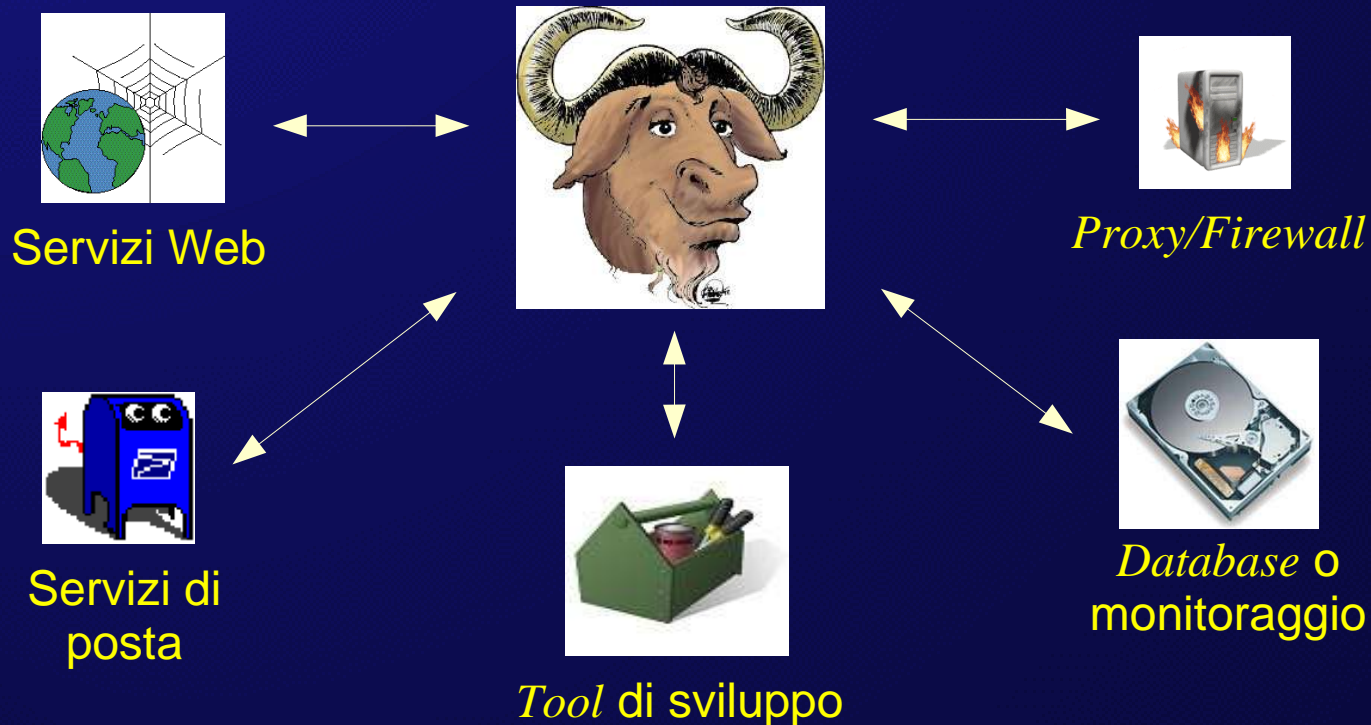
- ◆ Avere il nucleo distinto dai processi (applicazioni) permette di modularizzare il sistema definendo bene i compiti dei vari «pezzi», evita che il *crash* di una parte coinvolga il tutto e per aggiornare una funzionalità basta sostituire solo un modulo.
- ◆ Il *filesystem* può risiedere su diversi dispositivi: *Flash*, *ROM*, *RAM*, *Compact Flash*, ecc.. Può anche stare sulla macchina host (*NFS*) il che semplifica i tempi di sviluppo!

Linux ha poi come indubbi vantaggi:

- ◆ Ogni *driver* di sistema può essere definito come un modulo di nucleo.
- ◆ Nei sistemi con *MMU* è possibile utilizzare tutta una serie di meccanismi di protezione ed astrazione della memoria (memoria virtuale) nonché tecniche di accesso diretto ai dispositivi (`mmap()`).
- ◆ I meccanismi di *IPC* sono molteplici (*pipe*, *fifo*, *socket*, ecc.) e ben standardizzati.
- ◆ Essendo un sistema *UNIX-like* possiede l'astrazione file per cui ogni cosa nel sistema è visto come un file il che permette di risolvere molti problemi in maniera veloce ed elegante.

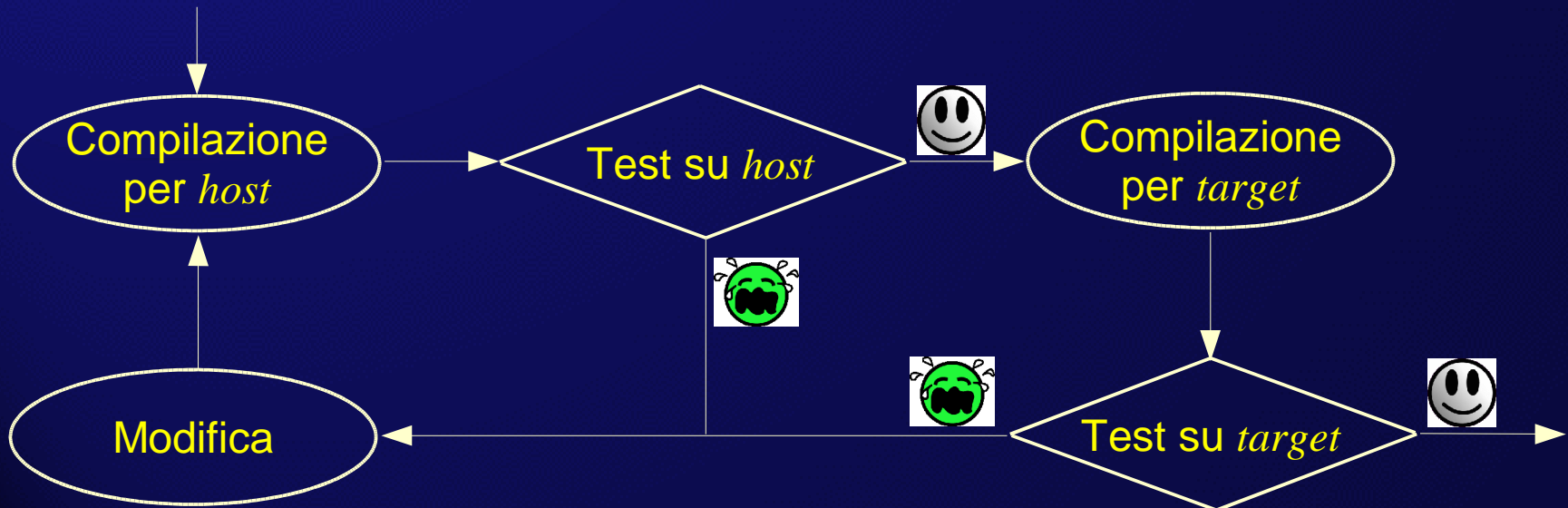
Gli applicativi (o *user space*)

Quando parliamo di sistemi *GNU/Linux* parliamo sostanzialmente di sistemi che posseggono già di per se una grande quantità di programmi applicativi.



Ogni applicativo è una parte a se stante e quindi è facilmente rimpiazzabile sia in fase di sviluppo che in una successiva fase di messa in opera.

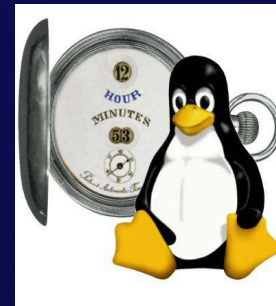
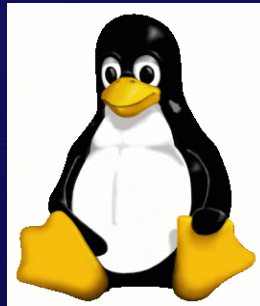
Addirittura la fase fase di test e validazione potrebbe essere effettuata sul sistema *host* e poi, semplicemente ricompilando il tutto trasferirla sul sistema *target*!



Embedded real-time

Il nucleo *Linux* è un sistema *UNIX*-like e quindi non pensato per gestire politiche di schedulazione di tipo *hard real-time*.

Quello che possiamo fare è però adottare alcuni accorgimenti per avvicinare il più possibile *Linux* ad un sistema real-time adatto per gestire sistemi di controllo e/o sistemi in cui è necessario avere una buona «prevedibilità» del sistema.



Soft real-time

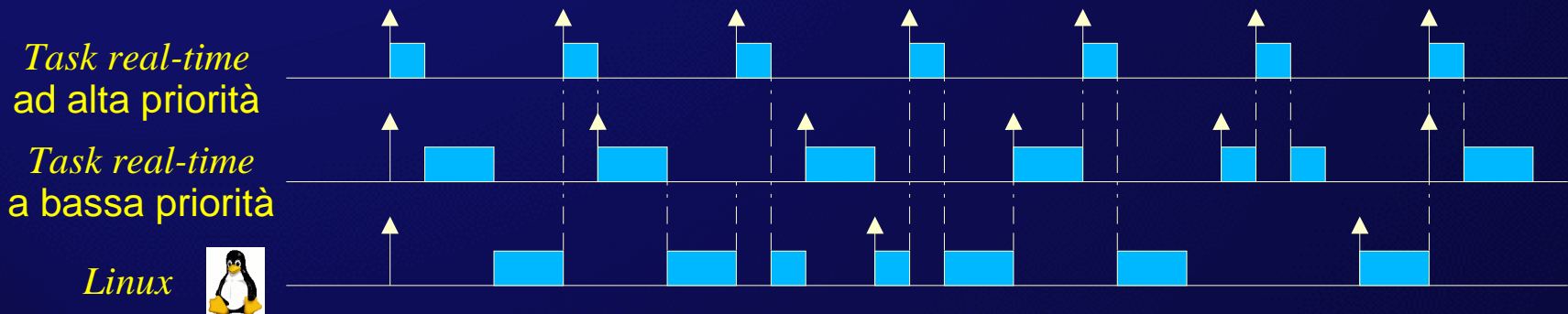
Linux di per se possiede già alcune caratteristiche che lo rendono un sistema di tipo *soft real-time* aumentandone quindi la prevedibilità.

- ◆ Le chiamate di sistema `sched_setscheduler()` e `mlock()`.
- ◆ Le patch «kernel preemption» e «low latency» disponibili nella versione 2.6.

(Quasi) *hard real-time*

Se abbiamo però bisogno di avere un alto grado di prevedibilità possiamo allora pensare di utilizzare delle soluzioni ancora più spinte come *RTAI* o *ADEOS*.

Questi sistemi, anche se con metodi diversi, riescono a far apparire *Linux*, al sistema *real-time*, come un *task* a più bassa priorità.



Il vantaggio sta nel fatto che si possono ancora utilizzare tutti i *tool* di sviluppo usati per *Linux*.

I vantaggi di usare un embedded libero!

Utilizzare software libero in generale, oppure in un sistema *embedded* ha enormi vantaggi:

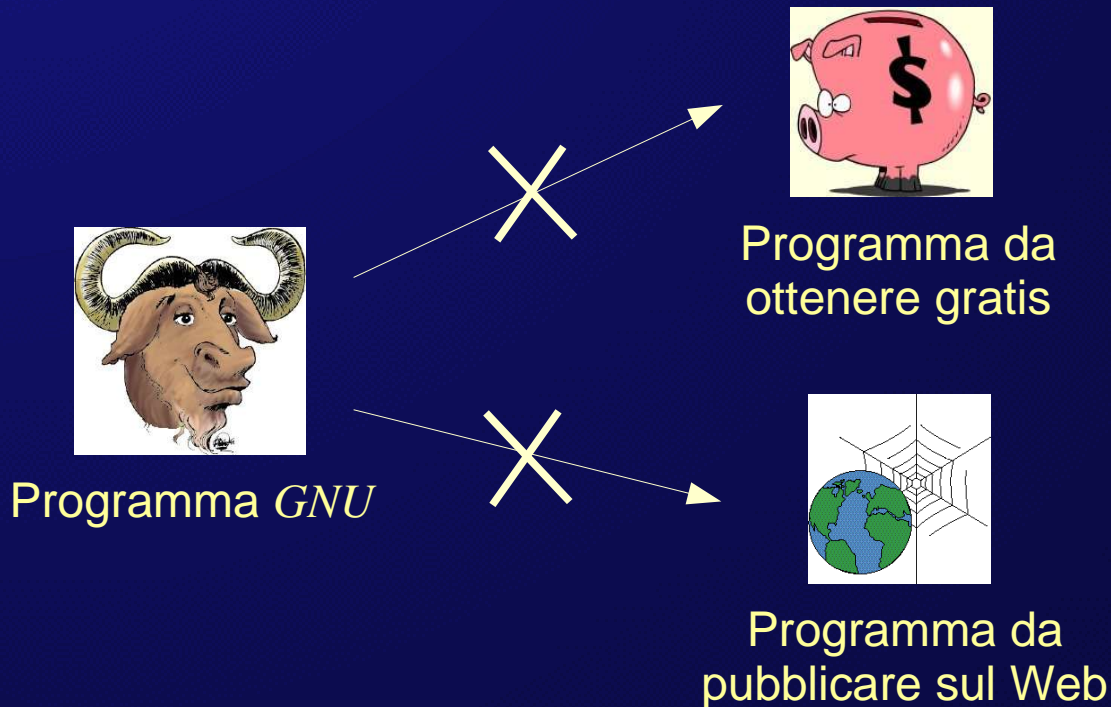
- ◆ Tutto il codice sorgente è disponibile e può essere analizzato per sapere come il sistema reagirà ad una data sollecitazione.
- ◆ Oltre ad essere disponibile, il codice è anche ***modificabile*** e ***ridistribuibile***! Questo permette quindi di adattarlo alle proprie esigenze e di correggerne gli eventuali malfunzionamenti.

La licenza *GNU/GPL* ci permette questo.

- ◆ Da parte dell'utente quindi abbiamo la ***completa indipendenza dal fornitore del servizio!***
- ◆ È possibile allora avere più consulenti per diverse parti del sistema o anche per la stessa parte. Inoltre i costi di queste consulenze possono essere ripartite sul numero di prodotti realizzati poiché non siamo in presenza di *royalties*.
- ◆ Abbiamo molte risorse di aiuto direttamente in Internet (*maillist*, siti dedicati, autore del software, ecc.) che ci possono fornire spunti per risolvere i problemi implementativi.

Alcuni miti da sfatare

Molti erroneamente pensano che un programma coperto dalla licenza *GNU/GPL* sia automaticamente gratis e da pubblicare sul *Web* coercitivamente.



Una regola da seguire

L'unico che ha dei diritti verso di voi è solo il cliente a cui viene fornito il prodotto software (magari in forma compilata), la licenza *GNU/GPL* dice infatti che solo in questo caso siete obbligati a fornire gratuitamente (o al costo di copia) i sorgenti coperti da tale licenza.



Programma *GNU*
compilato

← Legame indissolubile →



Programma *GNU*
sorgente

La *GNU/GPL* richiede questo per garantire che anche i vostri clienti godano gli stessi diritti di cui voi avete goduto.