relatore
Marco Cavallini
KOAN

Rights to copy

info@koansoftware.com

Document sources, updates and translations:
ftp://ftp.koansoftware.com/public/talks/Confsl-2009/

Corrections, suggestions, contributions and translations are welcome!

Latest update: 13 giu 2009

Sistema Embedded

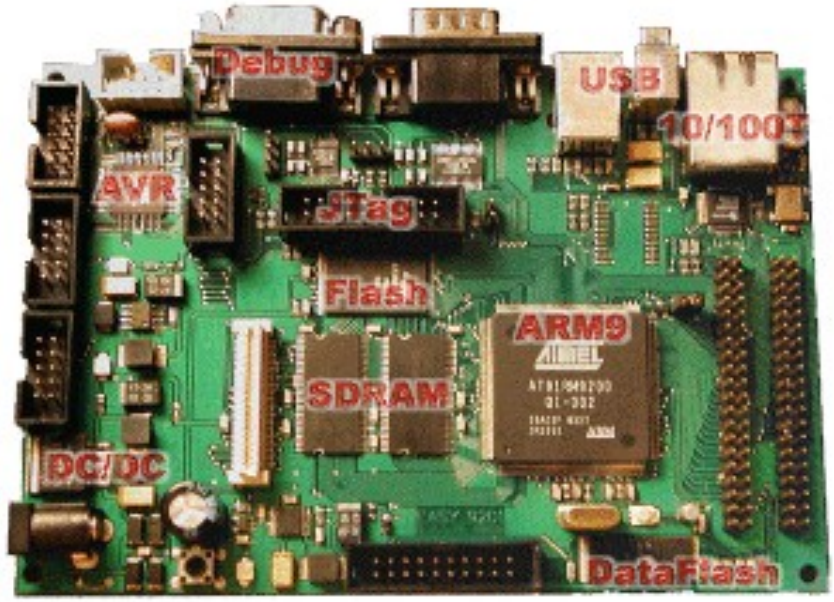
Il termine "sistema embedded" identifica genericamente dei sistemi elettronici a microprocessore progettati appositamente per una determinata applicazione, spesso con una piattaforma hardware ad hoc.

In questa area si collocano sistemi di svariate tipologie e dimensioni, in relazione al tipo di microprocessore, al sistema operativo, ed alla complessità del software che può variare da poche centinaia di bytes a parecchi megabytes di codice.

http://www.embedded.it/?q=content/definizioni

Con 'Embedded Linux' ci si riferisce a quell'insieme di distribuzioni concepite per essere utilizzate su sistemi embedded.

Le caratteristiche principali di tali sistemi impongono dei vincoli molto severi al sistema operativo in termini di memoria flash occupata, memoria centrale necessaria, tempi di avvio brevi.



Requisiti Linux Embedded

I requisiti indispensabili per un sistema linux embedded sono:

- Dimensione contenuta
- Riproducibilità
- Affidabilità

Creazione di una distribuzione

Per creare una distribuzione Linux embedded esistono diversi approcci tipici:

- Do It Yourself (DIY) Linux From Scratch (LFS)

- Downscaling (Debian, Fedora, Slack)

- Distro compatte preesistenti (DamnSmall, PeeWee, Midori)
- Tools per generazione automatica...

Tutti gli sforzi di KOAN nel mantenere la propria distribuzione KaeilOS embedded sono mirati al raggiungimento di un obiettivo ambizioso: fornire ai nuovi utenti/clienti che si approcciano per la prima volta a Linux embedded un sistema guidato per la generazione del sistema target.

Ciò offre all'utente la possibilità di potersi concentrare solo sull'applicativo o le attività strettamente legate al proprio core-business. Questo genere di soluzioni risulta quindi essere un importante strumento per aumentare la produttività ed il time-to-market.

Già dalla fine del 1999 KOAN era in grado di offrire questo tipo di prodotto (denominato Klinux) completamente Open Source.

Tools

Alcuni dei più noti tool per la generazione automatica di sistemi Linux embedded sono:

- buildroot
- PTXdist
- OpenEmbedded
- Poky (basato su OpenEmbedded)
- Scratchbox
- uClinux
- emDebian
- OpenWRT

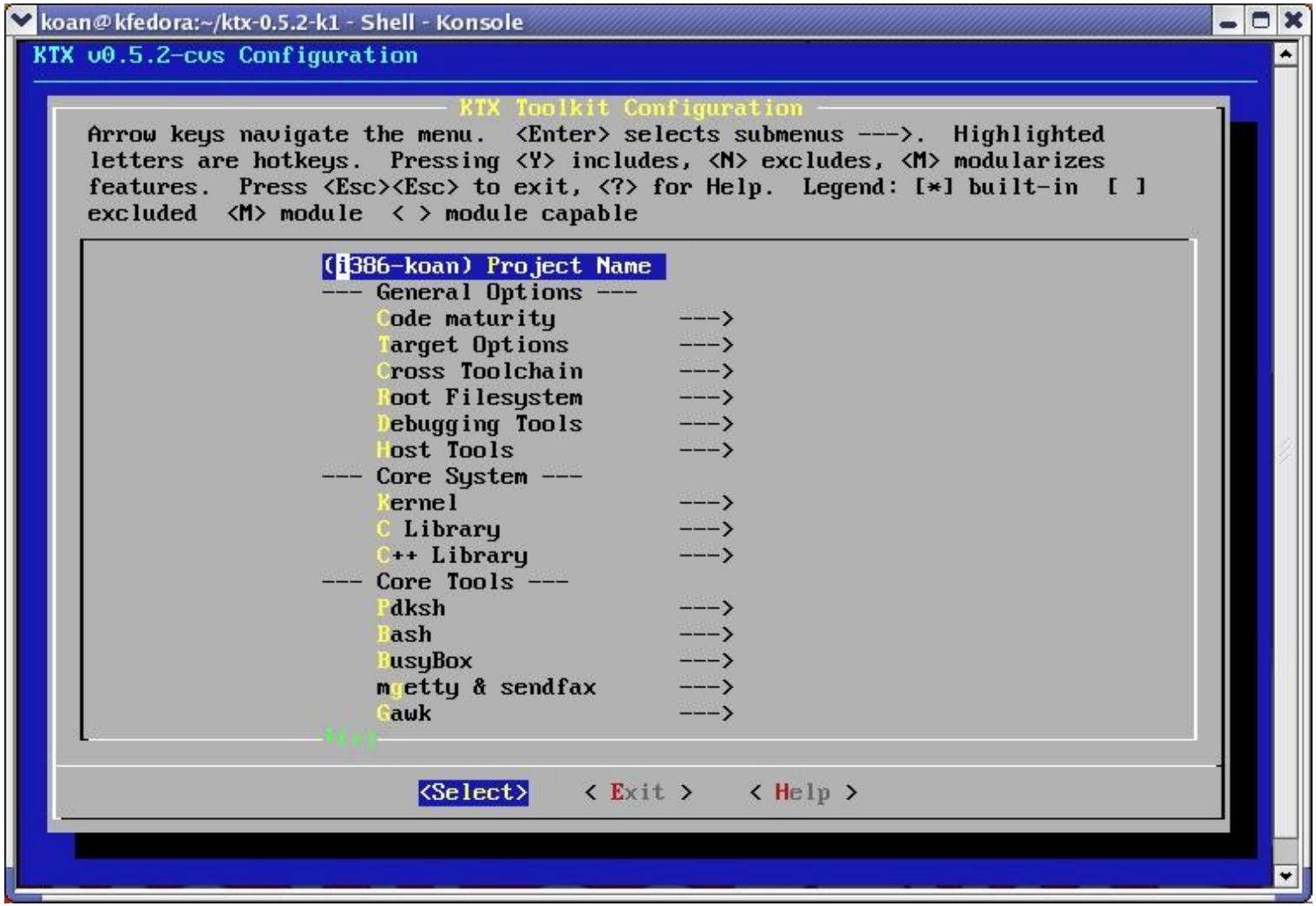
Tools: PTXdist...

PTXdist è un sistema per la generazione di sistemi Linux embedded rilasciato in licenza GNU/GPL e sviluppato dall'azienda tedesca Pengutronix.

PTXdist è basato sul sistema di configurazione Kconfig del kernel di linux ed è stato il sistema su cui KOAN ha basato Klinux alla fine del 1999 (rinominandolo KaeilOS nel 2006).

Tools: PTXdist

PTXdist definisce tramite degli switch di configurazione che devono essere selezionati con un'interfaccia a caratteri.



Tools: Poky...

Poky è un progetto derivato da OpenEmbedded che però ha delle particolarità ben precise pur rimanendone molto simile nello sviluppo, infatti, conserva la scelta di bitbake come sistema per la gestione di dipendenze e compilazione di pacchetti, ma ha un albero di directorty completamente diverso.

Poky era supportato dall'azienda inglese Opened-Hand, ora acquisita da Intel. Il progetto Poky ha risentito gravemente del cambio di proprietà dell'azienda mantainer e si trova ora in una situazione di stallo.

Ciò ha provocato la migrazione a OpenEmbedded di molti utenti di Poky, così come diversi pacchetti e ricette disponibili solo in Poky sono stati ora importati in OE.

Tools: Poky...

Le principali caratteristiche di Poky sono:

Un sistema completo di Linux kernel, X11, Matchbox, GTK+, Pimlico, Clutter, e altri applicativi GNOME, oltre ad una completa piattaforma di sviluppo.

Un sottoinsieme di OpenEmbedded focalizzato e stabile che può essere usato facilmente.

Pieno supporto ad un'ampia gamma di hardware x86 e ARM.

Tools: Poky...

Un importante componente integrato con Poky è Sato, un'interfaccia grafica GNOME ottimizzata per dispositivi mobile. Questa interfaccia è stata sviluppata per lavorare bene con schermi ad alta risoluzione e ristretta dimensione, come quelli che spesso si trovano nei PDA.

Tools: Poky

Poky è principalmente una piattaforma per sviluppare un'immagine di filesystem basata su software opensource come il server X Kdrive, il window manager Matchbox, il toolkit GTK+.

L'immagine del file system può essere generata per diversi tipi di device, ma è possibile anche usare come target una macchina virtuale QEMU che emula hardware x86 e ARM.

Tools: Scratchbox...

Scratchbox è un toolkit di cross-compilazione ideato per rendere più semplice lo sviluppo di applicazioni Linux embedded.

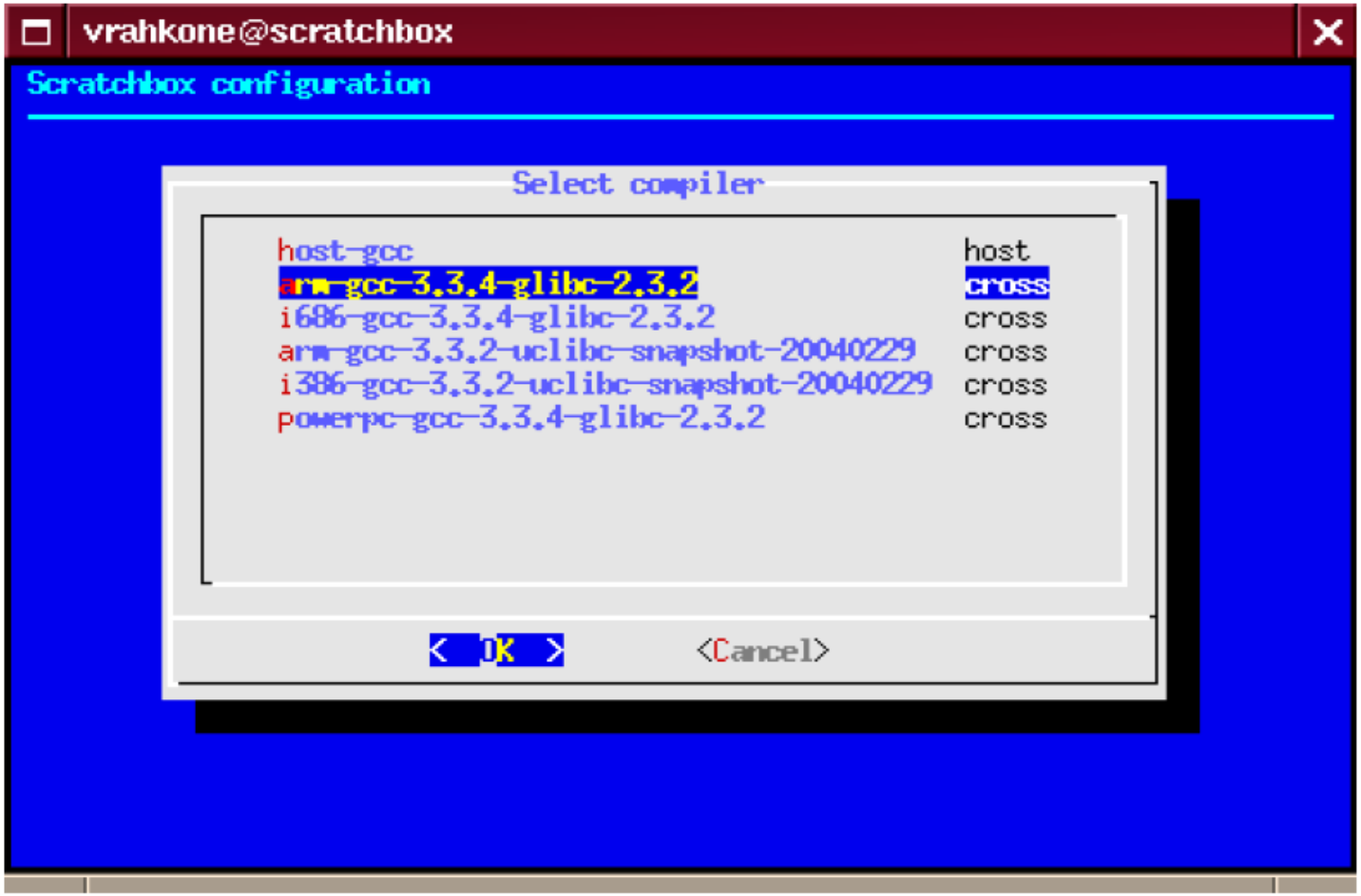
Mette a disposizione un set completo per integrare e cross-compilare una intera distribuzione Linux.

Scratchbox ha un'area (test environment) dove viene verificato il corretto funzionamento dei suoi componenti in modo da assicurare l'affidabililtà d'uso di determinate versioni di librerie, header e applicativi.

Scratchbox permette di cross-compilare pacchetti software per x86 e ARM.

Tools: Scratchbox...

L’ambiente definisce le opzioni tramite degli switch di configurazione che devono essere selezionati con un'interfaccia a caratteri.



Tools: Scratchbox

Scratchbox permette di agganciarsi ai repository ufficiali Debian (anche Slackware) e installare un sistema da zero. E' possibile utilizzare il comando apt-get e iniziare a costruire il proprio filesystem con software compilato da debian per le architetture x86 e arm.

Scratchbox è utilizzato da Nokia che ha sviluppato il progetto Maemo. Maemo ha un Xserver e un ambiente grafico basato su gtk interamente emulabile grazie alla shell scratchbox (tramite Qemu). Scratchbox inoltre mette a disposizione diverse toolchain che comprendono gcc ottimizzato per diverse CPU e uClibc

Tools: OpenEmbedded...

OpenEmbedded: storia

Il progetto OpenEmbedded è stato creato originariamente nel 2003 da un gruppo di sviluppatori del progetto OpenZaurus ed in particolare da Chris Larson (overall architecture), Holger Schurig (first implementation), e Michael Lauer (first loads of packages and classes).

Altre distribuzioni hanno iniziato ad adottare OE: Unslug, OpenSimpad, GPE Phone Edition, Ångström, OpenMoko e recentemente anche KaeilOS.

Ognuna di queste distribuzioni apporta il proprio bagaglio di esperienze e di specifiche esigenze al progetto OE, rendendolo una vera e propria fucina di pacchetti e architetture supportate.

OpenEmbedded: concetti

Il concetto sviluppato da OpenEmbedded, come per altri sistemi analoghi, è di prendere del software e creare qualcosa che si può eseguire su un altro dispositivo.

Ciò comporta il download del codice sorgente, la compilazione, la creazione di pacchetti (come .deb .rpm .ipk) e la creazione dell'immagine di boot che può scritta sul dispositivo di storage del sistema target.

Le difficoltà di cross-compilazione e la varietà di dispositivi che possono essere supportate, porta un po' complessità in più in una distribuzione basata su OpenEmbedded che quella che si può trovare in una tipica distribuzione desktop (dove la cross-compilazione non è necessaria).

OpenEmbedded: fasi di generazione

Per ogni progetto viene generalmente richiesta la stessa sequenza dei seguenti compiti:

- Scaricare il codice sorgente, e relativi file di supporto (come initscripts);
- Estrarre il codice sorgente e applicare tutte le patch che possono essere necessarie;
- Configurare il software, se necessario (come si fa eseguendo lo script 'configure');
- Compilare il tutto;
- Pacchettizzare tutti i file in uno dei formati disponibili, come .deb o .rpm o .ipk, pronti per l'installazione.

OpenEmbedded: aspetti critici

Nel caso di sistemi embedded, ciò che rende difficile queste procedure sono i seguenti aspetti:

- Cross-compilazione: cross-compilare è difficile, e un molto software non ha alcun supporto per la cross-compilazione - tutti i pacchetti inclusi nel OE sono cross-compilati;
- Target e l'Host sono diversi: questo significa che non è possibile compilare un programma e poi eseguirlo – esso è compilato per funzionare con il sistema target, non sul sistema di Host di compilazione.
- Toolchains (compilatore, linker, ecc...) sono spesso difficili da compilare. Le cross toolchains sono ancora più difficili. In genere si tende a scaricare una toolchain fatta da qualcun altro.

OpenEmbedded: caratteristiche

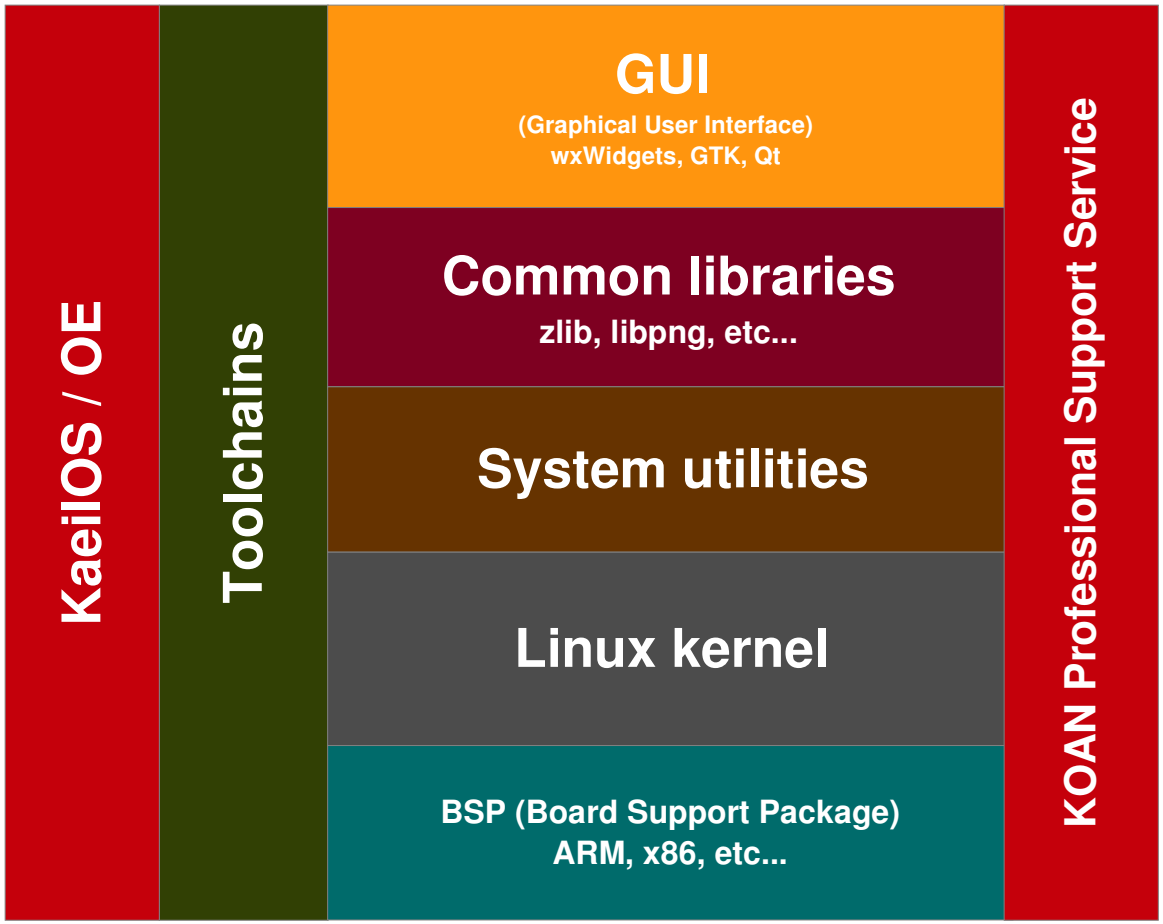
Naturalmente in OE c'è molto di più che la semplice compilazione dei pacchetti, alcune delle caratteristiche che supporta comprendono:

- Supporto per entrambe glibc e uclibc;
- Generazione per diversi dispositivi target da un'unica base di codice;
- Automatizzare tutto ciò che è necessario per compilare e/o eseguire il pacchetto (compilare ed eseguire le sue dipendenze);
- Creazione di immagini disco flash (jffs2, ext2, gz, squashfs, ecc...)
- Supporto per vari formati di pacchettizzazione;
- Costruzione automatica di tutti gli strumenti di cross-compilazione necessari;


based on
openembedded
Embedded linux
for industrial applications
KaeilOS®

OpenEmbedded: KaeilOS

Il progetto KaeilOS offre componenti, tools e servizi virtualmente comuni a tutti i progetti Linux embedded, racchiudendoli sotto un unico prodotto.



Perchè usare KaeilOS

KaeilOS/OE attualmente utilizza Ångström come distribuzione di riferimento ma ha come obiettivo caratteristiche differenti:

- boot più rapido (< 15÷20 sec.)
- supporto principale del filesystem in ramdisk
- eliminazione della gestione dinamica dei dispositivi con udev, a favore di mdev
- eliminazione del supporto DHCP e servizi non espressamente richiesti in fase di configurazione
- possibilità di estendere il sistema con patch Real-Time (Xenomai o RTAI)
- visione più “industriale” nell'utilizzo di OE

Perchè usare KaeilOS...

OpenEmbedded +

Industrial Linux +

Supporto tecnico =

Come usare KaeilOS

Indicazioni per installare KaeilOS/OE:

http://www.kaeilos.com/?q=download

Le indicazioni sono semplificate al massimo e permettono all'utente non pratico di linux di ottenere facilmente un sistema di sviluppo pronto per l'uso e uniformato secondo lo standard previsto da KOAN per un più rapido supporto.

Indicazioni originali di OE:

http://wiki.openembedded.net/index.php/Getting_Started

Poiché KaeilOS è basato su OpenEmbedded (OE),
i riferimenti a KaeilOS/OE che seguiranno
possono essere applicati anche ad OE.

KaeilOS/OE dettagli

bitbake: ambiente di compilazione per dispositivi embedded.

OpenEmbedded: collezione di *ricette* per bitbake (metadata) che descrivono come compilare:

Ricette per migliaia di pacchetti (applications, libraries, kernels, bootloaders...). *Marzo 2009*:
1800 *ricette* in `recipes/<tool>/`
>6000 versioni di pacchetti in `recipes/<tool>/*.bb`

Target machines. *Marzo 2009*:
212 *machines* definite in `conf/machine/`

Distribuzioni: configurazioni di macchine+pacchetti. *Marzo 2009*:
34 distro definite in `conf/distro`

bitbake

bitbake dettagli...

Genera **tutto** from scratch, usando file di definizione (.bb) denominati in OE come *ricette* (*recipes*)

Implementato in Python.
Creato dal pacchetto emerge (Gentoo).

Scarica i sorgenti da Internet, sia da tarballs che da source control repositories (svn, cvs, git...). Può usare source mirrors.

Applica le patches, contenute nelle ricette (.bb).

Per default, identifica e compila la versione più recente dei componenti prescelti.

Crea sempre il proprio compilatore e cross-compilatore specificato, e anche i tool di configurazione (autoconf...).

bitbake dettagli

Configura, compila e installa pacchetti binari e librerie sul filesystem.

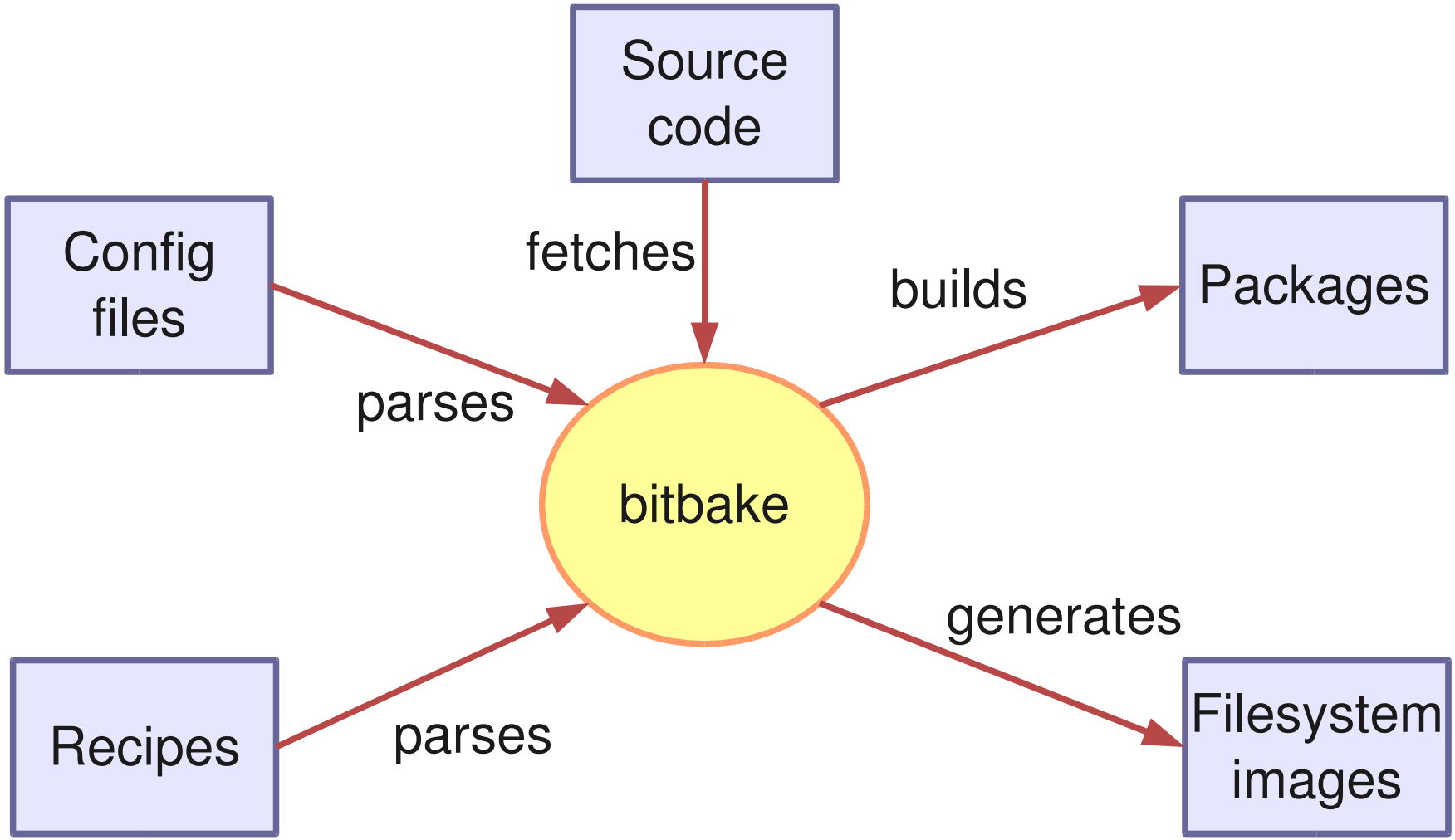
Supporta sia glibc che uClibc

Può essere usato per compilare diverse architetture.
Però necessita di duplicare le directory di build.

Crea l'immagine del filesystem: `.ext2`, `.jffs2`
Supporta diversi formati di pacchetti: `.rpm`, `.ipk`, `.deb`

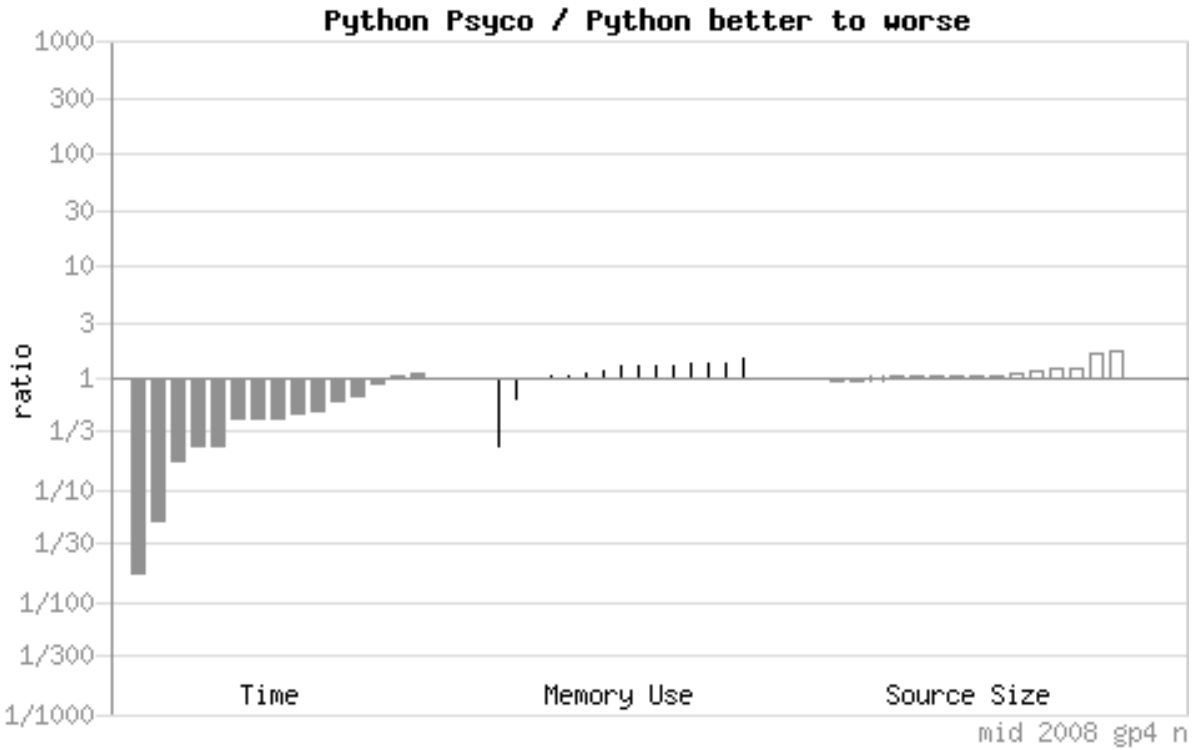
Può essere usato per (cross)compilaer anche un singolo pacchetto: `bitbake devmem2`

Fasi di bitbake



Bitbake e python-psyco

Per velocizzare la compilazione è utile installare Psyco, un compilatore python runtime che offre un incremento medio di velocità di esecuzione del codice pari a 4x



http://shootout.alioth.debian.org/gp4/benchmark.php?test=all&lang=psyco

Bitbake opzioni

bitbake ...

--version show program's version number and exit

-b BUILDFILE, --buildfile=BUILDFILE

execute the task against this .bb file, rather than a package from BBFILES

-c CMD, --cmd=CMD

Specify task to execute. Note that this only executes the specified task for the providee and the packages it depends on, i.e. 'compile' does not implicitly call stage for the dependencies (IOW: use only if you know what you are doing). Depending on the base.bbclass a listtaks tasks is defined and will show available tasks

-e, --environment show the global or per-package environment

-d, --disable-psyco disable using the psyco just-in-time compiler (not recommended)

-g, --graphviz emit the dependency trees of the specified packages in the dot syntax

bitbake -e x11-image

Bitbake comandi OE

bitbake ...

-c clean	clean the specified package
-c fetchall	fetches all the packages
-c rebuild	clean and rebuild the specified package
--c listtasks	show the global or per-package environment
-c devshell	open a shell with all environment variables properly set

bitbake -cclean x11-image

Git è un sistema software di controllo versione distribuito

Creato da Linus Torvalds nel 2005

La progettazione di Git è stata ispirata da BitKeeper e da Monotone, dove ogni sviluppatore ha una copia locale dell'intera cronologia di sviluppo, e le modifiche vengono copiate da un repository a un altro. Queste modifiche vengono importate come diramazioni aggiuntive di sviluppo, e possono essere fuse allo stesso modo di una diramazione sviluppata localmente.

E' usato da KaeilOS/OE per la gestione delle patch

git impostazioni...

La configurazione avviene a livello utente con i comandi seguenti:

```
$ git config --global user.name "Nom Cognome"
$ git config --global user.email "nome@dominio.com"
$ git config --global color.status auto
$ git config --global color.branch auto
$ git config --global color.diff auto
$ git config --global color.ui auto
$ git config --global core.editor vim
```

e si ottiene il seguente risultato nel file di configurazione:

```
$ cat ~/.gitconfig

[user]
        name = Nome Cognome
        email = nome@dominio.com
[color]
        status = auto
        branch = auto
        diff = auto
        ui = auto
[core]
        editor = vim
```

git comandi

git pull	Fetch from and merge with another repository or a local branch
git branch	List, create, or delete branches
git checkout	Checkout a branch or paths to the working tree
git add	Add file contents to the index
git commit	Record changes to the repository
git status	Show the working tree status
git log	Show commit logs
git reset	Reset current HEAD to the specified state

Conclusione e riferimenti

BitBake / OE resources

ELC 2008 presentation, by Matthew Locke:
http://www.celinux.org/elc08_presentations/mlocke-elc2008-oe.pdf

OpenEmbedded getting started guide:
http://wiki.openembedded.net/index.php/Getting_started

BitBake user manual
http://bitbake.berlios.de/manual/

OpenEmbedded Wiki:
http://wiki.openembedded.org

OE mailing lists
http://wiki.openembedded.net/index.php/Mailing_lists

Contribuire

Qualunque contributo allo sviluppo del progetto e alla traduzione, correzione o integrazione del presente documento è gradito, e dovrà essere inviato al seguente indirizzo email:

m.cavallini@koansoftware.com

Eventuali aggiornamenti del presente documento saranno pubblicati al seguente indirizzo web:

ftp://ftp.koansoftware.com/public/talks/Confsl-2009/KaeilOS-Confsl09.pdf

Bibliografia

Swicegood, Travis. (2008). Pragmatic Version Control Using Git, USA: The Pragmatic Bookshelf

Vellei, Simone. (2008). KOAN documentazione interna aziendale. Analisi frameworks per sistemi embedded

Lauer, Michael. (2005). FOSDEM presentation. Building Embedded Linux Distributions with BitBake and OpenEmbedded http://linuxtogo.org/~mickeyl/tools/FOSDEM2005.pdf

Opdenacker M. & Petazzoni T. (2009). Free Electrons presentation. OpenEmbedded http://free-electrons.com/doc/openembedded-lab.pdf

Rigo, Armin. Representation-based Just-in-time Specialization and the Psyco prototype for Python

OpenEmbedded. User manual. http://docs.openembedded.org/usermanual/usermanual.pdf

Bitbake. Manual page. http://bitbake.berlios.de

Git. Manual page. http://www.kernel.org/pub/software/scm/git/docs/git.html

Wikipedia. http://it.wikipedia.org - http://en.wikipedia.org

KOAN services

Embedded Linux Training

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux
uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization

Custom Development

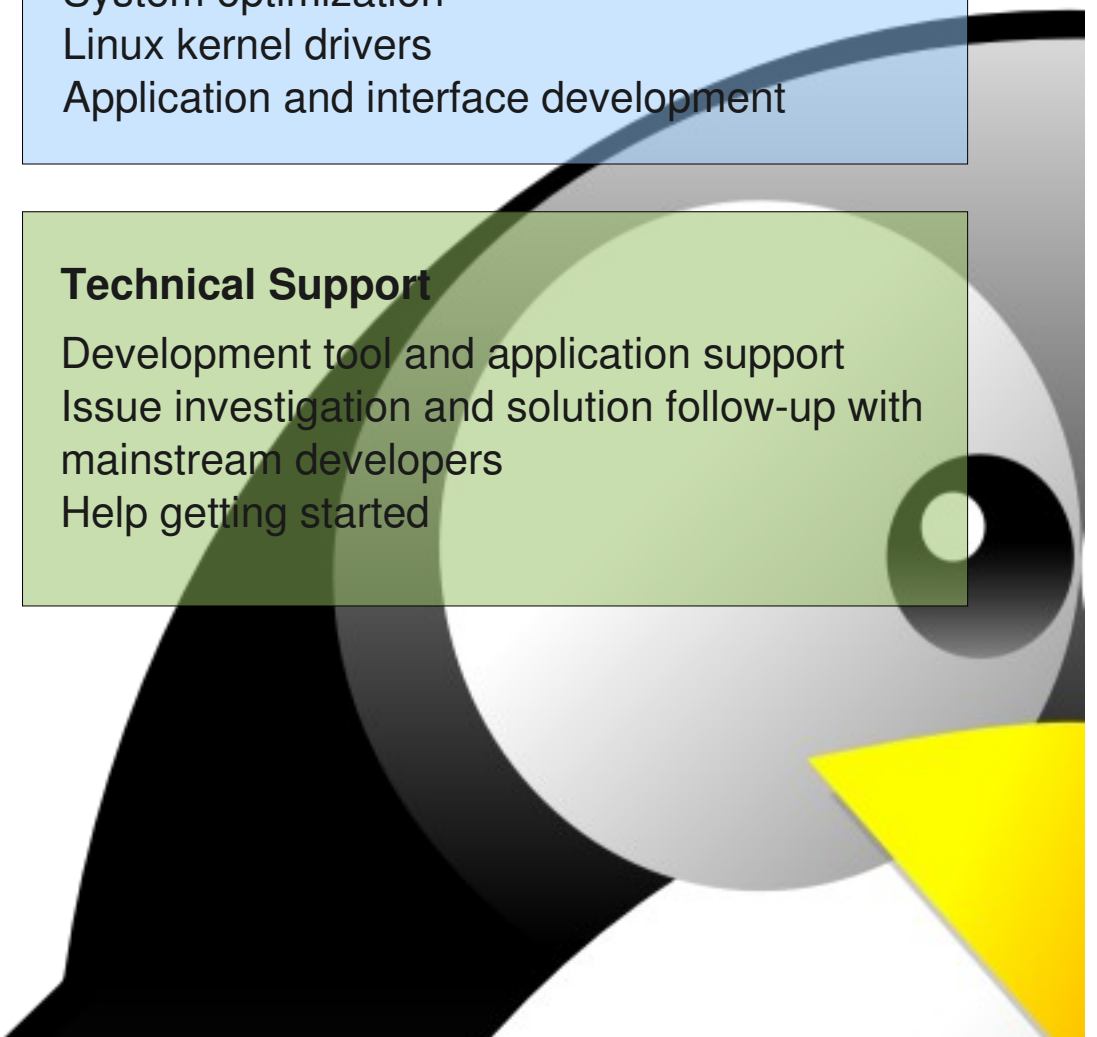
System integration
Embedded Linux demos and prototypes
System optimization
Linux kernel drivers
Application and interface development

Consulting

Help in decision making
System architecture
Identification of suitable technologies
Managing licensing requirements
System design and performance review

Technical Support

Development tool and application support
Issue investigation and solution follow-up with mainstream developers
Help getting started



http://www.koansoftware.com