OpenEmbedded on the Simputer

Nov 26, 2006
FOSS.in, Bangalore

by

Harald Welte <laforge@gnumonks.org>

About the Speaker

Who is speaking to you?

- an independent Free Software developer
- one of the authors of Linux kernel packet filter
- busy with enforcing the GPL at gpl-violations.org
- working on Free Software for smartphones (openezx.org)
- ...and Free Software for RFID (librfid)
- ...and Free Software for ePassports (libmrtd)
- ...among other things ;)
- who is not a member of the OpenEmbedded project
- ... but a proud owner of an Amida Simputer

The Problem (A)

What is the Problem? (Variant A)

- You build an embedded device
- You decide to run linux
- You do your own embedded distribution
  - which is a lot of work in the first place
  - which will cause even more work for maintainance
  - in the end, you will not provide security updates
  - and you end up having a ’one time throw away’ product
- Your users will not get your full build system (if there is such a thing)

The Problem (B)

What is the Problem? (Variant B)

- You build an embedded device
- You decide to run linux
- You license a commercial embedded Linux variant
  - you will most likely end up with something stale like kernel 2.4.x
  - you will have to spend a lot of money on it
  - you will still require quite a bit of porting
- Your users will never get the source packages ("SRPMS") to it

The Problem (Summary)

- Summary of the situation
  - You exclude the FOSS community from your product
  - You end up with low-quality code and lots of maintenance work
  - Your customers get a suboptimal product with limited feature set
- Result of that situation
  - Your customers will start their own embedded distributions
    - OpenWRT
    - OpenEZX
    - OpenZaurus
    - Familiar
    - ...

Introduction to OE

What is OpenEmbedded (OE)

- Not a distribution, but distribution building framework
- Not a software program
- Consists of thousands of rules
  - Rules for definition of a machine type (78)
  - Rules for definition of a distribution (32)
  - Rules for individual packages (4095)
- Plus a program to interpret those rules
  - bitbake
- "One system to rule them all"

Introduction to OE

- What does OpenEmbedded (OE) do for you?
  - Build a toolchain
    - specifically for your target device
    - with the optimizations you need
    - for your host platform (I crosscompile from quad G5!)
  - Build a kernel image
    - your preferred version with your patches
  - Build a distribution
    - with the packages you want
    - with the initial configuration / fs layout you want
  - Build distribution images
    - using rootfs of your choice (cramfs, jffs2, ...)
    - matching for direct flash writing
    - optionally in your own firmware update image format
  - Build thousands of individual packages
    - using the package manager of your choice (.ipk, .deb)
    - packages can be later installed
    - package repositories can be published as ’feed’ (apt-get like)

Who uses OpenEmbedded

Who uses OpenEmbedded

- Until 07/2006, only community projects
  - OpenZaurus, OpenEZX, etc.
- Since 07/2006, the first commercial user
  - FIC-sponsored OpenMoko.org (Linux GSM phone)
    - OpenMoko distribution
    - Neo1973 machie
    - QT2410 machine
    - .. more devices in 2007!
- Why not more commercial users
  - as usual: not all that much documentation about the system
  - but: OE core team members available for consultancy
  - not many commercial embedded vendors interested in sustainable, long-term development

The heart of OE: bitbake

What is bitbake

- Program to interpret
  - local configuration (.conf files)
  - package specification (.bb files)
  - machine/distro configuration (.conf files)
- Can be used to
  - build individual native (host) and target packages
  - build tasks (task == set of packages)
  - build ready-made firmware images

Devices (’machines’)

Overview of built-in device support

- Motorola A780 / E680
- HTC Blueangel
- Various Sharp Zaurus models
- VIA EPIA boards
- iPAQ H1910, H1940, H2200, H3600, H3900, H4000, H5xxx, H6300
- HP Jornada 6xx, 7xx
- i.MX31 ADS
- Nokia 770
- Linksys NSLU2, WRT54g
- Asus WL-500g
- QEMU/ARM (for testing)
- Samsung SMDK 2440
- PC-Engines WRAP
- Amida Simputer (not yet mainline)
- ...

OE Packages

An OE package is...

- a .bb (bitbake) file containing
  - description
  - license
  - section
  - maintainer
  - dependencies
  - source code + patch URI’s (local or remote)
- so it is basically similar to a RPM spec file or debian ’rules’

OE Distributions

An OE distribution is

- a .conf file that indicates
  - name (DISTRO_NAME)
  - version (DISTRO_VERSION)
  - how to build the crosscompiler
  - which package format to use (INHERIT += package_ipk)
  - which images to build by default (IMAGE_FSTYPES)
  - preferred versions of many packages

OE Tasks

Tasks are virtual packages

- You can find OE Tasks in
  - openembedded/packages/tasks
- Commonly used tasks are
  - task-bootstrap (all packages for basic userspace with login)
  - task-xterminal (boostrap + x11 + xterm)
  - gpe-image (xterminal + GPE project)
  - opie-image (OPIE project)

OE Images

An OE Image is

- a set of OE packages pre-installed into a root filesystem
- again implemented as virtual package
- OE Image rules are found in openembedded/packages/images
- result provided as .tar.gz, .tar.bz2, cramfs or jffs
- Commonly-used images:
  - bootstrap-image (basic system with console access)
  - xterminal-image (bootstrap + X11 + xterm)
  - e-image (xterminal + enlightenement e11)
  - gpe-image (xterminal + GPE)
  - opie-image (QtEmbedded, OPIE, no X11)

OE Build Setup

OE Build Setup

- create a ’build/conf/local.conf’ file
  - TMPDIR - directory with lots of space (30G)
  - MACHINE - the device you want to build for
  - DISTRO - the distro you want to build
  - BUILD_ARCH - the native architecture of the host PC (optional)
- install bitbake into

OE Build Tree

OE Build Tree layout

- my-oe/openembedded
  - the openembedded rules checked out via monotone (mtn)
- my-oe/openembedded/packages
  - package rule files
- my-oe/openembedded/conf/machine
  - machine rule files
- my-oe/openembedded/conf/distro
  - distro rule files
- my-oe/build/conf
  - local.conf configuration
- my-oe/build/tmp/work
  - work directory of build process
- my-oe/build/tmp/deploy/ipk
  - completed ipk packages
- my-oe/build/tmp/deploy/images
  - completed filesystem images

The Amida Simputer

The Amida simputer is a device with

- Intel SA-1100 StrongARM Processor
- 64 MB RAM
- 32 MB Flash
- USB Host port
- USB Device port
- Serial port (console)
- Smart Card Reader
- ...

OE for the Amida Simputer

Adding a new device to OE

- is extremely easy
  - in most cases, architecture / SoC support already there
- you just create a "conf/machine/foobar.conf" rule file
- content of the file
  - size of root flash image
  - which rootfs format to create (jffs2, ...)
  - which kernel to build
  - which compiler architecture + flags to use
- see following example for Amida 4200

OE for the Amida Simputer

Which strategy to go

- oeputer
  - use old compiler
  - use original kernel
  - try to maintain binary compatibility with existing apps
  - this was the initial attempt, now abandoned
- oeputer-ng
  - use latest toolchain (compiler, ...)
  - use latest versions of libraries, X11 server, ...
  - use current kernel
  - this is the current approach, esp. after Alchemy is becoming Free Software
    - which means we can theoretically re-compile it
    - in practise, there's probably quite a bit of porting needed
    - volunteers? (see next presentation on OpenAlchemy!)

OE for the Amida Simputer

oeputer

- create kernel package with original kernel tree from amida
  - see example
- create bootloader package with original bootloader tree from amida
  - this is optional
    - we could just leave the existing bootloader
    - if we want to do modifications, create pacakge
- use existing glibc, zlib, ... packages
  - works if the versio is compatible with what simputer uses
  - in most cases, simputer software versions are too old

OE for the Amida Simputer

oeputer-ng

- kernel
  - initially, use oeputer kernel package
  - later, port drivers/machine support to mainline and use 2.6.x
- bootloader
  - just leave as-is or use oeputer package
- userspace
  - just use most current (stable) versions of everything in OE
    - glibc-2.4
    - gcc-4.1.1
    - x11-kdrive from X11R7.1
    - ...

OE Build Timeline

OE Build Timeline (bitbake bootstrap-image)

- Build Order:
  - some native (host) libraries/tools
    - autotools
    - coreutils
    - ipkg
    - libxml
    - m4
    - fakeroot
  - the cross-toolchain
    - binutils
    - gcc
  - the basic packages (from task-bootstrap)
    - linux-libc-headers, glibc, module-init-tools
    - zlib, ncurses, util-linux, kernel
  - the boostrap-image (from bootstrap-image.bb)
    - all packages from task-bootstrap
    - create jffs / tar.bz2

Status of OE on Simputer

Status of OE on Simputer

- Proof-of-concept bootstrap-image exists
- Project is stalled because of lack of time
  - Did I mention how many projects I’m involved in?
- Volunteers wanted
  - If there are no volunteers taking it further, it will probably be still-born
  - Talk to Anush Shetty!
- OpenAlchemy
  - will make the whole project even more interesting
  - the idea is to create bitbake rules for OpenAlchemy
  - which can then be built for ’oeputer-ng’
  - but also for many (all?) other OE supported systems!

Links

Links

- The OpenEmbedded project
  - http://openembedded.org/
- Getting Started with OpenEmbedded
  - http://www.openembedded.org/wiki/GettingStarted
- The Amida Simputer
  - http://www.amidasimputer.com/
- OE on Simputer project
  - http://simputer.gnumonks.org/
- OpenMoko project
  - http://www.openmoko.org/