Rapid Board Support Package prototyping with OpenWrt

Florian Fainelli

Grenoble, ELC-E 2009

October 15, 2009

Outline I

Outline II

Introduction

- Project started in December 2003 as a replacement firmware for Linksys WRT54G
- Based on uClibc's Buildroot with some modifications to generate IPKG packages
- Strong community of users and lot of cheap and hackable devices (Fonera, Asus, D-Link, Netgear ...)

Evolutions

- 2006: Whiterussian, 134 packages,
- 2007: Kamikaze 7.0, 8 hardware targets (Atheros, Broadcom, x86 ...), 250 packages
- 2009: Kamikaze 8.0, 17 hardware targets, 300 packages
- November 2009 "Backfire": current trunk, 34 targets, 2.6.30+ kernels
- growing base of users and contributors

Evolutions

Accross the different versions, we wanted to:

- get rid of all platform specific assumptions (settings storage, filesystems, architecture ...)
- write as little lines as possible when adding new targets or packages
- focus on integrating more software and platforms
- address original buildroot issues (recompilation, integration, redundancy)

Board Support Package

A Board Support Package is composed of the following items:

- documentation
- toolchain
- kernel
- root filesystem
- debugger
- bootloader
- (IDE)

Challenges

There are a couple of challenges to address when working with an existing environment:

- mastering the complete environment (integration, recompilation, modifications ...)
- decoupling the software environment from the system
- reusability accross several hardware targets
- integrating security fixes and software updates
- use the same environment for production use and development

Components division

OpenWrt is designed to build an ease the maintenance of:

- host-side tools (lzma, mtd-utils, dtc, mkimage ...)
- toolchain (assembler, linker, compiler, C standard library)
- kernel
- root filesystem packages
- (bootloaders)

OpenWrt focus

OpenWrt provides a safe base to develop with:

- genericity across targets
- hardware abstraction and configuration
- kernel-version abstraction (packages, scripts ...)
- working toolchains combinations
- working base system (dhcp, ssh, network configuration)

Everything is a Makefile template

OpenWrt makes an extensive use of GNU make templates:

- All defaults are defined as generic templates (downloading, decompressing, compiling, installing ...)
- Components define their specific parts (stamps, compilation process, installation steps)
- Allows parallelization of jobs
- Checks steps consistency (resuming, restart when appropriate)
- Checks host-side tools requirements

Example

```
include $(TOPDIR)/rules.mk

PKG_NAME:=gpioctl
PKG_RELEASE:=1
PKG_VERSION:=1.0

include $(INCLUDE_DIR)/package.mk

define Package/gpioctl
  SECTION:=utils
  CATEGORY:=Utilities
  TITLE:=Tool for controlling gpio pins
  DEPENDS:=@GPIO_SUPPORT
endef

define Build/Prepare
        mkdir -p $(PKG_BUILD_DIR)
        $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

define Build/Compile
        $(MAKE) -C $(PKG_BUILD_DIR) \
                $(TARGET_CONFIGURE_OPTS) CFLAGS="$(TARGET_CFLAGS) -I$(LINUX_DIR)/include"
endef
```

Building steps

Each component can be cleaned, compiled or built separately:

```
make toolchain/{clean,compile,install}
make[1] toolchain/clean
make[2] -C toolchain/kernel-headers clean
make[2] -C toolchain/binutils clean
make[2] -C toolchain/gcc clean
make[2] -C toolchain/uClibc clean
[...]
make[2] -C toolchain/uClibc compile
[...]
```

Works the same way for packages:

```
make package/busybox/{clean,compile,install}
```

And the kernel:

```
make target/linux/{clean,compile,install}
```

Tools integration

OpenWrt integrates the following tools:

- quilt is used to apply kernel patches (easy maintenance between upgrades)
- any package can be downloaded using its SCM (git, svn, cvs, hg ...) or tarball at an URL
- possibility to work with different kernel, binutils, gcc, libc versions
- easy to add host-side required tools (dtc, mkimage, firmware generation utility)

Developer focus

Developers can focus on :

- developing target-specific tools and software
- easily testing new kernel versions
- doing quality assurance on the running systems
- **what** to compile and integrate, not **how** to do it

Key features

- support for initramfs, jffs2, squashfs, yaffs2 ...
- multiple C libraries: uClibc, eglibc, glibc (all selectable)
- work with several hardware targets in the same directory via environments (**scripts/env**)
- support for an external toolchain
- support for building an external kernel source tree (directory or Git repository)
- generates packages under the IPKG (almost deb-compatible) format
- kernel modules packages generated on-the-fly
- firmware upgrade mechanisms
- ready-to-use Web interface for demo/config (LuCI)

Adding a hardware target

As simple as:

- defining the toolchain architecture (mips, powerpc, armeb ..) in **target/linux/myboard/Makefile**
- put a kernel configuration **target/linux/myboard/config-default**
- put board-specific kernel patches in **target/linux/myboard/patches/**
- write board specific scripts (LEDs blinking, upgrade procedure ...)
- make [...]

Handling multiple kernel versions

Easy switching between multiple kernel versions:

- define kernel version in **target/linux/myboard/Makefile**
- use a 2.6.x kernel configuration **target/linux/myboard/config-2.6.x**
- board-specific kernel patches in **target/linux/myboard/patches-2.6.x/**

Providing third-party source packages

Feeds are useful for third-party packages to be integrated in the final rootfs:

- setup a "source" feed
- write and maintain your packages Makefiles separately with your SCM
- packages are generated by OpenWrt just like other packages (busybox, iptables ...)
- suitable for binary drivers, applications ..

Providing software ipks

- HTTP/FTP repositories of ipks
- software updates for deployed systems
- allows other environments to generate those packages and still be usable on the target system

Cavium Networks Octeon

Cavium Networks Octeon porting steps:

- MIPS64R2 toolchain generated by OpenWrt (w/o Octeon optimizations)
- download Octeon-specific patches on the linux-mips and netdev mailing-lists
- package board-specific drivers : SATA, Ethernet, USB
- testing the kernel without reflashing the device: initramfs image
- estimated time to a booting system : 2h
- migration over a GCC-4.4.1 Toolchain w/ Octeon optimizations

Thank you

Thank you very much, question session is now open