

# Rapid Board Support Package prototyping with OpenWrt

Florian Fainelli

Nantes, RMLL 2009

July 9, 2009

# Outline I

- 1 Introduction
  - Evolutions
  - Users
  - Board Support Package
  - Challenges
- 2 Design
  - Components division
  - Components building steps
  - Tools integration
- 3 Working with the environment
  - Key features
  - Adding a hardware target
  - Providing third-party source packages
  - Providing third-party ipks
  - Example and demo

# 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
- next one: trunk, 34 targets ...

# Goals

Across the different versions, we wanted to:

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

# Users

- Wireless communities: Athens, Berlin, France Wireless ...
- Fon
- SMC, Accton
- Sony
- Ubiquity Networks
- Infineon
- SFR (Efixo)

# 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 (recompilation, modifications ...)
- reliability and stability issues
- 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 (dtc, mkimage ...)
- toolchain (assembler, linker, compiler, C standard library)
- kernel
- root filesystem packages
- (bootloaders)

# 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 (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
include $(INCLUDE_DIR)/kernel.mk

PKG_NAME:=gpiotl
PKG_RELEASE:=1
PKG_VERSION:=1.0

include $(INCLUDE_DIR)/package.mk

define Package/gpiotl
    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)

# Key features

- support for initramfs, jffs2, squashfs, yaffs2 ...
- multiple libcs: uClibc, eglibc, glibc (all selectable)
- work with several hardware targets in the same directory via environments (**scripts/env**)
- support for an external (binary) 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 (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**
- (add kernel patches)  
**target/linux/myboard/patches/001-myboard.patch**
- write board specific scripts (LEDs blinking, upgrade procedure ...)

# 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

# Example and demo

Broadcom BCM6338 (ADSL System-on-a-Chip) porting:

- MIPS32 toolchain generated by OpenWrt and known to be working
- kernel work to support BCM6338 System-on-a-Chip: patches in `target/linux/brcm63xx/patches-2.6.27`
- testing the kernel without reflashing the device: `initramfs` image
- need to integrate binary drivers and driver control applications
- CFE downloads the TFTP image in SDRAM
- it works !

# Thank you

Thank you, question session is now open