OpenWrt : fresh air for (wlan) routers

Florian Fainelli
florian@openwrt.org

Rencontres Mondiales du Logiciel Libre 2006
Vandoeuvre-lès-Nancy
Lenght : 30 minutes

Thursday July 6th 2006

Summary I

Summary II

Summary III

What if the manufacturer does not provide sources
Working basis
Evaluation of the porting effort
Adding a new architecture to buildroot-ng
Conventions
target/Config.in
target/linux/architecture-2.x/Makefile
target/image/architecture/Makefile
include/target.mk

Summary IV

What is OpenWrt

- contraction of Opensource Wireless Technology
- minimalist GNU/Linux distribution GPL licensed
- set of Makefile providing the building of a full filesystem
- package and updates repository

Project history

- OpenWrt was created by Gerry Rozeman (aka Groz) and Mike Baker (aka [mbm]) in november 2003.
- Since the beginning, Gerry and Mike felt the great potential offered by a Linux-based firmware, and at the same time the limitations provided by the Linksys one. That is why they decide to replace the later by a minimalist one, built with the current uClibc buildroot.
- The philosophy is simple : everything is configured in commmand-line using SSH

Context

- At the time the first OpenWrt version is released, Sveasoft firmwares were already available since few months and add various features, configurable through the Linksys web interface.
- Few months later, DD-WRT firmware comes out, an OpenWrt fork, the main reason for its developpement is the lack of an OpenWrt web interface.

State of art

Nowadays, OpenWrt team is composed of 5 main developpers, helped by many contributors :

- Mike Baker ([mbm])
- Imre Kaloz (Kaloz)
- Nicolas Thill (Nico)
- Felix Fietkau (nbd)
- Florian Fainelli (florian)

Development tools

- subversion repository
- Trac web interface : https ://dev.openwrt.org

Subversiion repository organisation

The subversion repository is divided into several directories :

- 2 branches : **whiterussian/** and **buildroot-ng/**
- 5 tags : whiterussian_rc1 to 5
- 1 packages directory : **packages/**
- kamikaze in trunk (currently being migrated to **buildroot-ng/** and **packages/**

Whiterussian

Whiterussian is currently the stable version of the OpenWrt firmware. It runs fine on devices based on Broadcom 947xx and 953xx boards, such as :

- Linksys WRT54G v1.0 to v4
- Asus WL-500g (Deluxe, Premium)
- Motorola WR850G, WE500G
- Buffalo WBR-B11, WBR-G54, WLA-G54

It is being used a firmware basis by several Wireless User Groups, and some companies, such as FON(fonbasic firmware).

Whiterussian limitations

Altough the firmware runs fine, it is currently being limited by :

- the usage of a binary Broadcom Wi-Fi driver, thus restricting to a 2.4 kernel
- the difficulty to maintain and port packages
- the hardware support limited to Broadcom 47xx/53xx boards
- a web interface too much relying on the existence of a NVRAM

Kamikaze I

As a consequence to these difficulties, and the more and more increasing market of Linux-based hardware, the **Kamikaze** branch was opened.
New hardware platforms were then supported :

- Texas Instruments AR7 (noyau 2.4)
- Atheros AR531x (noyau 2.4)
- Aruba (noyau 2.6)
- x86 (noyaux 2.4 et 2.6)

Kamikaze II

- Broadcom SiByte (noyau 2.6)
- AMD Alchemy (noyau 2.6)
- Intel Xscale IX42x (noyau 2.6)
- Router Board RB532 (noyau 2.6)

Kamikaze limitations

Kamikaze has a certain number of drawbacks :

- difficulty in stabilising the kernels, most of the hardware platforms are not fully functionnal (Wi-Fi is not working most of the time)
- adding and maintaining packages is too close to the whiterussian way
- maintaining 2 distinct repository using different toolchains

buildroot-ng

- abstraction et simplication d'écriture des fichiers **Makefile** et compatibilité avec la syntaxe précédente
- les paquetages dépendant fortement du noyau vont dans **buildroot-ng**, les autres dans **packages/**
- dépôt multi-architectures indépendamment du du système de base

Main tasks

- Finish **buildroot-ng**
- Porting AR7-2.4 to AR7-2.6
- Porting Broadcom 63xx 2.6
- Rewriting **webif**
- Rewrite of the user documentation

Interests of OpenWrt

- fully customizable system from kernel to filesystem
- strictly identical firware independently from the platform runned on
- vendor version independent
- fully GPL code

Legal concerns

Some legal concerns are raised when you know a hardware is running Linux : Plusieurs questions légales se posent lorsque vous avez connaissance qu'un matériel donné fonctionne sous Linux :

- does the manufacturer provide the firmware source code ?
- does this hardware use binary drivers ?
- are we sure it is Linux or uClinux ?
- is the GPL code compliant with GPL or compatible ?

Proving that a hardare is running Linux

You have different ways of proving that a hardware is running Linux :

- downloading a firmware and trying to split it in : bootloader, kernel, filesystem (beware of the Big/Little Endian traps !)
- pluging a serial console and/or JTAG
- using a bug in the web interface to get the result of a dmes, cat /proc/xxxx

GPL violation

In conformance to the GPL, using GPL codes for commercial products implies the following things :

- publishing kernel sources
- publishing source code of the GPL applications used in the filesystem
- publishing sources of the GNU toolchain and the filesystem creation tools

In cas of a GPL violation, please inform :

`http ://gpl-violations.org`

Working basis

Your working basis is composed of the following elements :

- Linux kernel sources, modified to support the hardware, with the latest patches for your architecture (arm, mipsm ppc ...)
- binary drivers and firmwares for the Wi-Fi card, Ethernet, ADSL ...
- binary tools to create the firmware : CRC calculation, version, padding ...

You are very likely not to be able to get a functionnal firmware with the manufacturer tools.

Evaluation of the porting effort

According to what we have, to get a working port for the architecture with OpenWrt and being GPL compliant, we have to :

- analyse and generate differences between a vanilla kernel and the given one
- create a program adding the corect header in the firmware file (CRC calculation, version, padding ...)
- keep compatibily with the binary drivers and the current kernel version (beware of the VERSIONING option)
- eventually reverse engineer the binary drivers

Adding a new architecture to buildroot-ng

Now that we have the requirements for having an OpenWrt system for our arhictecture, let's add it :

- add and entry in **target/Config.in**
- add a directory **target/linux/architecture-2.x** (2.4 or 2.6 kernel) containing the arch-specific patches and kernel configuration
- add a directory **target/image/architecture** describing how to build the firmware image
- calling the kernel template in **include/target.mk**

Conventions

- Architecture naming must respect the kernel naming in **arch/**
- We recommend you get a vanilla kernel booting, rather than changing the filesystem
- Please separate patches as much as possible : architecture, drivers, various patches ...

target/Config.in

```
config LINUX_2_6_ARCHITECTURE
        bool "Architecture foo [2.6]"
        select mips
        select LINUX_2_6
        select PCI_SUPPORT
        select PCMCIA_SUPPORT
        help
          A short description
```

target/linux/architecture-2.x/Makefile I

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

LINUX_VERSION:=2.6.16.7
LINUX_RELEASE:=1
LINUX_KERNEL_MD5SUM:=9682b2bd6e02f3087982d7c3f5ba824e

include ./config
include $(INCLUDE_DIR)/kernel.mk
```

target/linux/architecture-2.x/Makefile II

```
include $(INCLUDE_DIR)/kernel-build.mk

$(LINUX_DIR)/.patched: $(LINUX_DIR)/.unpacked
        [ -d ../generic-$(KERNEL)/patches ] &&
$(PATCH) $(LINUX_DIR) ../generic-$(KERNEL)/patches $(MAKE_T
        [ -d ./patches ] &&
$(PATCH) $(LINUX_DIR) ./patches $(MAKE_TRACE)
        @$(CP) config $(LINUX_DIR)/.config
        touch $@
```

target/image/architecture/Makefile I

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

define Build/Compile
        rm -f $(KDIR)/loader.gz
        $(MAKE) -C lzma-loader \
                BUILD_DIR="$(KDIR)" \
                TARGET="$(KDIR)" \
```

target/image/architecture/Makefile II

```
                    install
endef

define Build/Clean
        $(MAKE) -C lzma-loader clean
endef
define Image/Prepare
        cat $(KDIR)/vmlinux |
$(STAGING_DIR)/bin/lzma e -si -so -eos -lc1 -lp2 -pb2 > $(K
```

target/image/architecture/Makefile III

```
endef

define Image/Build/hardware
        dd if=$(KDIR)/loader.elf
of=$(BIN_DIR)/openwrt-hardware-$(KERNEL)-$(2).bin
bs=131072 conv=sync
        cat $(BIN_DIR)/openwrt-$(BOARD)-$(KERNEL)-$(1).trx
>> $(BIN_DIR)/openwrt-hardware-$(KERNEL)-$(2).bin
endef
```

target/image/architecture/Makefile IV

```
define trxalign/jffs2-128k
-a 0x20000
endef
define trxalign/jffs2-64k
-a 0x10000
endef
define trxalign/squashfs
-a 1024
```

target/image/architecture/Makefile V

```
endef

$(eval $(call BuildImage))
```

include/target.mk

```
...
$(eval $(call kernel_template,2.6,architecture,
2_6_ARCHITECTURE))
...
```

Debuging and stabilizing

Common debuging tools :

- GDB
- EJTAG (si disponible)
- ksymoops
- usage of printk
- debug options enabled in the kernel
- bootloader documentation (RedBoot, CFE, YAMON, RomE ...)
- asking for help of users and developpers

Further problems

Once you get a kernel booting on your hardware, it is very likely not to be directly usage, you may encounter the following issues :

- drivers working not correctly or not at all
- unrecognized flash mapping
- low reaction system (processor caching)

Customizing the system

You can highly customize your system, such as :

- adding a captive portal, RADIUS server
- doing advanced filtering using iptables
- adding network stacks and protocols ...
- adding drivers for various hadware : webcam, additionnal Wi-Fi stick ...
- adding features to **webif**

Adding packages

We invite you to participate to the migrating effort of the packages in **kamikaze** and make them use the **buildroot-ng** syntax.
In opposition to the previous system, where you had to create 3 files :

- Makefile
- Config.in
- ipkg/paquetage.control

buildroot-ng describes and abdstracts everything in a **Makefile**.

Hierarchy

Packages are structured this way :

```
packages/
  section/
    package-name/
      Makefile
      patches/
```

packages/section/Makefile I

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

PKG_NAME:=foo
PKG_VERSION:=alpha-beta-4
PKG_RELEASE:=1
PKG_MD5SUM:=5988e7aeb0ae4dac8d83561265984cc9

PKG_SOURCE_URL:=ftp://ftp.openwrt.org/foo
```

packages/section/Makefile II

```
PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION).tar.gz
PKG_CAT:=zcat

PKG_BUILD_DIR:=$(BUILD_DIR)/$(PKG_NAME)-$(PKG_VERSION)
PKG_INSTALL_DIR:=$(PKG_BUILD_DIR)/ipkg-install

include $(INCLUDE_DIR)/package.mk

define Package/foo
```

packages/section/Makefile III

```
SECTION:=libs
CATEGORY:=Libraries
TITLE:=My sample package
DESCRIPTION:=My other descriptiong
URL:=ftp://ftp.openwrt.org/foo
endef

define Build/Configure
$(call Build/Configure/Default,--option-foo=bar)
```

packages/section/Makefile IV

```
endef

define Build/Compile
        rm -rf $(PKG_INSTALL_DIR)
        mkdir -p $(PKG_INSTALL_DIR)
        $(MAKE) -C $(PKG_BUILD_DIR) \
                DESTDIR="$(PKG_INSTALL_DIR)" \
                all install
endef
```

packages/section/Makefile V

```
define Package/foo/install
        install -m0755 -d $(1)/usr/lib
        $(CP) $(PKG_INSTALL_DIR)/usr/lib/libfoo.so.* $(1)/u
endef

define Build/InstallDev
        mkdir -p $(STAGING_DIR)/usr/include
        $(CP) $(PKG_INSTALL_DIR)/usr/include/foo-header.h $
```

packages/section/Makefile VI

```
        mkdir -p $(STAGING_DIR)/usr/lib
        $(CP) $(PKG_INSTALL_DIR)/usr/lib/libfoo.{a,so*} $(S
        touch $(STAGING_DIR)/usr/lib/libfoo.so
endef

define Build/UninstallDev
        rm -rf \
          $(STAGING_DIR)/usr/include/foo-header.h \
          $(STAGING_DIR)/usr/lib/libfoo.{a,so*}
```

packages/section/Makefile VII

```
endef

$(eval $(call BuildPackage,foo))
```

Getting support

Do not hesisate to contact us via the following ways :

- IRC : irc.freenode.net #openwrt and #openwrt-devel
- Mailing-list : openwrt-devel@openwrt.org
- Forum : http ://forum.openwrt.org

Becoming a developper

- Do not hesitate to submit patches adding packages to the repository
- Do as much test and bugreport as you can
- Port OpenWrt to a new device ...

Thank you very much

Thank you very much for your attention, question session is now open.