The OpenWrt embedded development framework

Florian Fainelli
florian@openwrt.org

Fosdem 2008
Brussels
Lenght : 1 hour

February 24, 2008

Summary I

Summary II

Summary III

What is OpenWrt

- Minimalistic Busybox/Linux distribution GPL licensed
- Set of Makefiles and tools building an embedded rootfs
- Packages and repositories
- Hardware donators, package maintainers and kernel hackers community

Challenges

- Lots of different hardware platform can run Linux
- Lots of binary drivers
- Strong memory footprint constraints
- Hardware configuration and maintenance abstraction

What is OpenWrt

OpenWrt is a set of Makefiles and sources that :

- Builds the appropriate toolchain for your device
- Compiles the appropriate kernel w/ patches and options
- Provides software as IPKG packages
- Builds the optionnal tools to flash your device

Getting OpenWrt

- Subversion reposirotyr at `https://svn.openwrt.org/openwrt` and Trac interface
- **trunk/** directory for development branch
- **kamikaze** and **whiterussian** tags for stable versions
- **packages/** directory for non-kernel related packages

uClibc buildroot heritage

OpenWrt was created in late 2003, and used a modified uClibc buildroot :

- Needed to be hacked to have support for a package maintainer
- Lots of Makefile writting to add support for a software
- Could use more Makefile templating

Key directories

There are four key directories in the base:

- `tools`
- `toolchain`
- `package`
- `target`

Packages and external repositories

OpenWrt uses IPKG as the package format and manager.

- OpenWrt provides essential and kernel-related packages in **trunk/**
- Other packages are split into a different repository **packages/** which subsections
- Support for external repository can be done either in **/etc/ipkg.conf** or in the **feeds**

Packages feeds

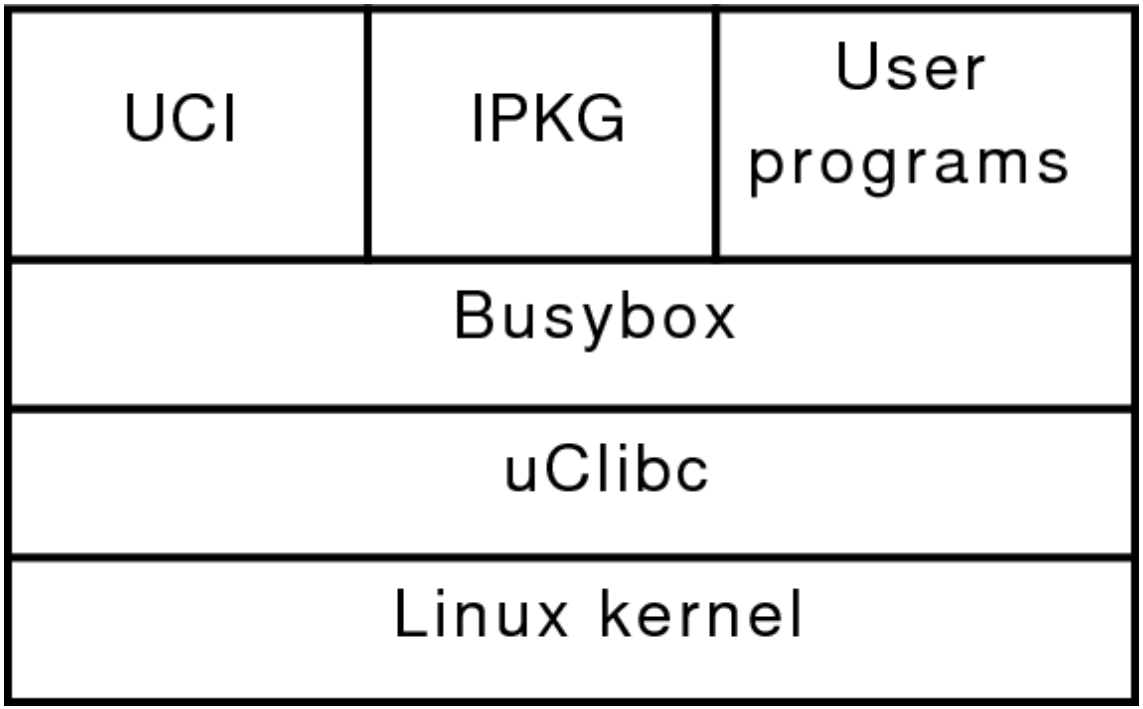
Package feeds allows you to provide your own set of packages :

- Source feeds can be fetched using svn
- They appear as packages while running make menuconfig
- User can choose to build them directly into the rootfs or as separate packages

Toolchain

- Switch between binutils, gcc, kernel-headers versions and uClibc tuples
- Change compiler CFLAGS to tune/optimize size and features
- Add custom patches to any of the above components

Software architecture



System and package configuration

OpenWrt uses UCI :

- Universal Configuration Interface
- MIB-like structure (**config.section.key=value**)
- Born from the lack of NVRAM on all supported hardware

UCI specificites

- C library, easy to link with
- program can interface with it to store their configuration
- Config to MIB bridges to easily manage devices using SNMP
- More configuration storage backends : LDAP, SQL
- Web interface with integrated UCI support

Add UCI configuration for your package

For instance adding a new configuration file is as simple as creating a new file in **/etc/config/package** which should contain the following lines:

```
config      <type> ["<name>"]      # Section
    option  <name> "<value>"       # Option
```

Creating packages

Typical package directory layout :

- `package/<name>/Makefile`
- `package/<name>/patches`
- `package/<name>/files`

Package source download

Support for different download methods :

- GIT
- Subversion
- CVS
- HTTP
- local source

Example

For instance, if you wish to checkout a particular revision of a package using Subversion, just define the following download method in your package Makefile

```
PKG_VER:=963
PKG_BRANCH:=batman-adv-userspace
PKG_VERSION:=r\$(PKG\_REV)
PKG_SOURCE_PROTO:=svn
PKG_SOURCE_URL:=http://downloads.open-mesh.net/svn/batman/
trunk/
```

Creating kernel modules packages

Extending the kernel/rootfs with modules is really easy within OpenWrt :

- Build-time update the kernel configuration based on your kernel modules selection
- External kernel modules are seen as packages, they use the KernelPackage instead

Adding support for a new target

Adding support for a new kernel is made easy :

- Create `target/linux/<my target>` directory
- Define the kernel version you want to use
- Add your additionnal patches in `target/linux/<my target>//patches`

Copying architecture files

You can also put your files in `target/linux/<my target>//files` :

- They will be copied at kernel build time
- They should match the kernel directory structure `arch/mips/kernel/*`
- You can directly version C files that are part of your drivers, architecture code ..
- Only need to generate patches against Kconfig and Makefiles

Using quilt

OpenWrt natively supports using quilt :

- Refresh, create and update patches for any component
- Test modifications with `make component/subcomponent/compile`
- Patches modifications are tracked with Subversion in your local copy

Building an external kernel tree

OpenWrt supports the building of an external kernel tree :

- Test a git kernel tree with your board code
- Use the OpenWrt toolchain to cross-compile for other architectures
- Test local modifications to a git subtree for later inclusion in mainline

Supported root filesystems

OpenWrt has currently support for :

- JFFS2, SquashFS, Ext2/3, CPIO, TGZ
- With LZMA, GZIP, BZIP2 compression (when applicable)
- Binary firmware generation tools (Broadcom, ZyXEL, Mikrotik ...)

The Image builder

- Deployment tools
- Contains compiled toolchain and kernel for your architecture
- Add custom files / IPKG packages to include in the rootfs

The SDK

- Contains the compiled toolchain
- Useful for packages upgrading and live testing
- Package maintainers tool

Becoming a developper

- Submit patches to the mainling-list **openwrt-devel@lists.openwrt.org**
- Do as much test and bugreport as you can
- Port OpenWrt to a new device / architecture
- Write documentation

Promoting OpenWrt

- Ideal firmware for Wireless communities, Wireless ISPs ..
- Fastest embedded platform to port a new architecture on
- Abstracted network and Wireless configuration for all supported hardware

Thank you very much

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