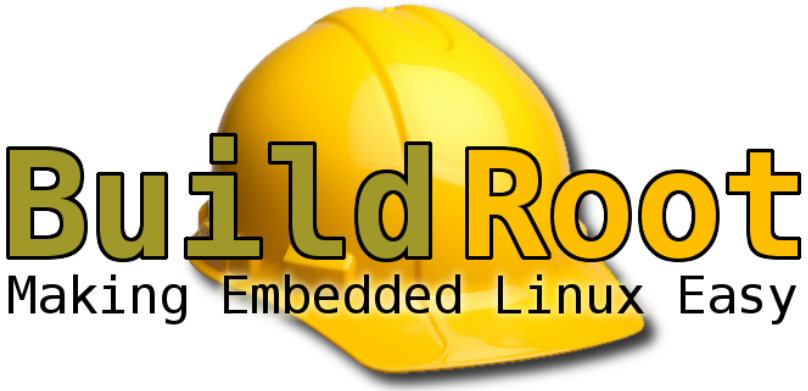
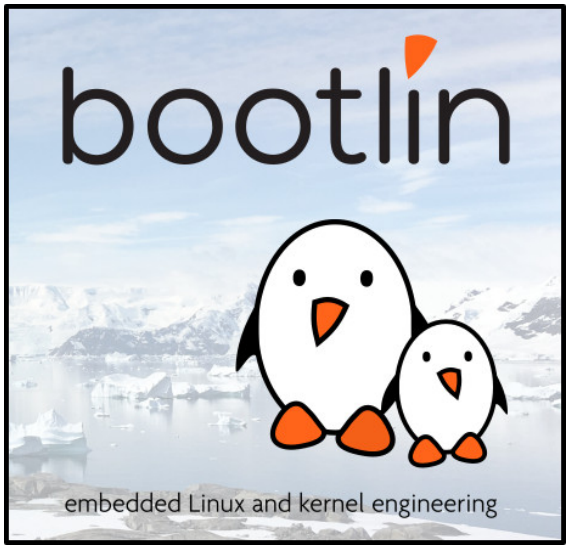
Building embedded Linux systems with Buildroot



Thomas Petazzoni
Bootlin
https://bootlin.com/



Rights to copy

Document sources, updates and translations:
https://bootlin.com/doc/training/buildroot/

Corrections, suggestions, contributions and translations are welcome!

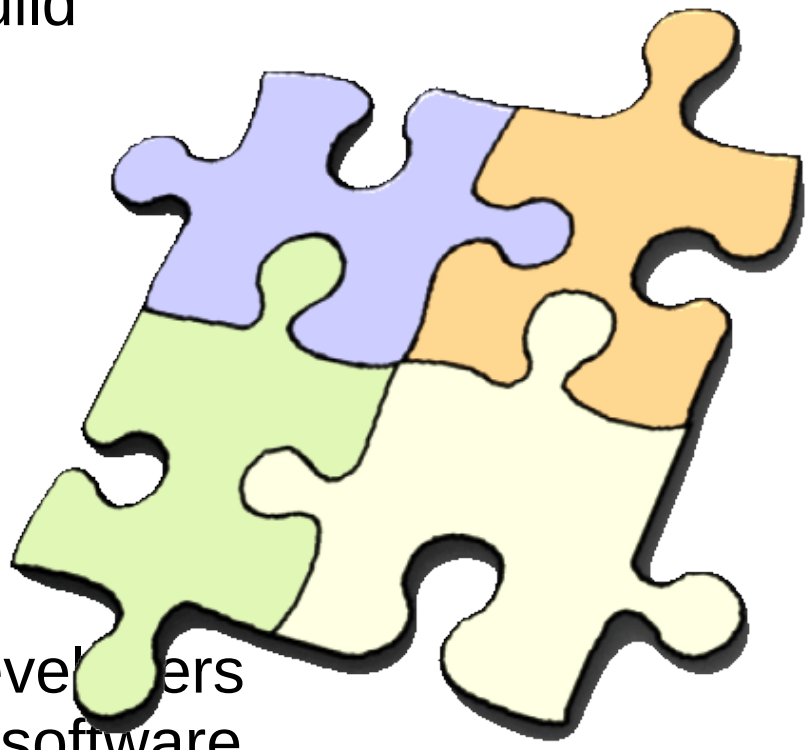
Latest update: Jul 25, 2018

Leveraging free software

Thousands of free software packages are available and can be leveraged to build embedded systems

With free software

- You have control over the source
- Bugs can be fixed
- New features can be added
- Your system can be customized



In theory, the system designers and developers have a lot of flexibility thanks to free software

However, leveraging the existing free software packages may not be very easy.

Using a distribution

Distributions provide ready-to-use binary packages

Some distributions, such as Debian, are available for embedded architectures (ARM, MIPS, PowerPC)

They make it relatively easy to get a working system, but

- The maintainer of each package has made configuration choices that don't necessarily match your choices (soft float vs. hard float, EABI vs OABI, feature A or not feature A)
- You don't have a lot of control on system integration (initialization scripts, dependencies of the components)
- If you want to integrate fixes or new features, you need to rebuild the packages. While this may not be very difficult, there's usually no automated infrastructure to rebuild the whole system.

Manually

You can configure, compile, install and integrate all the free software components needed to build a working Linux system manually.

It gives you full flexibility, but

- You need to know the dependencies between all the components
- Cross-compiling is usually a tedious process, sometimes requiring package source code changes
- You don't have an automated procedure to rebuild your system, which might be needed if you want to integrate a bug fix or a new feature

Build systems

- Build systems allow an embedded Linux developer to generate a working embedded Linux system from scratch.
- They automate the process of downloading, configuring, compiling and installing all the free software packages
  - You have a well-known procedure for rebuilding your system from scratch
  - You can easily integrate patches, bug fixes or new upstream versions
  - Your colleagues can easily take over your work, as there's a documented procedure for system generation
- The build system already knows about most free software packages
  - Dependencies are managed, and cross-compiling issues are already solved.

Entering Buildroot

Buildroot <http://www.buildroot.net> is a set of Makefiles that automates the process of building a cross-compiling toolchain and a root filesystem for an embedded system.

Buildroot has been initially developed by uClibc developers.

For a long time, it had no maintainer and no releases were delivered

Since January 2009, Buildroot now has an official maintainer, Peter Korsgaard

In February 2009, the first stable release has been published, Buildroot 2009.02. A new release will be published every 3 months.

An opportunity to take a fresh look at Buildroot !

Buildroot users

They are already using Buildroot

ATMEL for their AVR32 development kit
http://www.atmel.no/buildroot/buildroot-doc.html

Gumstix
http://docwiki.gumstix.org/index.php/Buildroot

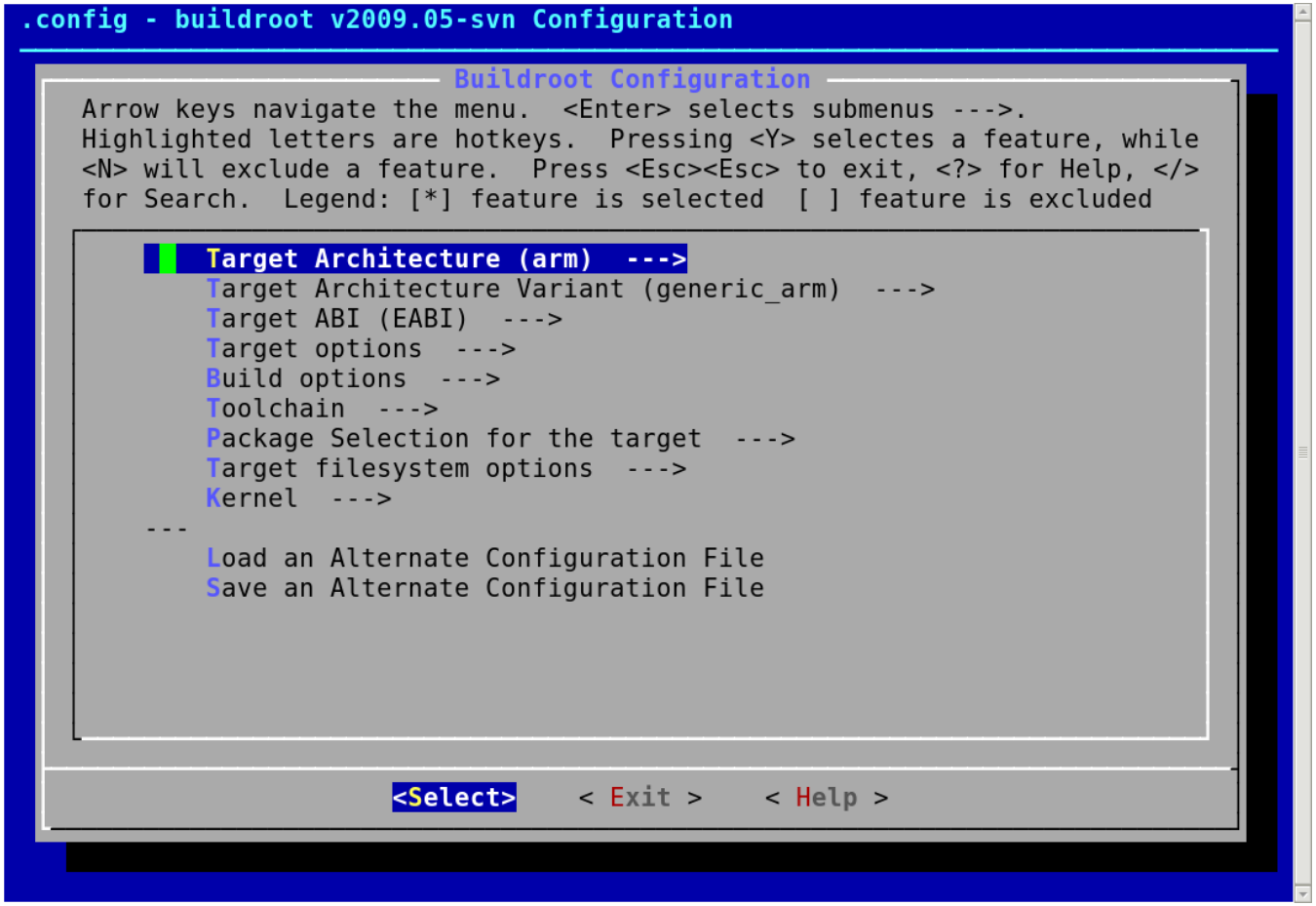
Armadeus (ARM boards with FPGA)
http://www.armadeus.com

Calao Systems
http://www.calao-systems.com/

And probably a lot more !

Buildroot configuration

- Buildroot uses the Kconfig configuration system
- Offers a menuconfig-like interface
- Saves the configuration in a .config file



Main configuration

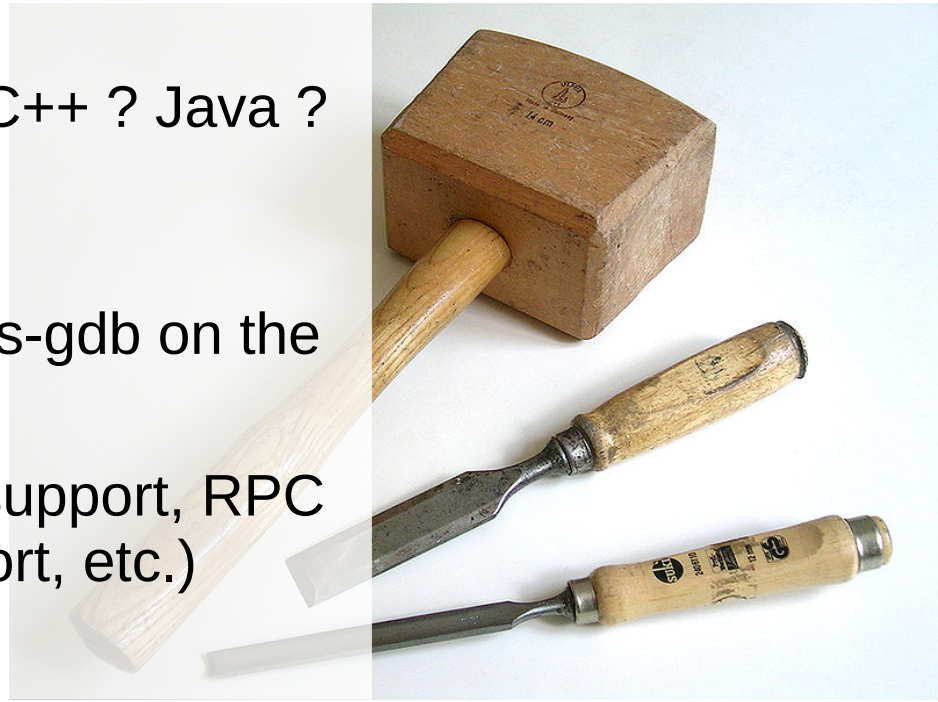
- Target architecture
  - arm, armeb, avr32, cris, i386, mips, mipsel, powerpc, superh, superh64, x86_64
- Architecture variant
  - For ARM, for example: Generic, ARM7TDMI, ARM610, ARM710, ARM720T, ARM920T, Xscale, etc.
- ABI selection
  - For ARM, EABI or OABI
- Target options
- Build options
  - Host tools to use, directory definition
  - Install documentation? Debugging symbols? Strip? Level of gcc optimization?

Toolchain configuration

Buildroot can either build a toolchain (limited to uClibc toolchains) or re-use an existing external toolchain (limited to sysroot-able toolchains)

In the case of a toolchain built by Buildroot, the configuration allows to select things such as

- Kernel headers version
- uClibc version and configuration
- GCC version and options (Fortran ? C++ ? Java ? Objective C?)
- GDB version and options (gdbserver, gdb on the target, cross-gdb on the host)
- General toolchain options (large file support, RPC support, IPv6 support, locale support, etc.)



Package selection (1/3)

Several hundreds of packages can be selected, with of course a focus on packages useful for embedded devices

Dependencies between packages are handled, either through `depends on` relation or `select` relation.

Main packages: busybox, bash, bzip2, diffutils, flex, native toolchain, grep, bootutils, cups, at, beecrypt, dash, file, gamin, less, lsof, ltrace, memstat, module-init-tools, procps, psmisc, screen, strace, sudo, syslogd, klogd, util-linux, which, etc.

Core libraries: libconfig, libconfuse, libdaemon, libelf, libevent, libgcrypt, libiconv, libidn, liblockfile, liboil, libsysfs, etc.

Package selection (2/3)

Databases: MySQL client, sqlite

Text editors: ed, nano, uemacs, vim

Networking: argus, avahi, axel, boa, bind, bridge-utils, DHCP support, dnsmasq, dropbear, ethtool, haserl, ifplugd, iperf, iproute2, ipsec-tools, iptables, kismet, l2tp, lighttpd, linkns, lrzsz, mDNSresponder, mii-diag, mrouted, nbd, ncftp, netcat, netkitbase, netkittelnet, netplug, netsnmp, nfs-utils, ntp, openntpd, openssh, openssl, openvpn, portmap, pppd, pppoe, pptp-linux, proftpd, quagga, isisd, samba, rsync, stunnel, tcpdump, tftpd, thttpd, vsftpd, wireless tools, etc.

Hardware/system tools: dm, dmraid, e2fsprogs, fis, libfuse, hal, hdparm, hotplug, i2c-tools, input-tools, iostat, libaio, libraw1394, libusb, lm-sensors, lvm2, mdadm, mtd utils, pciutils, setserial, udev, usbutils, etc.

Package selection (3/3)

Audio/video: aumix, flac, gstreamer with plugins, libmad, libmpd, libogg, libtheora, libvorbis, madplay, mpg123, mplayer, speex, vlc, festival

Graphic libraries: ncurses, slang, dialog, DirectFB, imagemagick, jpeg, libpng, libungif, pixman, SDL, QT Embedded, Gtk (atk, cairo, pango, glib), fontconfig, Freetype, Matchbox, X.org Kdrive and a few X applications (window managers, etc.)

Compressor/decompressors

Package managers: ipkg, portage, rpm

Interpreters, languages: lua, microperl, python, ruby, tcl, php

Misc: XML libraries, Java, Games

Filesystem and kernel configuration

- Output format selection for the root filesystem
  - cramfs, cloop, ext2, jffs2, ubifs, squashfs, tar, cpio, initramfs, romfs.
  - Of course, the raw root filesystem is also available
- Bootloader configuration
  - U-Boot supported
- Kernel configuration
  - The version, configuration, additional patches and the kernel binary format (uImage, zImage, bzImage) can be specified

Demo time !

Building a system with Busybox, DirectFB and example applications

Directory hierarchy

In the sources

- `docs/`, documentation
- `package/`, the configuration items and Makefiles for building the userspace packages
- `project/`, the configuration items and Makefiles for the project concept (several projects in the same Buildroot tree)
- `scripts/` various utilities
- `target/linux/`, building the Linux kernel
- `target/<fstype>/`, generating the root filesystem image
- `target/device/`, ready-made configuration for supported boards
- `target/generic/target_skeleton/`, the default root filesystem skeleton
- `toolchain/`, building a cross-compiling toolchain

Using Buildroot in a nutshell

Download the latest stable version, or get a development version from the SVN (better to report issues)

```
make menuconfig
```

```
make
```

Get your raw root filesystem in `project_build_ARCH/PROJECT/root/`

Get your root filesystem image and kernel image in `binaries/PROJECT/`

Location of output directories can be changed using `O=/path/to`

Generated directories

`build_ARCH`

- One directory for each package, where it has been configured and built
- `staging_dir`, where the toolchain and packages are installed, and where non-stripped versions of the binaries/libraries can be found

`project_build_ARCH/PROJECT`

- `autotools_stamps`, stamps for handling dependencies between build steps
- `buildroot-config`, header files related to Buildroot configuration mechanism
- `root`, the root filesystem for the target

`toolchain_build_ARCH`

- Where the toolchain components are configured and built

Buildroot: adding new packages

Create a new directory, for example `package/gqview/`

Add a `Config.in` file in this directory to describe the configuration options

```
config BR2_PACKAGE_GQVIEW
        bool "gqview"
        select BR2_PACKAGE_PKGCONFIG
        help
          GQview is an image viewer for Unix
          operating systems

          http://prdownloads.sourceforge.net/gqview
```

Insert the new `Config.in` file in the configuration system by adding to `package/Config.in`

```
source "package/gqview/Config.in"
```

Buildroot: adding new packages (2)

Create the `gqview.mk` file to describe the build steps. The following example uses the `Makefile.autotools.in` machinery, which makes it easy to support autotools based packages

```
GQVIEW_VERSION = 2.1.5
GQVIEW_SOURCE = gqview-$(GQVIEW_VERSION).tar.gz
GQVIEW_SITE = http://prdownloads.sourceforge.net/gqview
GQVIEW_AUTORECONF = NO
GQVIEW_INSTALL_STAGING = NO
GQVIEW_INSTALL_TARGET = YES
GQVIEW_DEPENDENCIES = uclibc pkgconfig libgtk2
$(eval $(call AUTOTARGETS,package,gqview))
```

Patches can also be added in the package directory, with filenames like `pkgname-version-feature.patch`. They will be automatically applied before configuring the package.

Buildroot: adding new packages (3)

Packages not based on autotools, or needing specific configuration, compilation and installation steps can be handled manually.

```
GZIP_VERSION:=1.3.12
GZIP_SOURCE:=gzip-$(GZIP_VERSION).tar.gz
GZIP_SITE:=$(BR2_GNU_MIRROR)/gzip
GZIP_DIR:=$(BUILD_DIR)/gzip-$(GZIP_VERSION)
GZIP_CAT:=$(ZCAT)
GZIP_BINARY:=$(GZIP_DIR)/gzip
GZIP_TARGET_BINARY:=$(TARGET_DIR)/bin/zmore

$(DL_DIR)/$(GZIP_SOURCE):
	 $(call DOWNLOAD,$(GZIP_SITE),$(GZIP_SOURCE))

gzip-source: $(DL_DIR)/$(GZIP_SOURCE)

$(GZIP_DIR)/.unpacked: $(DL_DIR)/$(GZIP_SOURCE)
	$(GZIP_CAT) $(DL_DIR)/$(GZIP_SOURCE) | \
	  tar -C $(BUILD_DIR) $(TAR_OPTIONS) -
	touch $(GZIP_DIR)/.unpacked
```

Buildroot: adding new packages (4)

```
$(GZIP_DIR)/.configured: $(GZIP_DIR)/.unpacked
        (cd $(GZIP_DIR); rm -rf config.cache; \
                $(TARGET_CONFIGURE_OPTS) \
                $(TARGET_CONFIGURE_ARGS) \
                ./configure \
                --target=$(GNU_TARGET_NAME) \
                --host=$(GNU_TARGET_NAME) \
                --build=$(GNU_HOST_NAME) \
                --prefix=/usr \
                --exec-prefix=/ \
                $(DISABLE_NLS) \
                $(DISABLE_LARGEFILE) \
        )
        touch $(GZIP_DIR)/.configured

$(GZIP_BINARY): $(GZIP_DIR)/.configured
        $(MAKE) CC=$(TARGET_CC) -C $(GZIP_DIR)
```

Buildroot : adding new packages (5)

```
$(GZIP_TARGET_BINARY): $(GZIP_BINARY)
        $(MAKE) DESTDIR=$(TARGET_DIR) CC=$(TARGET_CC) \
              -C $(GZIP_DIR) install-strip
ifneq ($(BR2_HAVE_INFOPAGES),y)
        rm -rf $(TARGET_DIR)/usr/share/info
endif
ifneq ($(BR2_HAVE_MANPAGES),y)
        rm -rf $(TARGET_DIR)/usr/share/man
endif

gzip: uclibc $(GZIP_TARGET_BINARY)
gzip-clean:
        $(MAKE) DESTDIR=$(TARGET_DIR) CC=$(TARGET_CC) \
          -C $(GZIP_DIR) uninstall
        -$(MAKE) -C $(GZIP_DIR) clean
gzip-dirclean:
        rm -rf $(GZIP_DIR)

ifeq ($(BR2_PACKAGE_GZIP),y)
TARGETS+=gzip
endif
```

Future work

- Cleanup
  - The project has been unmaintained for some time, so some cleanup is needed
- More packages
  - Of course !
- Better support for external toolchains
  - Including glibc toolchains
- $YOURIDEA

Feedback and community

- Buildroot is organized like a typical free software community, it is not trusted by any company, even though some of the developers are obviously professionals
  - Mailing-list, at http://lists.busybox.net/mailman/listinfo/buildroot
    - Very helpful support !
  - Bug tracker at https://bugs.uclibc.org/
  - Subversion repository at svn://uclibc.org/trunk/buildroot
  - Documentation at http://www.buildroot.net/buildroot.html
- Feel free to test, report and contribute !

Alternatives

Very similar to Buildroot

PTXdist, developed by Pengutronix
http://www.pengutronix.de/software/ptxdist/index_en.html

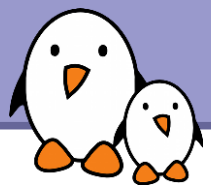
LTIB, developed mainly by Freescale.
Good support for Freescale boards
http://www.bitshrine.org/

Slightly different approaches

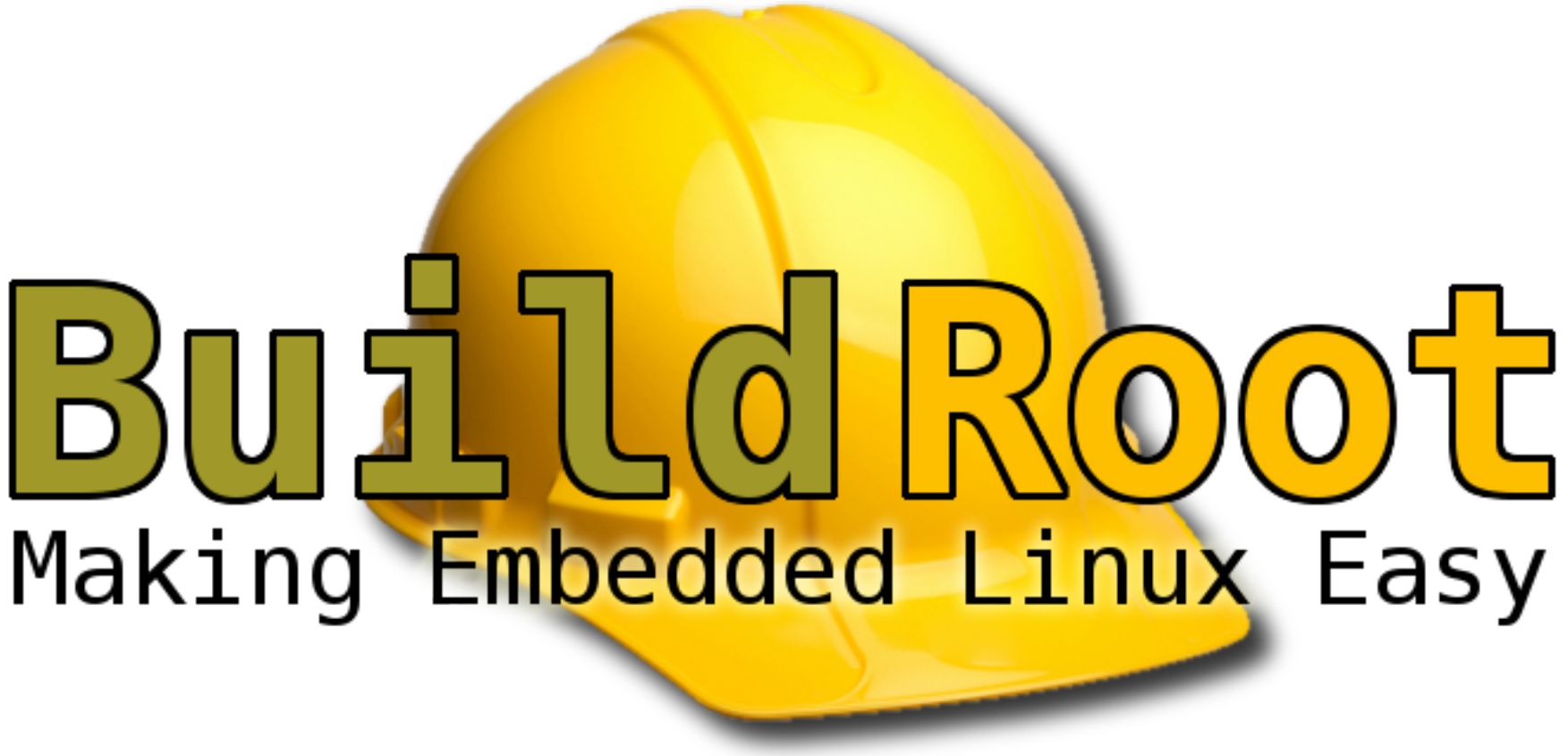
OpenEmbedded, more flexible but also far more complicated
http://www.openembedded.org

Gentoo Embedded
http://www.gentoo.org/proj/en/base/embedded/handbook/

Let's discuss these, other alternatives, their advantages, drawbacks, and future work in a BOF session on Wednesday at 1 PM



Questions?



http://www.buildroot.net