Buildroot

BASC2020 seminar

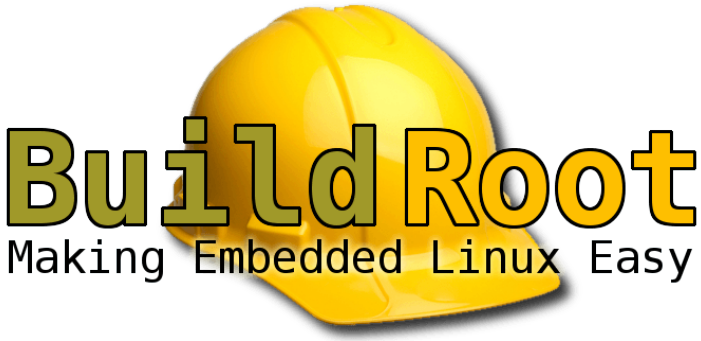
Giacomo Longo

University of Genoa

TODO

Table of contents

BuildRoot



Official website: https://buildroot.org

- Born in 2005
- Entirely based on makefiles and kconfig
- Only one goal: *producing root file system images* for *100% custom Linux systems*

BuildRoot users

The most prominent users of BuildRoot are using it for building:

- IoT devices
- Automated factory controllers
- Point of sale devices
- Car multimedia units

Why BuildRoot

- Each buildroot is a 100% custom Linux "mini-distro"
- Buildroot images can be less than 100MB or even 10MB
- Complete customization of target architecture and build flags
- Multiple compiler / libc / system layout choices
- Updated every 3 months current version is 2020.08.1
- Easily extendable

Why BuildRoot: architecture support

$\approx$ 20 architectures supported

- ARC LE & BE
- **ARM** LE & BE
- AArch64 LE & BE
- csky
- **i386**
- Microblaze AXI & Non-AXI
- MIPS LE & BE
- MIPS64 LE & BE
- nds32
- Nios II
- PowerPC
- PowerPC64 LE & BE
- RISCV
- SuperH
- SPARC
- **x86_64**
- Xtensa

The BuildRoot process

What the user sees

1. Create a configuration file
2. Start the build
3. Flash the image on the device

What BuildRoot does

1. Build a cross compiler on our machine
2. Resolve the configuration dependencies
3. Compile from source the requested packages
4. Assemble an image

Prerequisites

Packages for an ARM BuildRoot

Ubuntu 20.04

```
sudo apt-get update
sudo apt-get install -y \
  curl tar \
  make \
  gcc g++ \
  libncurses-dev libssl-dev \
  qemu-user-static \
  qemu-system-arm
```

Others

Binaries needed

Downloaders: curl & wget

Extractor: tar

Compilers: gcc & g++

Libraries: ncurses & openssl

Execution: QEMU system for ARM & QEMU static

Obtaining BuildRoot

Download from:
https://buildroot.org/downloads/buildroot-2020.08.1.tar.gz
Extract with `tar -xzf`
Your BuildRoot files will be in `buildroot-2020.08.1`

Creating an ARM cross compiler

TODO

Creating an ARM root filesystem

TODO

Creating a bootable ARM root filesystem

TODO

Customizing our images

Build time overlay

- Create a directory
- Add `BR2_ROOTFS_OVERLAY=my-overlay` to `.config`
- Rebuild using `make`
- The structure of `my-overlay` will be copied to the rootfs

How to specify multiple overlays

Multiple overlays can be specified by separating them with spaces in the `BR2_ROOTFS_OVERLAY` directive

Customizing our images

Build time script

Add `BR2_ROOTFS_POST_BUILD_SCRIPT=my-script.sh` to `.config`
Available environment variables inside:

`BR2_CONFIG`	path of `.config`
`HOST_DIR`	path of `output/host`
`STAGING_DIR`	path of `output/staging`
`TARGET_DIR`	path of `output/target`
`BUILD_DIR`	path of `output/build`
`BINARIES_DIR`	path of `output/images`
`BASE_DIR`	path of `output`

How to specify multiple scripts

Multiple scripts can be specified by separating them with spaces in the `BR2_ROOTFS_POST_BUILD_SCRIPT` directive

Customizing our images

Editing the target directory

1. Add your files to the `output/target` directory
2. Rebuild using `make`

Warning

Your files might be rewritten / deleted by buildroot

Customizing our images

D.I.Y. approach

1. Unpack your rootfs (with `tar -xzf` for instance)
2. Perform your modifications
3. Repack your rootfs (with `tar -cf` for instance)

Running dynamic executables in Docker

TODO

Running dynamic executables with systemd-nspawn

TODO

Tips and tricks

TODO

Using ltrace

TODO

Using strace

TODO

Using gdb

TODO