# Getting started with Buildroot : How to build your own Embedded Linux OS

Abdul Qadeer
Research Assistant
Centre for AI & Big Data
Namal University
Mianwali

# Contents

- **Introduction to Linux**
- Traditional Linux vs Embedded Linux
- Introduction to Buildroot
- Target Processor Architectures
- How Much Powerful Linux It Can Provide?

Buildroot Design Principles

How to create your own embedded Linux OS using Buildroot?

# Introduction to Linux

**What is Linux?**

- Linux is An open-source operating system kernel originally developed by Linus Torvalds in 1991.
- Powers a wide range of devices, from servers and desktops to embedded systems and smartphones.
- Various "distributions" like Ubuntu, Fedora, and Debian, each tailored for different use cases.

**Why Linux?**

- Open Source
- Linux can be tailored to a wide range of applications, from lightweight embedded systems to powerful supercomputers

# Embedded Linux vs Traditional Linux

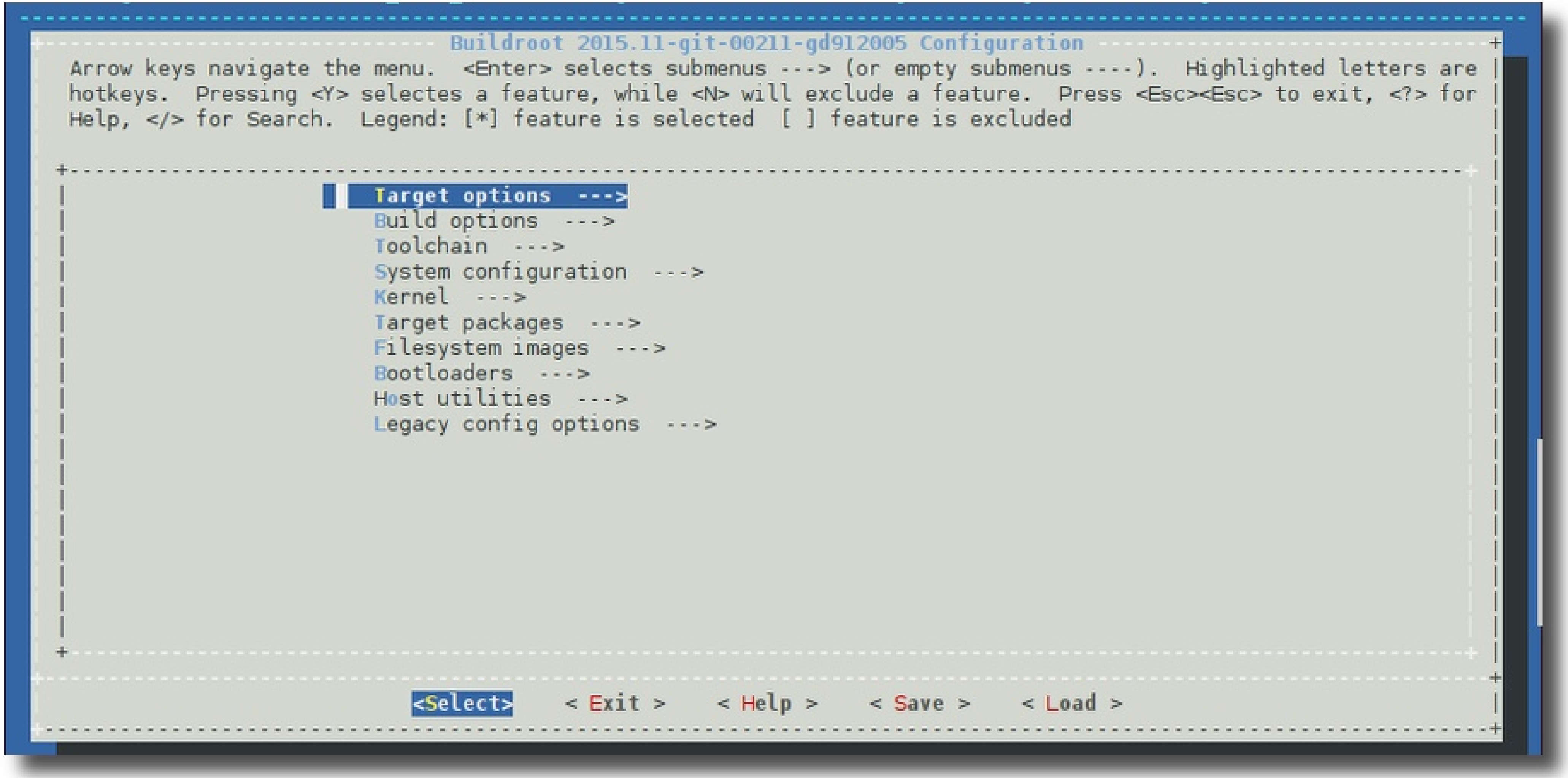
| Feature | Traditional Linux | Embedded OS |
|---|---|---|
| **Purpose** | General-purpose, versatile | Purpose-built, specialized for specific tasks |
| **Package Management** | Includes package managers (e.g., apt, yum) | No package manager, software is static |
| **Development Environment** | Supports on-target development with compilers and tools | Cross-compilation, development done on host system |
| **Hardware Requirements** | Higher CPU, memory, and storage requirements | Minimal hardware requirements, optimized for efficiency |
| **Flexibility** | Dynamic, can install/run new software post-deployment | Static, limited or no capability to add new software |
| **User Interface** | includes GUIs like GNOME or KDE | Typically no GUI, may have simple interfaces |

Figure 1: Interface of Builroot

# Introduction to buildroot

- **Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation.**
- **Key Benefits:**
  - Easy Configuration: Simple menu-based configuration tool.
  - Customizable: Adaptable to specific needs and hardware.
  - Extensive Support: Large number of pre-configured packages.
  - Buildroot supports numerous processors and their variants like ARM, x86,  PowerPC, RISC-V & ARC etc

# Introduction to buildroot



```
Buildroot 2015.11-git-00211-gd912005 Configuration
Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
hotkeys.  Pressing <Y> selectes a feature, while <N> will exclude a feature.  Press <Esc><Esc> to exit, <?> for
Help, </> for Search.  Legend: [*] feature is selected  [ ] feature is excluded

        Target options  --->
        Build options  --->
        Toolchain  --->
        System configuration  --->
        Kernel  --->
        Target packages  --->
        Filesystem images  --->
        Bootloaders  --->
        Host utilities  --->
        Legacy config options  --->

    <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 1: Interface of Builroot

# Embedded Linux vs Traditional OS

| Feature | Traditional Linux | Embedded OS |
|---|---|---|
| **Purpose** | General-purpose, versatile | Purpose-built, specialized for specific tasks |
| **Package Management** | Includes package managers (e.g., apt, yum) | No package manager, software is static |
| **Development Environment** | Supports on-target development with compilers and tools | Cross-compilation, development done on host system |
| **Hardware Requirements** | Higher CPU, memory, and storage requirements | Minimal hardware requirements, optimized for efficiency |
| **Flexibility** | Dynamic, can install/run new software post-deployment | Static, limited or no capability to add new software |
| **User Interface** | includes GUIs like GNOME or KDE | Typically no GUI, may have simple interfaces |

Figure 1: Interface of Builroot

# Steps to create your own embedded Linux OS

**Step 1 :** Navigate to Buildroot Directory *cd buildroot*

**Step 2 :** Open Buildroot Configuration Menu *sudo make menuconfig*

**Step 3 :** Set target architecture, select Linux kernel, and use default kernel configuration.

**Step 4 :** Build the System. *sudo make*

**Step 5 :** Navigate to Output Images Directory *cd output/images*

**Step 6 :** Copy Kernel Image and Root Filesystem Archive *cp -r bzImage rootfs.tar*

**Step 7 :** Navigate to Home Directory cd ~

**Step 8 :** Create Distribution Directory mkdir distro

**Step 9 :** Move Files to Distribution Directory mv bzImage rootfs.tar distro/

**Step 10 :** Navigate to Distribution Directory cd distro

# Steps to create your own embedded Linux OS

**Step 11 :** Extract Root Filesystem Archive _tar xf rootfs.tar_

**Step 12 :** Remove Root Filesystem Archive _rm rootfs.tar_

**Step 13 :** Navigate to Home Directory _cd ~_

**Step 14 :** Create Boot Image File _truncate -s 100MB boot.img_

**Step 15 :** Create Mount Directory _mkdir mounted_

**Step 16 :** Format Boot Image _mkfs boot.img_

**Step 17 :** Install extlinux Bootloader _sudo apt install extlinux_

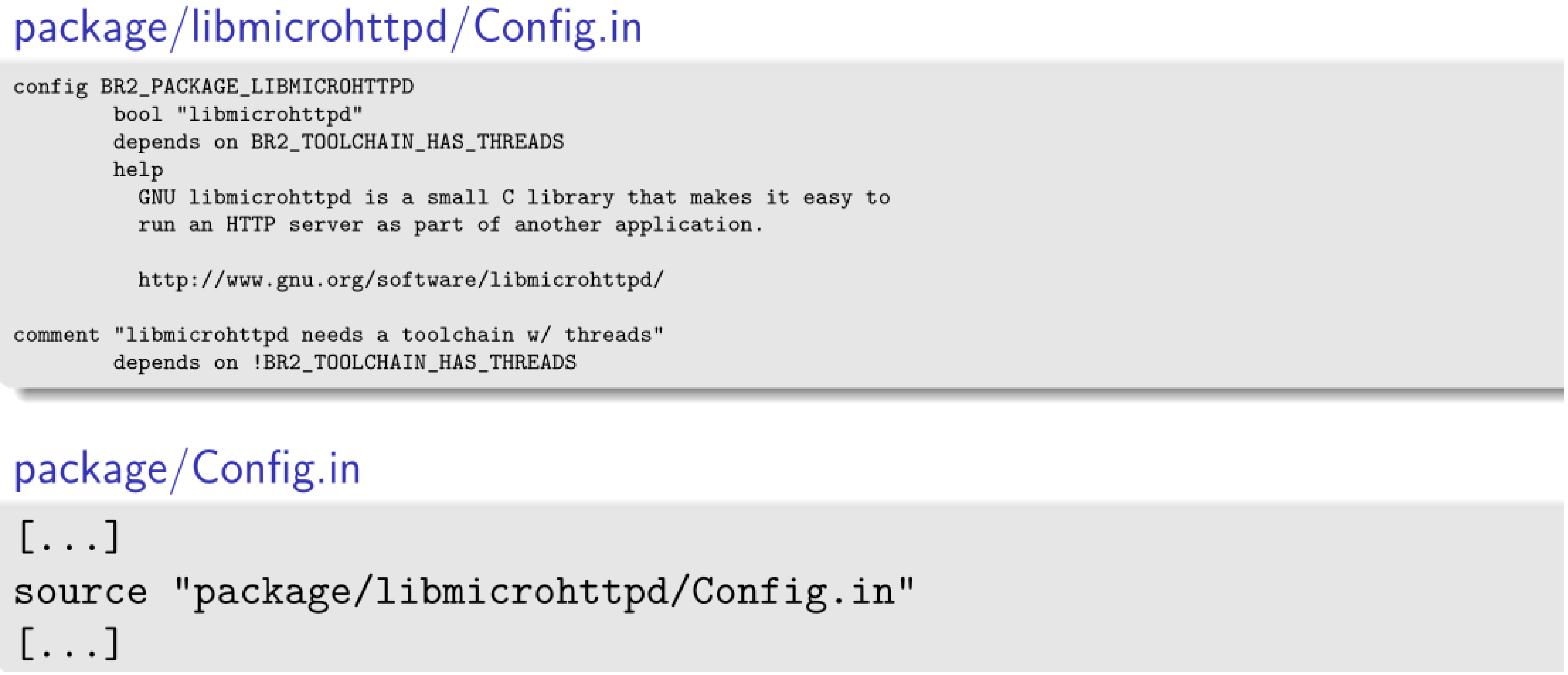
**Step 18 :** Mount Boot Image _sudo mount boot.img mounted/_

**Step 19 :** Install extlinux on Boot Image _sudo extlinux --install mounted_

**Step 21 :** Copy Distribution Files to Boot Image _sudo cp -r distro/* mounted_

**Step 22 :** Unmount Boot Image _sudo umount mounted_

# Adding a New Package: Config.in



package/libmicrohttpd/Config.in

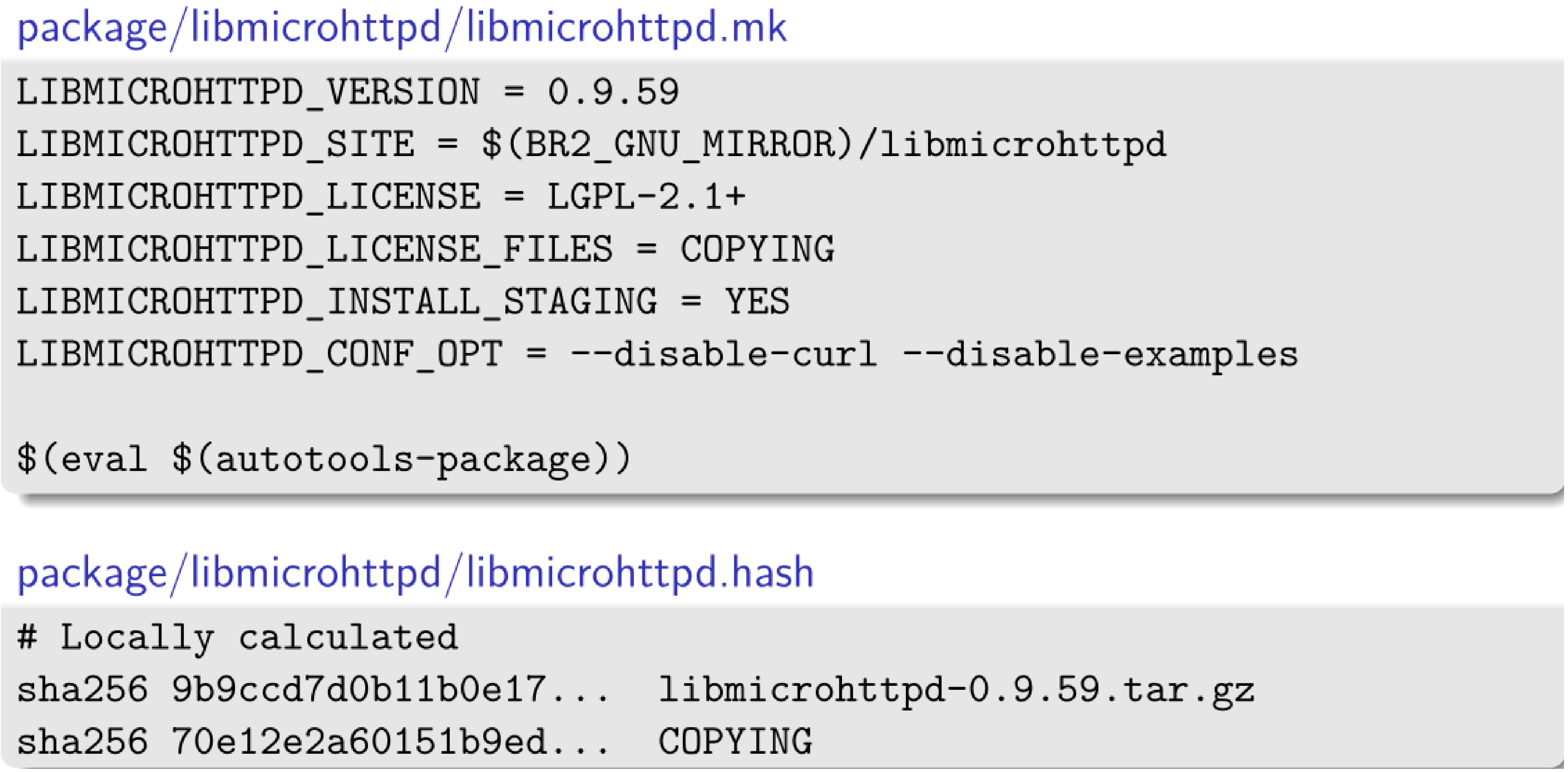
```
config BR2_PACKAGE_LIBMICROHTTPD
        bool "libmicrohttpd"
        depends on BR2_TOOLCHAIN_HAS_THREADS
        help
          GNU libmicrohttpd is a small C library that makes it easy to
          run an HTTP server as part of another application.

          http://www.gnu.org/software/libmicrohttpd/

comment "libmicrohttpd needs a toolchain w/ threads"
        depends on !BR2_TOOLCHAIN_HAS_THREADS
```

package/Config.in

```
[...]
source "package/libmicrohttpd/Config.in"
[...]
```

*Figure 2 : Adding a new package: Config.in*

# Adding a New Package: <pkg>.mk, <pkg>.hash



package/libmicrohttpd/libmicrohttpd.mk

```
LIBMICROHTTPD_VERSION = 0.9.59
LIBMICROHTTPD_SITE = $(BR2_GNU_MIRROR)/libmicrohttpd
LIBMICROHTTPD_LICENSE = LGPL-2.1+
LIBMICROHTTPD_LICENSE_FILES = COPYING
LIBMICROHTTPD_INSTALL_STAGING = YES
LIBMICROHTTPD_CONF_OPT = --disable-curl --disable-examples

$(eval $(autotools-package))
```

package/libmicrohttpd/libmicrohttpd.hash

```
# Locally calculated
sha256 9b9ccd7d0b11b0e17...  libmicrohttpd-0.9.59.tar.gz
sha256 70e12e2a60151b9ed...  COPYING
```

*Figure 3 : Adding a new package*

# How Much Powerful Linux It Can Provide?

- It can provide a embedded linux depending upon the hardware requirements( 32MB to GBs).
- Pre-defined configurations for popular platforms:

- ► RasberryPi
- ► BeagleBone
- ► CubieBoard
- ► PandaBoard
- ► Atmel development boards
- ► Several Freescale i.MX6 boards
- ► Many Qemu configurations

# Buildroot Design Principles

- **Cross-compilation only**: no support for doing development on the target.
- **No package management system**: Buildroot doesn’t generate a distribution, but a firmware
- **Don’t be smart**: if you do a change in the configuration and restarts the build, Buildroot doesn’t try to be smart. Only a full rebuild will guarantee the correct result.

# Thank You!