sagemcommunications

Open Source Linux Board Support Packages

RMLL – 1st July 2008

Outline

- BSP Definition
- OSADL BSP Specification
- Embedded Linux Systems - Quick Overview
- The need for a Linux BSP
- An Open Source BSP: Why ?
- Examples of Open Source Linux BSPs
- Advantages – Drawbacks
- Conclusion

BSP Definition

Wikipedia definition:

In embedded systems, a Board Support Package (BSP) is implementating specific support code for a given board that conforms to a given operating system. It is commonly built with a bootloader that contains the minimal device support to load the operating system and device drivers for all the devices on the board.

Definition:

Board Support Packages (BSP) provide a base for Embedded Linux projects. They adapt the generic Linux kernel to the vendor specific hardware platform, implementing the board-specific initializations and configurations required to run Linux on that platform.

OSADL BSP Specification

- Aims at providing quality criteria of a BSP, closely adapting best practice rules from the kernel community ➔ ***Kernel Modifications Rules***

- Following these rules makes it possible for different parties to interchangeably work on a particular BSP during the product life cycle.

- Defines several levels of conformance, making it easier for industrial end users to decide about a BSP
  - ➔ ***Patch handling mechanism levels***
  - ➔ ***BSP toolchain levels***

Embedded Linux Systems – Quick Overview

Introduction

- Embedded Linux is growing rapidly and is becoming widely used (mobile phones, PDA, networking equipments, etc)

- Linux embedded systems made it easier for a greater number of free software modules to be deployed in embedded products (sqlite, samba, thttpd etc)

- Linux is present even in resource critical devices (Uclinux, Uclibc, dietlibc etc)

Embedded Linux Systems – Quick Overview

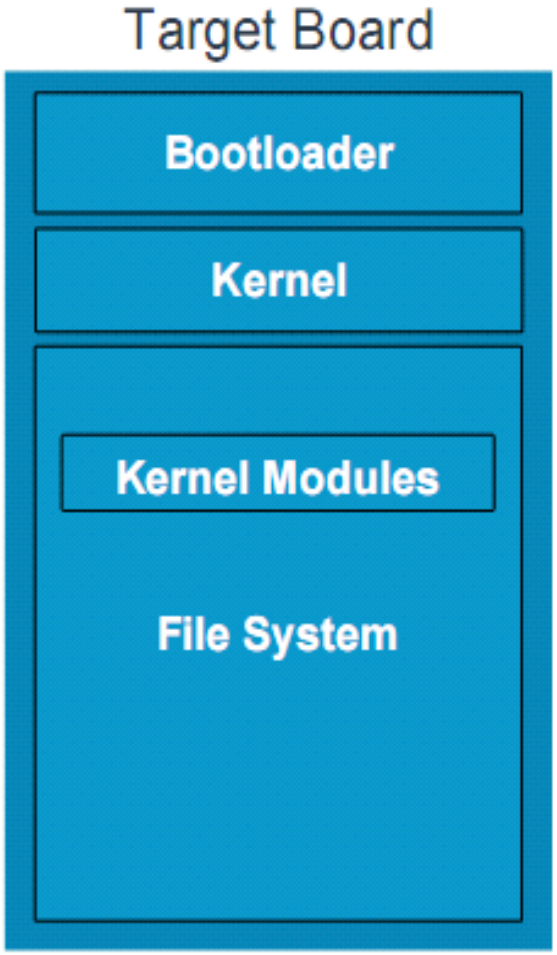
Development Phases

- **Configuration :**
  - **Choose functionalities to include into the system.(ususally lkc format)**
- **Build :**
  - **Build components (kernel, applications, libs)**
  - **Combining all these elements to create a file system (NFS in dev phase and JFFS2 in deploy phase)**
- **Deployment :**
  - **Installation (loading) of the generated file system in the target board and restart.**

Embedded Linux Systems – Quick Overview

Basic Components

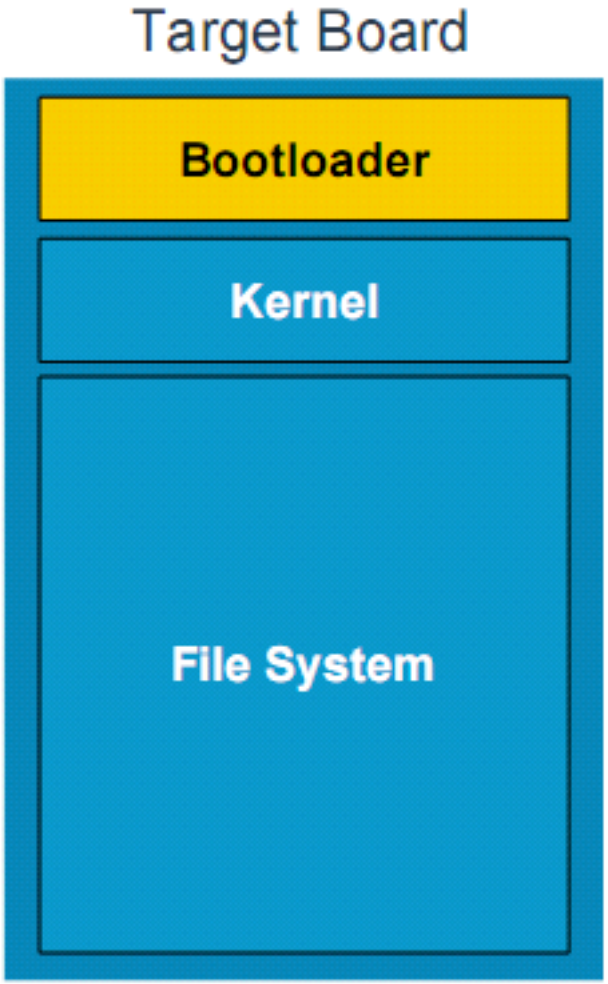
- Bootloader
- Kernel
- Kernel Modules
- File System
- Shared Libraries
- Applications



Embedded Linux Systems – Quick Overview

Basic Components

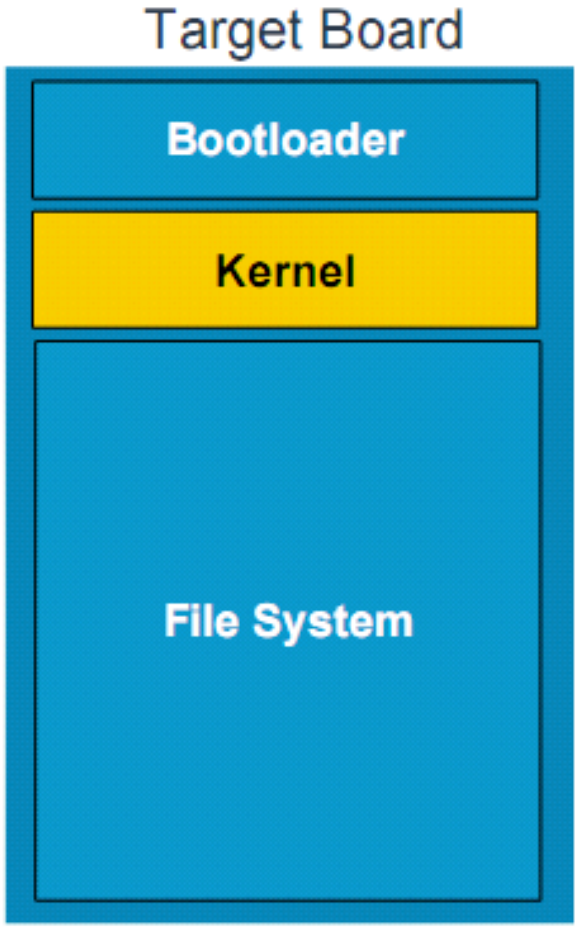
- **Bootloader**
- Used to initialize the board
- Provides a mechanism to boot kernel
- Configured and built for specific target board
- Common Architecture bootloaders
  - **ColdFire® : Colilo, u-boot, dBUG**
  - **Power Architecture™ : u-boot**
  - **ARM ™ : blob, redboot, u-boot**



Embedded Linux Systems – Quick Overview

Basic Components

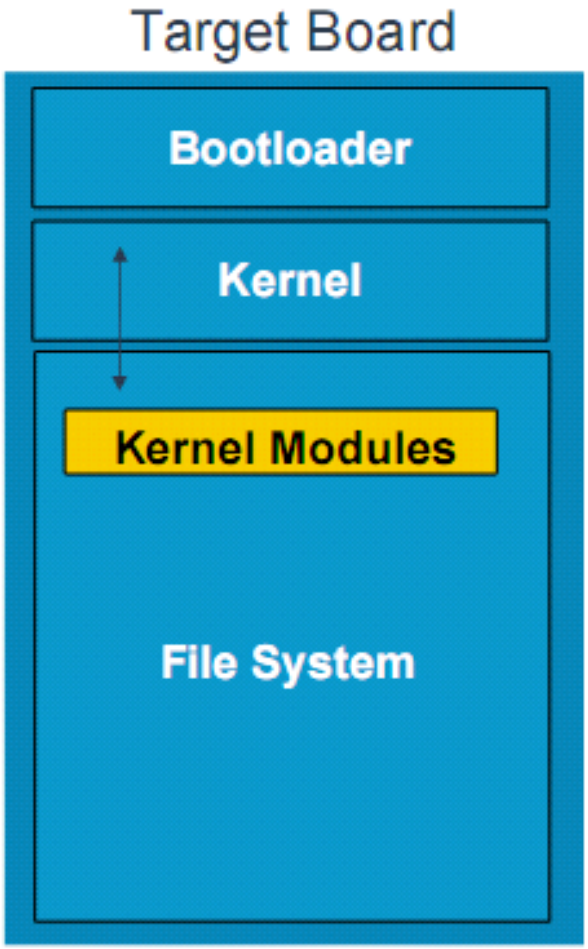
- **Kernel**
  - Continued initialization of the board
  - Provides a mechanism to interact with devices (drivers)
  - Provides underlying protocol support (TCP/IP) and common OS services



Embedded Linux Systems – Quick Overview

Basic Components

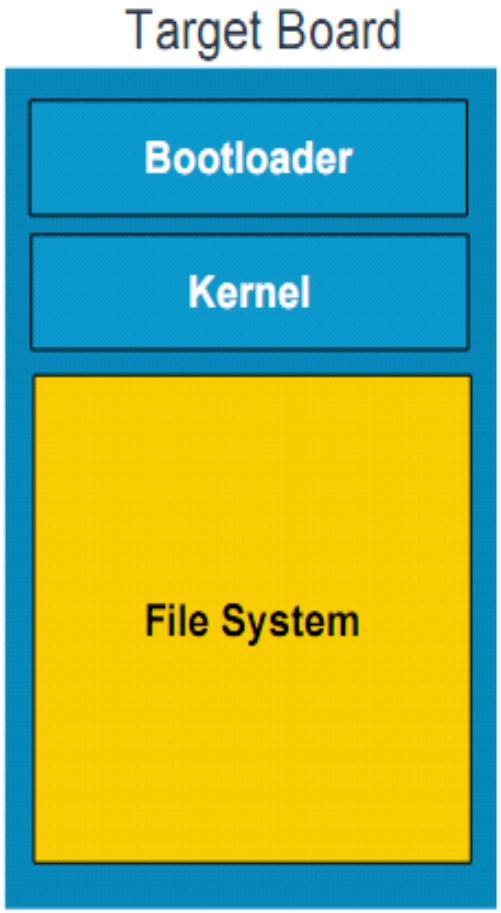
- **Kernel Modules**
- Device drivers
- Provides additional functionality to kernel
- Resides in the file system and can be loaded and unloaded from the kernel.



Embedded Linux Systems – Quick Overview

Basic Components

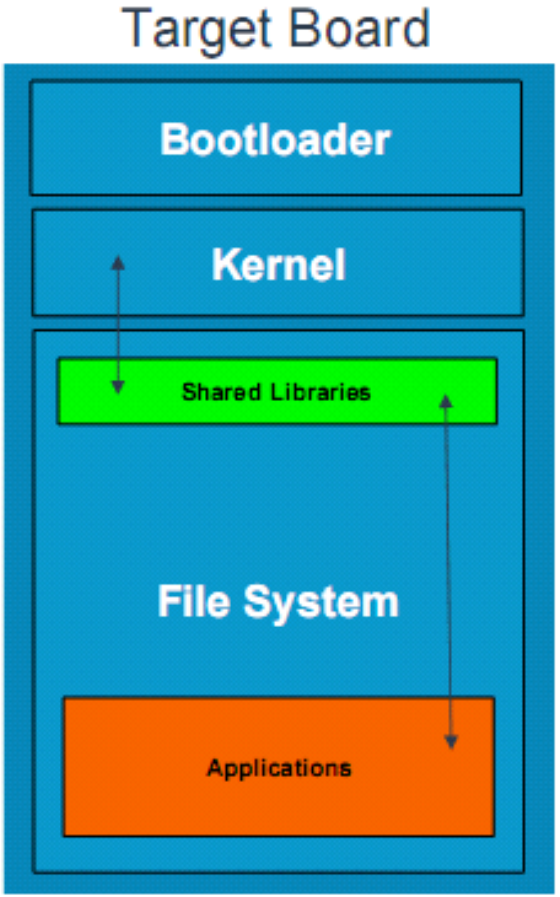
- **Basic File System**
  - Protected by Memory Management Unit (MMU) (user land)
  - Applications live here
  - Common embedded File System types
    - **Network File System (NFS)**
    - **ramdisk**
    - **Journaling Flash File System version 2 (JFFS2)**



Embedded Linux Systems – Quick Overview

Basic Components

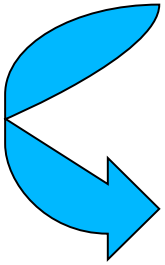
- **Applications**
  - Provides functionality to the system
  - Resides in the File System
  - Access to kernel functionality via the shared librarie
  - Cannot directly access kernel space (protected memory)
  - Must be compiled against the same version of share library that is located on the embedded system



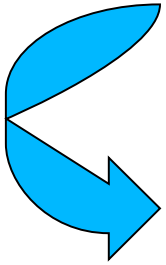
The need for a Linux BSP

In order to be able to generate an embedded linux system:

- cross-compile and manage dependencies between
  - **Kernel**
  - **librairies (static and shared)**
  - **applications (C, C++, java, …)**



How to automate these tasks and ease the generation of an embedded Linux system ?



Linux Board Support Packages

An Open Source BSP : Why ? (1/2)

- Same reasons for using free software:
  - Large community
  - Freedom of use and modification
  - Possibility to add value

- For the industry:
  - Avoid to be dependent on a board manufacturer
  - Accelerate TTM by relying on a larger community of contributors
  - Rapidly integrate open source LGPL software modules in products
  - Garanty portability and ease of maintenance of development environments
  - Can switch rapidly to a lower cost board without loosing too much time on porting a BSP
  - Concentrate effort on core business applications(differentiation)

An Open Source BSP : Why ? (2/2)

The educational sector:

- **Make it easier for students to reproduce lab exercises anywhere they are (Internet connection)**
- **The availability of BSP source code and the possibility to integrate user applications is a good learning experience**
- **Possibility to reproduce lab exercises for different architectures and boards**

Examples of Linux Open Source BSPs

- Buildroot
- LTIB
- OpenEmbedded

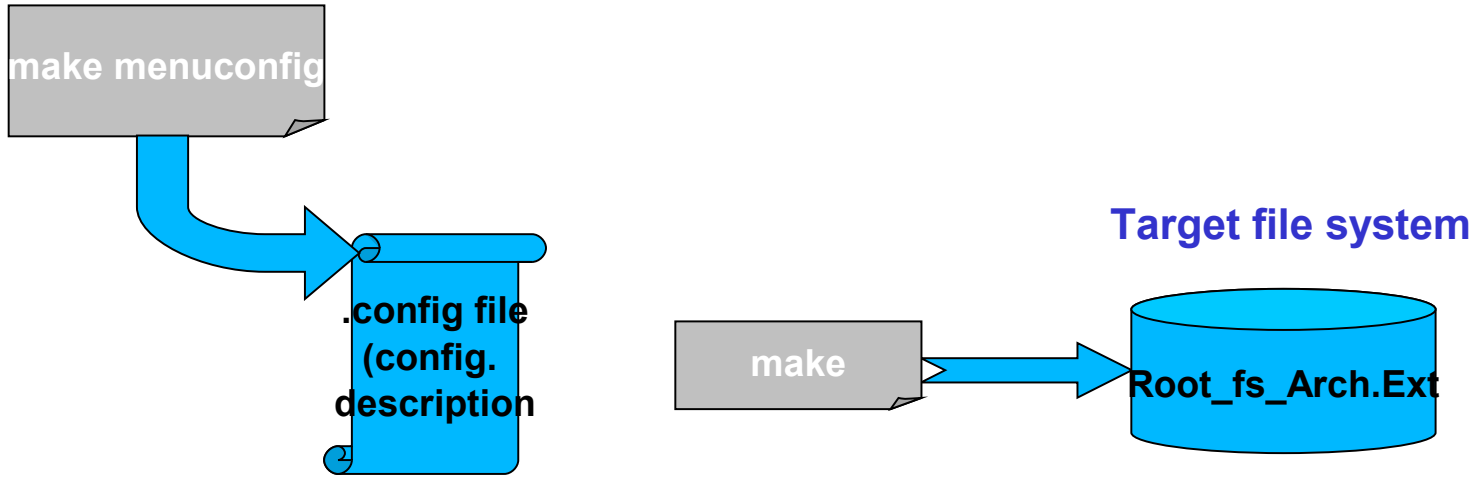
Buildroot (1/3)

- A set of Makefiles and patches that allow to easily generate both a cross-compilation toolchain and a root file system for the target board.

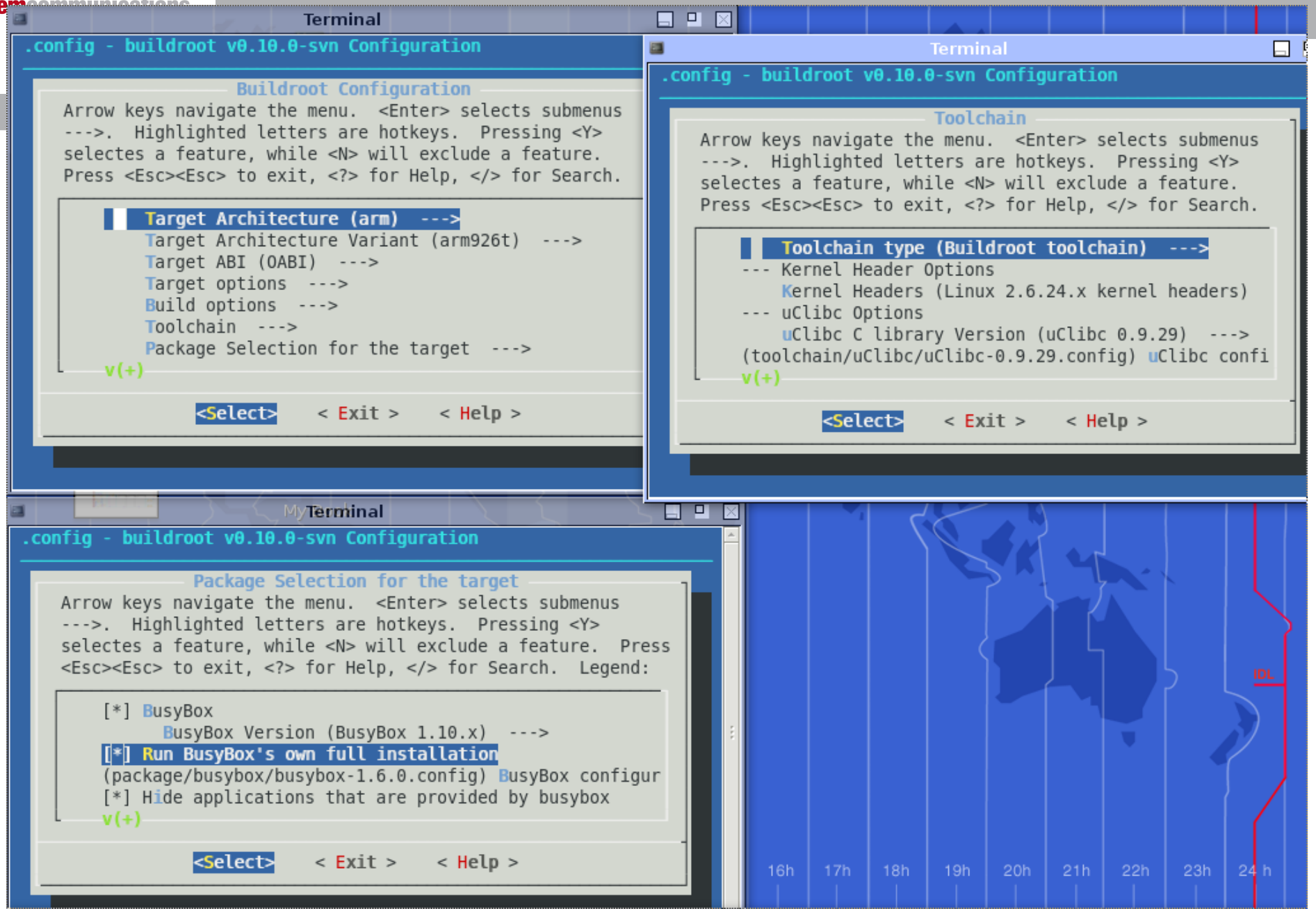
- The cross-compilation toolchain uses uClibc (http://www.uclibc.org/), a tiny C standard library.

- Automates the building of a root file system with all needed tools like *busybox*.

Buildroot (2/3)



- *ARCH* : architecture
- *EXT :* depends on the type of target file system selected in the Target options section of the configuration tool.
- The file is stored in the "*binaries/$(PROJECT)/*" directory


Terminal
.config - buildroot v0.10.0-svn Configuration
Buildroot Configuration
Arrow keys navigate the menu. <Enter> selects 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.
Target Architecture (arm) --->
Target Architecture Variant (arm926t) --->
Target ABI (OABI) --->
Target options --->
Build options --->
Toolchain --->
Package Selection for the target --->
v(+)
<Select> < Exit > < Help >
Terminal
.config - buildroot v0.10.0-svn Configuration
Toolchain
Arrow keys navigate the menu. <Enter> selects 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.
Toolchain type (Buildroot toolchain) --->
--- Kernel Header Options
Kernel Headers (Linux 2.6.24.x kernel headers)
--- uClibc Options
uClibc C library Version (uClibc 0.9.29) --->
(toolchain/uClibc/uClibc-0.9.29.config) uClibc confi
v(+)
<Select> < Exit > < Help >
Terminal
.config - buildroot v0.10.0-svn Configuration
Package Selection for the target
Arrow keys navigate the menu. <Enter> selects 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:
[*] BusyBox
BusyBox Version (BusyBox 1.10.x) --->
[*] Run BusyBox's own full installation
(package/busybox/busybox-1.6.0.config) BusyBox configur
[*] Hide applications that are provided by busybox
v(+)
<Select> < Exit > < Help >
IDL
16h
17h
18h
19h
20h
21h
22h
23h
24 h

LTIB (1/3)

- Freescale GNU/Linux Target Image Builder is a tool created by Freescale, that is used to build Linux target images, composed of a set of packages.
- LTIB has been released under the terms of GNU GPL
- LTIB BSPs draw packages from a common pool.
- All that needs to be provided for an LTIB BSP is :
  - **Cross compiler**
  - **Boot loader sources**
  - **Kernel sources**
  - **Kernel configuration**
  - **Top level config file : main.lkc**
  - **BSP config file : defconfig**

LTIB (2/3)

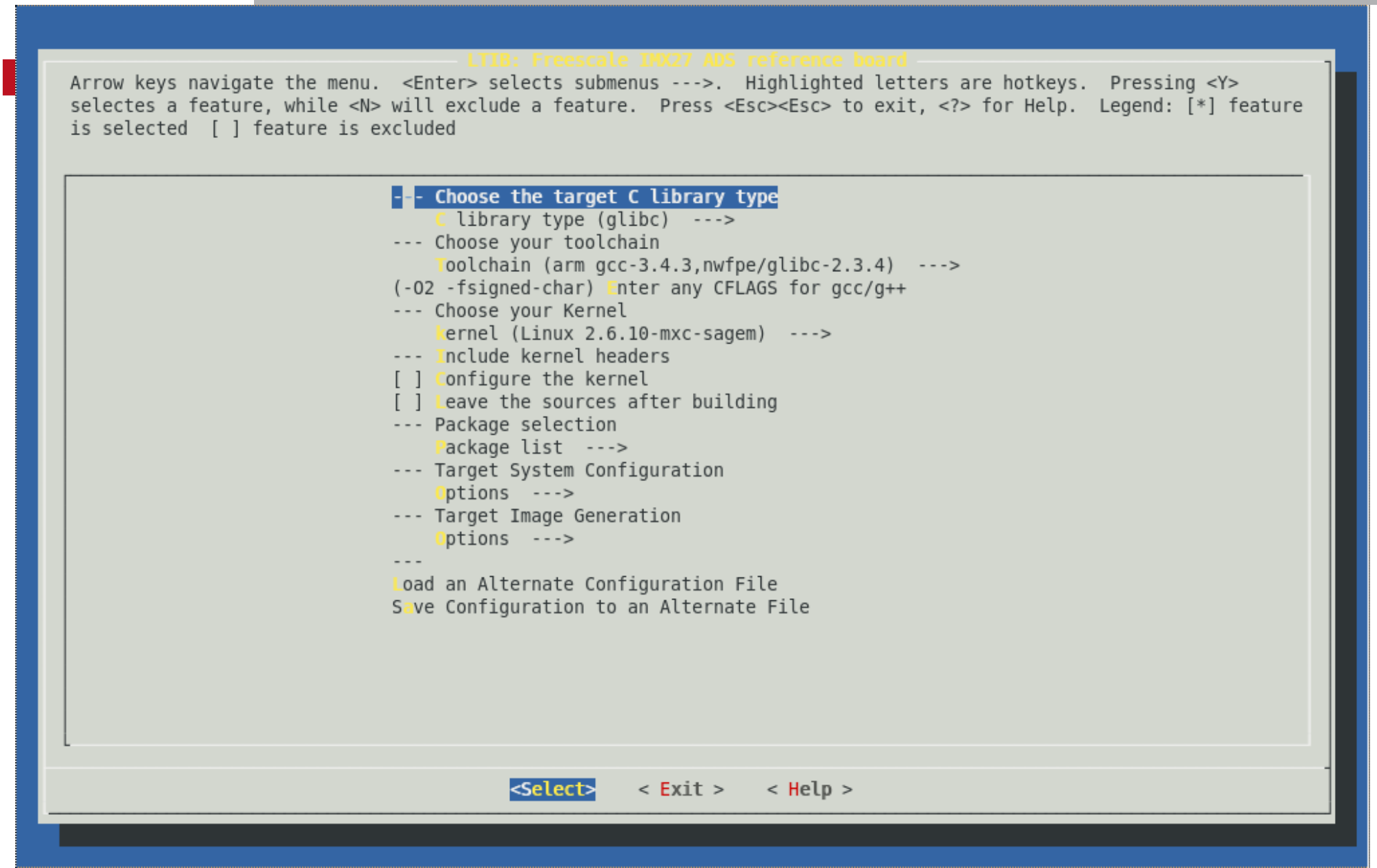
- To configure and build Linux OS for the target:

  ./ltib --configure

Upon exit, LTIB will execute everything needed to regenerate the image files in accordance with the desired configuration

- Most users initially will not have to make any changes to the BSP

- Supports FreeScale cards by default


LTIB: Freescale IMX27 ADS reference board
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> selectes a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] feature is selected [ ] feature is excluded
--- Choose the target C library type
C library type (glibc) --->
--- Choose your toolchain
Toolchain (arm gcc-3.4.3,nwfpe/glibc-2.3.4) --->
(-O2 -fsigned-char) Enter any CFLAGS for gcc/g++
--- Choose your Kernel
Kernel (Linux 2.6.10-mxc-sagem) --->
--- Include kernel headers
[ ] Configure the kernel
[ ] Leave the sources after building
--- Package selection
Package list --->
--- Target System Configuration
Options --->
--- Target Image Generation
Options --->

Load an Alternate Configuration File
Save Configuration to an Alternate File
<Select>
< Exit >
< Help >

OpenEmbedded

- OpenEmbedded is a build system, based on OpenZaurus, that can generate (cross-compile) Software packages for embedded targets, including Bootloader, Linux and Applications

- Able to handle different hardware architectures, support multiple releases for those architectures, and uses tools for speeding up the process of recreating the base after changes have been made.

- Uses compilation and configuration caching at most levels to increase developer productivity.

- Uses Bitbake task executor + OE Metadata

Advantages - Drawbacks

Buildroot

- Advantages:
  - Supports most architectures
  - Automatically downloads tool sources and applies patches
  - Can build most applications needed (except your custom application): 194 supported in May 2007. Examples: *BusyBox, bzip2, Cairo, dbus, Dillo, DirectFB, Dropbear, lighthttpd, Python, Qtopia4, sqlite, thttpd, tinyX, Xorg*
  - Very easy to use
- Drawbacks:
  - Does not support glibc-based toolchains

Advantages - Drawbacks

LTIB

- Advantages:
  - Works out of the box for Freescale boards
  - Integrates a new application module easily
  - Fast execution
  - Supports multiple file systems formats
  - Possibility to integrate user defined toolchains and kernels
- Drawbacks:
  - Does not support multiple architectures
  - Relatively small community
  - Reduced number of supported boards
  - Porting from a freescale board to a different manufacturer one is a tedious task

Advantages - Drawbacks

OpenEmbedded

Advantages:

- Supports various platforms (ARM, MIPS, X86, AVR32, etc)
- Flexible: could create one package, one toolchain, or a complete distribution
- Famous Telecom companies get involved (Nokia, Sharp, Siemens, etc)
- Can use other BSP's toolchains (Buildroot)
- Supports many FS types (cramfs, ext2/ext3, jffs2, etc)
- Supports various PMS (deb, ipkg, rpm, tarballs)
- Supports various repository types (git, svn, monotone, http, ftp, local disk)
- Benefits from Bitbake advantages

Drawbacks:

- Uses Monotone as Version Control System
- No stable branch (continuous work)
- Steep learning curve

Conclusion

- A stable and viable free software BSP is an important brick in the continuous effort to generalise the use of linux and free software modules in embedded industrial applications and products

- OE is an attractive candidate for a widely used and portable multi-plate forme free software Linux BSP