Joint ICTP-IAEA School on Zynq-7000 SoC and its Applications for Nuclear and Related Instrumentation

Introduction to Embedded Linux

Fernando Rincón
fernando.rincon@uclm.es

Contents

- Embedded OS
- Embedded Linux
- Linux Architecture

Embedded OS

- Embedded systems usually perform very simple tasks
  - Monitor status of sensors
  - React to events following simple rules
    - Move motors
    - Activate relays
    - Send data
    - ...
- Complex applications may have more requirements
  - Concurrent execution
  - Real-time management
  - Resource multiplexion
  - Advanced communication mechanisms
- Purpose of the OS → Release the user from those complex tasks

Desktop vs Embedded

- Main differences
  - Small footprint, and tailored to the specific need
  - Custom peripheral drivers.
  - Less protection mechanisms
    - Even no Memory management Unit (MMU)
  - Real-time
  - Lack of very simplistic user-interface
  - Sometimes no network connection
    - Although the trend is to have everything connected

Embedded OS Development

- Cross-development
  - Host machine: develoment enviroment and tools
    - Cross-compiler
      - Different versions for bare-metal and OS development
        - Mostly because of the linked libraries
        - Ex: arm-none- vs arm-xilinx-linux
    - Target root filesystem & basic libraries for compilation purposes
  - Target machine:
    - Bootloader
    - Kernel
    - Root filesystem
- Cross-platform design cycle
  - Compile → test → debug
  - Use of emulators
- Bootstrap process:
  - Flash
  - Network
  - ...

Why Linux?

- Low cost of ownership
- Open-source, full control and customizability
- Rich feature set
- Quality of Linux components
- Re-use of Linux components
- Big active developer community & resources
- Pool of talented embedded Linux developers
- Scalability, from embedded to the cluster
- Standard tools and run-time environment

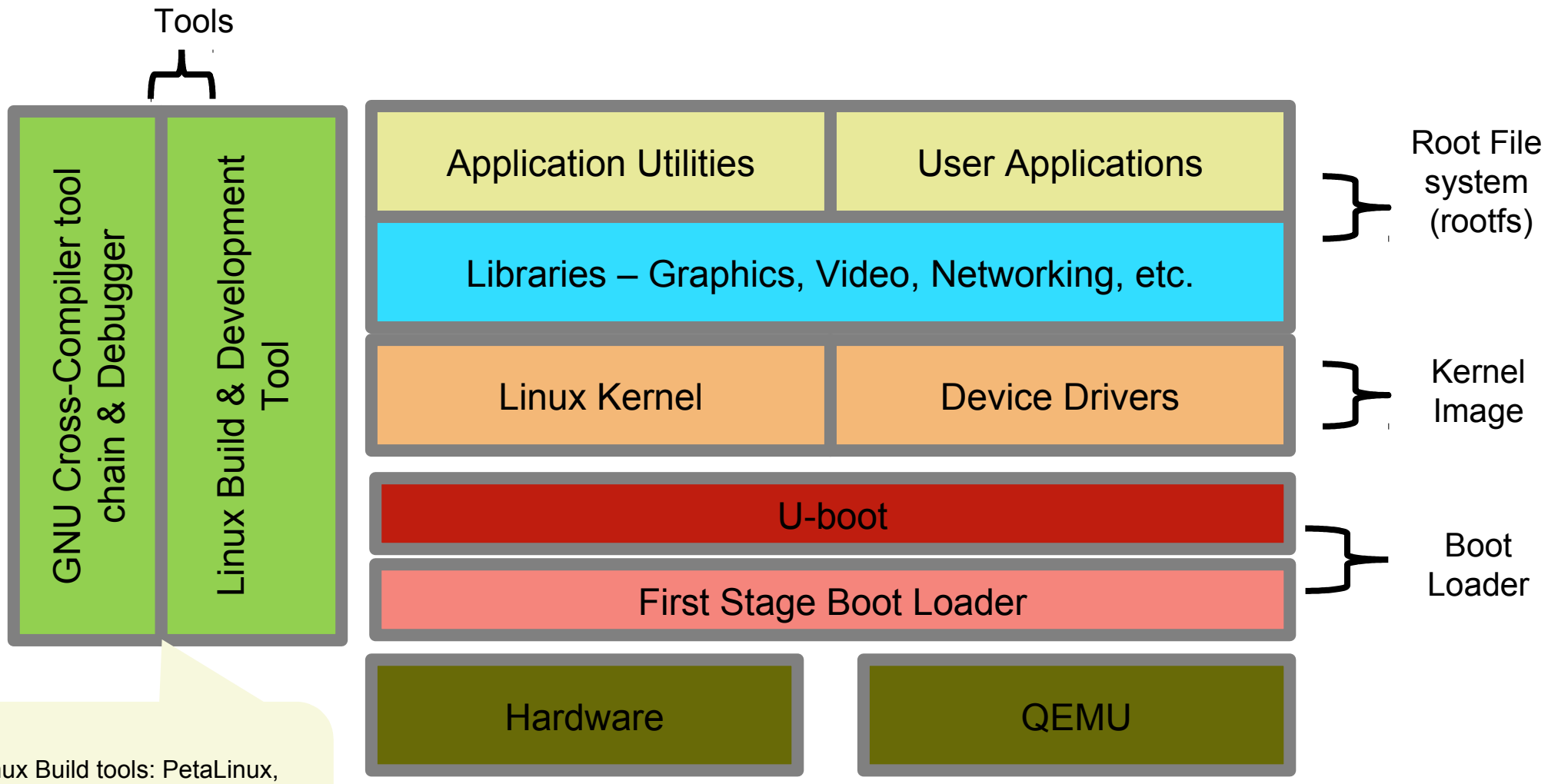
What does Linux Offer?

- Multiple processor achitectures supported
  - ARM, MIPS, PowerPC, x84, amd64, Microblaze, …
- Reduced foot print (a few MB) and fast booting times (~4 s)
- Networking:
  - full support for TCP/IP, Wifi, Bluetooth, USB, …
- Graphics:
  - X, Frame Buffer & DRM/KMS.
  - Graphical development libraries QT, Gnome, …
- Multimedia:
  - Video4Linux & Gstreamer, FFMPeg, OpenCV, …

What Does Linux Offer?

- Interoperability and OS infrastructure
  - Networking, file systems, device support
  - Scheduling, processes/threads, interrupt handling, interprocess communication, symmetric multiprocessing (SMP)
- Developer familiarity
  - Standard tools
  - Standard run-time environment
  - Application code can be prototyped on the desktop
- Scalability
  - Deeply embedded → single board computers → desktop → server → cluster
- Freedom and openness
  - Accessibility to modify source code

Linux Components

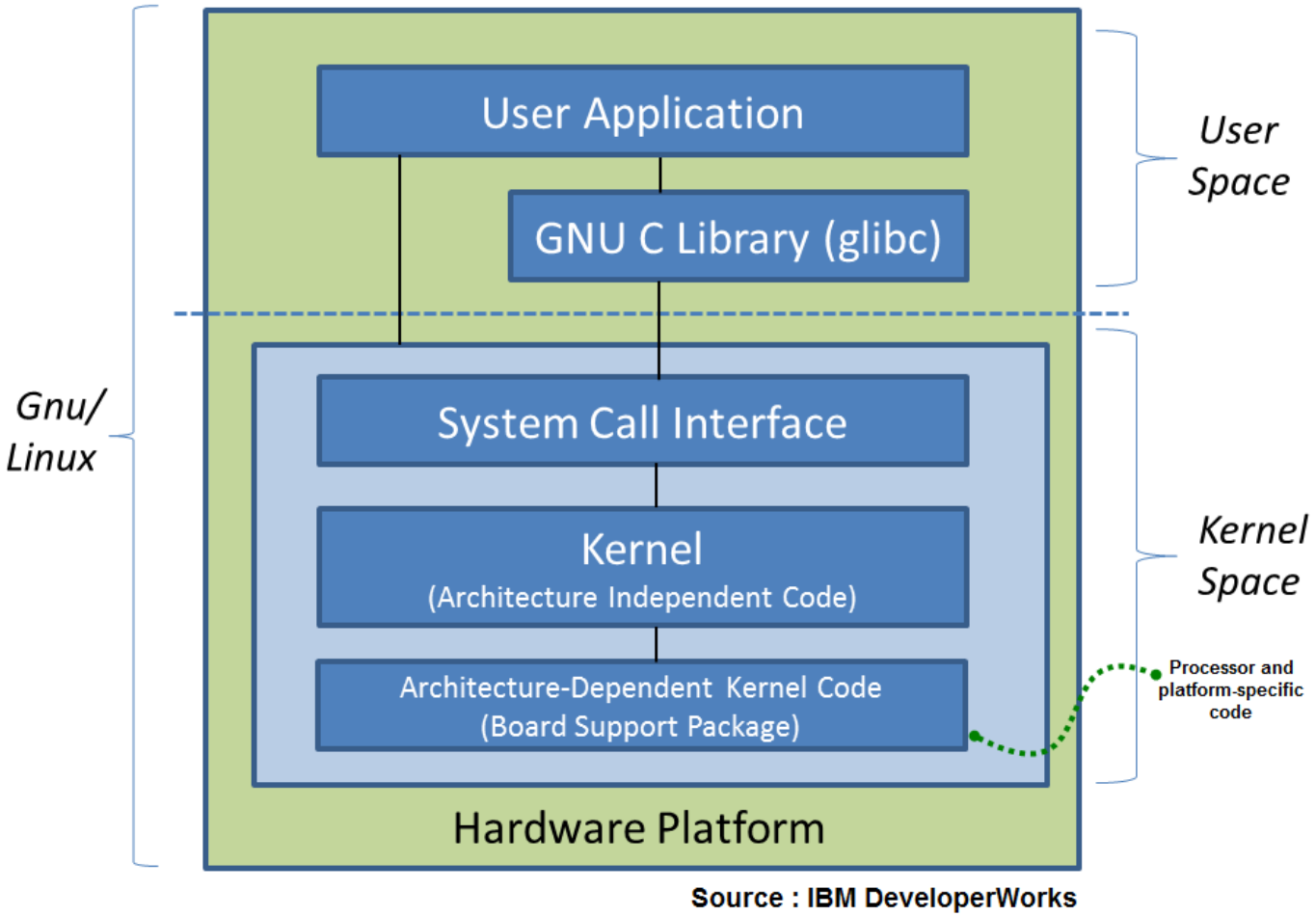


Linux Build tools: PetaLinux, Yocto, OpenEmbedded, buildroot
Other tool chains: Linaro, GCC mainline

Board Support Package (BSP): Collection of SoC/board specific Linux components that include Linux kernel & device drivers, boot loader and tools

Anatomy of a Linux System

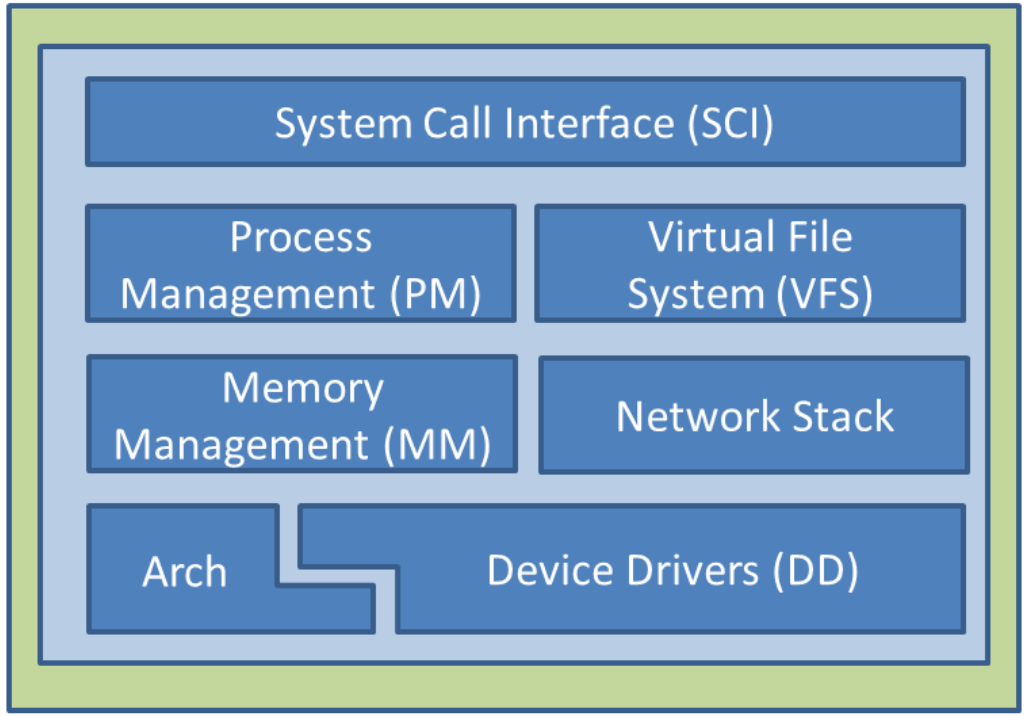
- User applications
  - Task specific
  - Libraries
  - C run time, networking
  - graphics, for example
- Kernel
  - More on this soon
- Hardware
  - CPU, memory
  - Timer, interrupt control



Source : IBM DeveloperWorks

Linux Kernel Architecture Overview

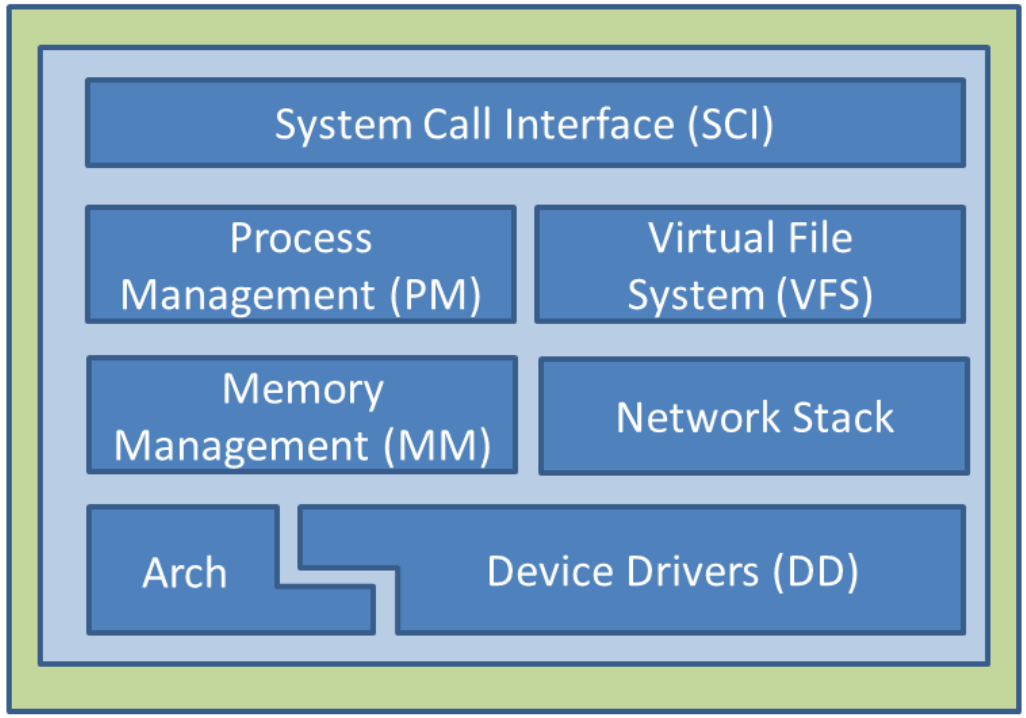
- System call interface (SCI)
  - Function calls from user space to kernel
- Process management
  - Processes or threads
  - Share the CPU between the active threads
- Memory management
  - Manages the virtual and physical memory
- Virtual file system
  - Switching layer between the SCI and the file systems supported by Linux



Source : IBM DeveloperWorks

Linux Kernel Architecture Overview

- Network stack
  - Above TCP, socket layer
  - Invoked through the SCI
- Device drivers
  - Supports most of the common devices
- Architecture-dependent code (BSP)
  - Processor and platform-specific code



Source : IBM DeveloperWorks

System Call Interface

- Contract between user space and kernel space
  - Request services
  - Memory, I/O
  - Process management
- The standard C library is a wrapper around the system call interface
  - For example, `malloc()` is a C library call
    - Maintains a pool of memory
    - Calls kernel memory allocator only when that pool is exhausted

Virtual Memory

Linux distributions