



Embedded Linux Solutions

General

- **The Linux kernel is a UNIX like OS kernel.**
- **Developed by contributors world wide.**
- **Released under GNU public license (GPL).**
- **The Linux vanilla tree is the main Linux distribution tree managed and maintained by the Linux consortium.**

General - Cont.

- **Vast & fast HW support Availability.**
- **Low cost In-House Linux distribution.**
- **Linux is designed for MMU based systems.**

Open Source

- **Open source is an approach to design, develop and distribute accessibility to a product's source (goods and knowledge).**
- **Linux is developed by a world wide open-source community.**
- **Development community tends to support new protocols, HW platforms & peripherals faster than RTOS vendors.**
- **Open source - all sources are available.**

GPL

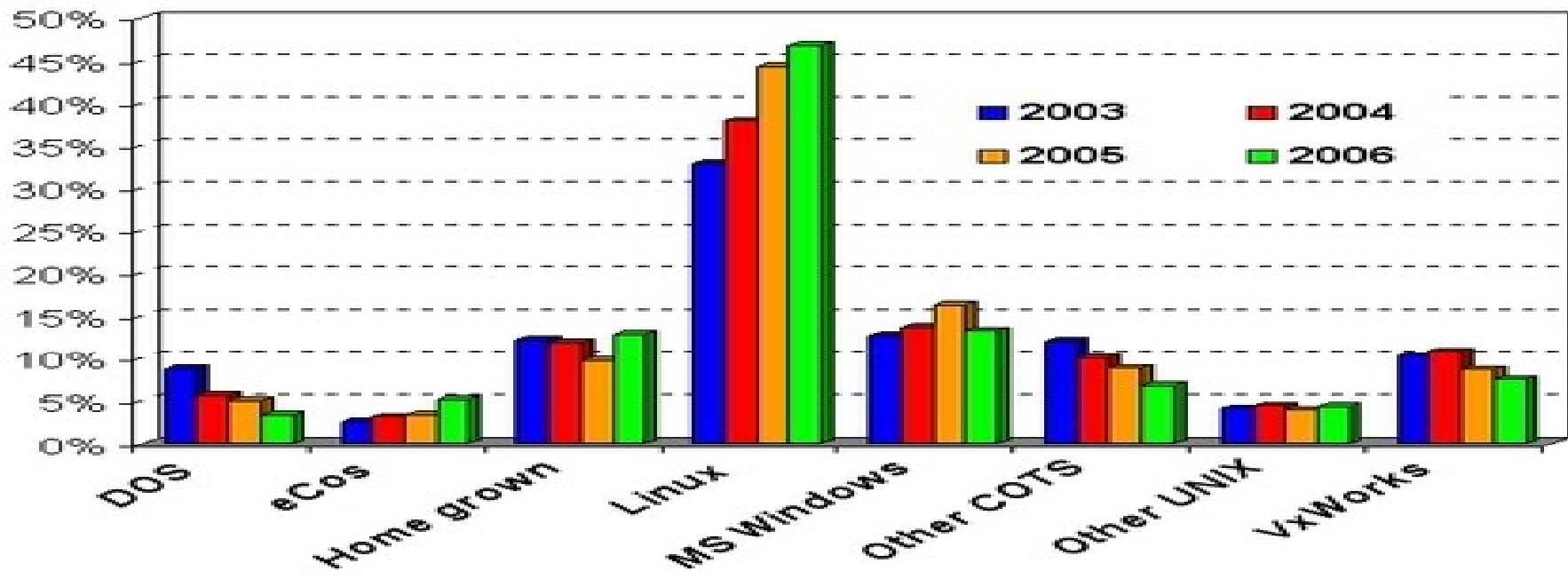
- **GNU General Public License**
- **Widely used free software license, originally written by Richard Stallman for the GNU project**
- **Requires derived works to be available under the same “copyleft”**
- **Ensures software freedom is preserved, even when the work is changed or added to.**

Business Model

- **Choose your Business model**
- **Multiple vendors offer embedded Linux product solutions with support, training and professional service options, using a variety of business models.**
- **Vendor independence - Create your In-house Linux dist (low cost)**

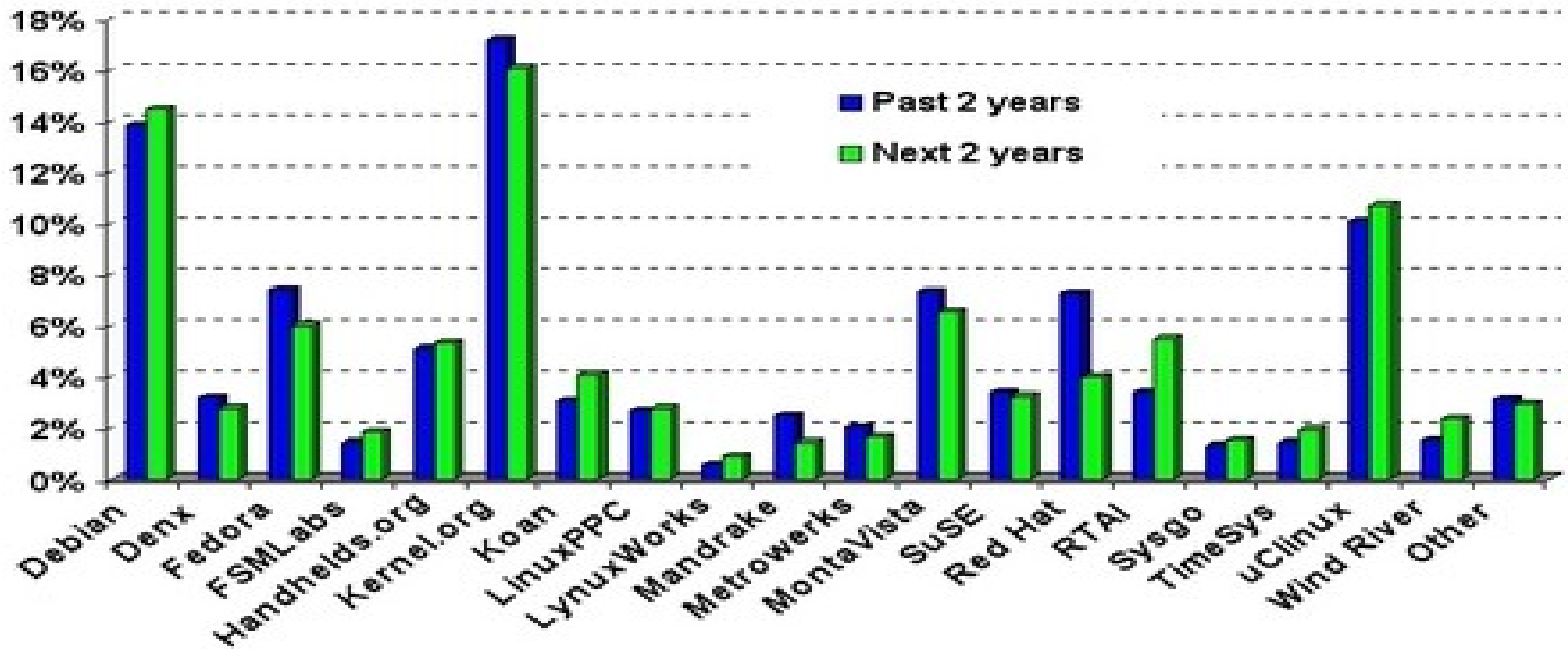
Embedded OS Trends

Embedded OS sourcing trends



Linux in embedded systems

Linux sourcing trends in embedded device applications



Why Use Linux?

- **Open Source**
- **Reliability**
- **Scalability**
- **Security**
- **Extensively supports most network communication protocols**
- **Initial startup costs are extremely small**
- **Royalty Free!**
- **Support – Linux Community, Leverage Unix experience base(POSIX)**
- **Broad Range of Hardware Support**
- **High-Performance Networking**

Focus on Linux – high level view



Hardware/Software Compatibility

- **Linux can run on most microprocessors with a wide range of peripherals and has a ready inventory of off-the-shelf applications.**
- **Various Linux kernel trees for different architectures are maintained using the git source code repositories.**
- **Software compatibility checks for the chosen hardware (choosing compatible boot loader, kernel and tool chain).**

Software, firmware and hardware suppliers support Linux

SOFTWARE

Device Drivers

File Systems

Networking Protocol Stacks

Middleware

Multimedia Software and CODECs

Databases

Development and Test Tools

HARDWARE

CPUs and Chipsets

Board and Bus Support (esp. CompactPCI*/AdvancedTCA*)

Peripherals and Peripheral Cards

Graphics Chipsets and Cards

Connectivity: USB, FireWire*, Ethernet, Bluetooth*, Wi-Fi*

Acceleration Technologies (IP Acceleration, Security, Network Processors)

Hardware Debugging and Test Tools

Security

Designs can be more secure when using Linux,

Take advantage of mature Linux kernel features and Open Source projects and products.

- **IPchains firewall**
- **IP tunneling**
- **Ipsec**
- **SSH/SSL**
- **PAM**
- **LSM**
- **Medusa**

Memory Architecture – Legacy RTOS

- **All code and data in RAM are exposed to accidental (or malicious) modification by any and all programs running in the system**
- **The OS itself and all memory-mapped I/O are also exposed to modification**
- **Task/thread stacks can overwrite one-another (underflow)**
- **Overwriting and other untoward access—when it does occur—happens “silently” and may go undetected for arbitrarily long times (unbounded)**

Memory Architecture – Linux

- employs *virtual addressing* wherein all programs in the system (including the Linux kernel) operate in and with *logical addresses*
- Linux virtual addressing presents a more robust application and system programming model to the programmer
- Applications execute in their own protected address spaces, and are, for the most part, invisible to one another
- They are also prevented from overwriting their own code through the use of hardware-based Memory Management Units (MMUs) present on most modern 32- and 64-bit processors.

Memory Architecture – Comparison

| | RTOS | LINUX |
|-------------------------------|-----------------------|-----------------------------------|
| ▪ Protection Type: | Ad hoc, if any | Process / Page-based |
| ▪ Application Data: | Exposed | Protected |
| ▪ Application Code: | Exposed | Protected |
| ▪ OS Kernel Code: | (RAM) Exposed | Protected |
| ▪ OS Kernel Data: | Exposed | Protected |
| ▪ Tasks/Thread Stacks: | Exposed | Protected with Guard Pages |
| ▪ Memory-Mapped I/O: | Ports Exposed | Protected, mapped |

Fault Response Comparison

| | RTOS | LINUX |
|-----------------------------------|---------------------|-------------------------------|
| ▪ Program Unit: | Task | Process / Thread |
| ▪ Failure Granularity: | Whole System | Single Process |
| ▪ Fault Notification: | None | Parent-Child / Signals |
| ▪ Safe Task Restart | No | Yes |
| ▪ Grow Resources | No | Yes |
| ▪ Dynamic Memory Recovery | No | Yes |
| ▪ Recover System Resources | No | Yes |

Real Time

- **Linux was designed to be a general-purpose operating system.**
- **Real-time improvements of Linux were made, so that open source software availability, can be used for real-time applications.**
- **Kernel preemption patch - Linux kernel is modified to reduce the time the kernel spends in non preemptive code sections.**
- **Real-time executive approach - small real-time kernel coexists with the Linux kernel.**

Real Time - Cont.

Real-time extensions to Linux 2.6

- **Improved O(1) scheduler**
- **Improved SMP scaling from 8 or 8 CPUs to 32, 64 and beyond.**
- **Support for true asynchronous I/O.**
- **high resolution timers**

Hard real-time in Linux

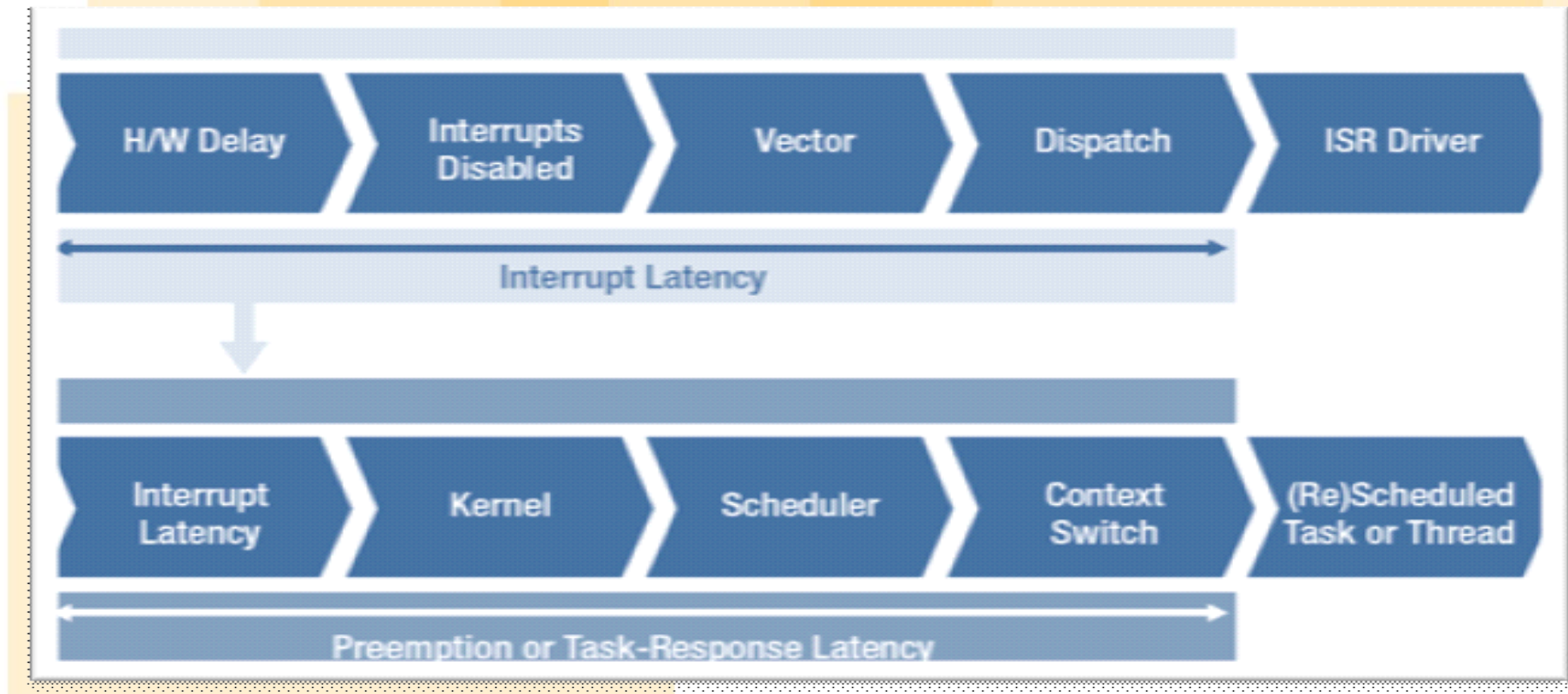
Hard Real-time Linux available solutions

- **RT-Linux**
- **RTAI**
- **Osware**
- **ART Linux**
- **KURT**

RT-Linux

- **A “hard real-time” mini operating system**
- **runs Linux as it’s lowest priority execution thread**
- **Linux thread completely preemptive**
- **Real time threads and interrupt handlers never delayed by non real-time operations**
- **Supports user level programming**
- **Mini RT-Linux implementation fits on a floppy**

Interrupt latency



Interrupt latency

- **RTOS and Linux kernels themselves disable interrupts in (hopefully) short critical regions when performing interrupt-related and/or non-reentrant operations. More frequently, interrupts are disabled for longer periods in device drivers.**
- **Independently published benchmarks for Linux Interrupt Latency on Intel Architecture reveals average latencies in the 10–15 microsecond range**
- **with driver-induced worst cases extending as long as 150–200 microsecond**
- **kernel-based worst-case Interrupt Latencies run between 40 and 80 microseconds.**

Options for low-latency response

- **Use the preemptive Linux kernel capabilities native to the 2.6 Linux kernel**
- **Other low-latency patches to the kernel from Ingo Molnar, MontaVista* and others**
- **Use of sub-kernels like RT-Linux* and RTAI***
- **Tuning the kernel and device drivers to support application-specific real-time needs**
- **Performing time-sensitive operations in the “top half” of drivers instead on in application code (appropriate for small numbers of interrupting devices)**

Shift to Linux – high level view

- _____
- _____
- _____
- _____
- _____

Working Environment

- **Optional working configurations:**
 - Linux servers with remote access
 - Linux PC per developer
 - Linux like environment for Windows
- **Setting up working environment:**
 - Linux environment with relevant services (samba servers, ftp,tftp,ssh,telnet,nfs)

Tool-Chain

- **Dedicated tool chain build for the chosen hardware platform.**
- **gcc and glibc based.**
- **Needed c & cpp support.**
- **ucLinux for mmu-less systems.**

Boot Loader

- **Boot loader build and configuration.**
- **Embedded environment free boot loaders. (U-Boot, Red-Boot, Blob...)**

Kernel

- **Building the Linux kernel.**
- **Network stack support (layers 2,3,4).**
- **File-system support (Ramdisk, NFS, flash file system).**
- **Various system services and kernel modules supporting the hardware.**

Linux Distribution

- **Building the system ram disk root file system.**
- **Requested services support (e.g. web server, ftp, tftp, ssh, telnet, nfs, ssl, ...).**
- **BusyBox support (many common UNIX utilities).**

Build System

- **Setting up GNU build environment.**
- **Recursive application makefiles & linkfiles.**
- **IDE configuration setup.**

Debug Tools

- **Linux debug and profiling tools.**
- **Valgrind suite for Linux application.**
- **Linux trace tool-kit support, strace,**
- **Exception handling.**
- **Post mortem debug (core dump).**
- **Memory monitoring tools.**
- **CLI infrastructure.**

Device Drivers

- **Linux conventions for kernel device drivers.**
- **External and internal kernel modules.**

Shell Scripts

- **Enabling bash & other scripting language (sed, perl, awk).**
- **Creating scripts for target boot and machine configuration.**

Links

- **Main kernel tree : <http://www.kernel.org/>**
- **Information source : <http://linuxdevices.com/>**
- **X86 open projects : <http://fedoraproject.org/>
<http://www.ubuntu.com/>**
- **Busybox utils : <http://www.busybox.net/>**
- **Boot Loaders : <http://www.denx.de/wiki/UBoot>
<http://sourceware.org/redboot/>**