Using Linux in Embedded Systems

Where and When does it apply?

21-Apr-2006

What is an "Embedded System"

- A combination of computer hardware and software, perhaps with additional mechanical or other components, designed to perform a dedicated function
  - Unlike a general-purpose PC, embedded systems typically have specific requirements and perform pre-defined tasks
  - The hardware and software typically is part of some larger system and expected to function without human intervention
  - Embedded systems often need to respond to events in real time
- Embedded systems can range in size from a single processing board to systems with full operating systems
  - Typical embedded system is a single-board microcomputer with software in ROM, which starts running a dedicated application as soon as power is turned on & stops only when power is turned off
- Using Linux as the operating system for embedded systems has captured the imagination of device designers

Constraints in Embedded System Design

- Embedded systems are designed to do a given task at a low cost
  - Most have specific real-time system constraints that must be met
  - They may need to be very fast for some functions, but not others
  - Often some parts of an embedded system need low performance compared to the primary mission of the system
  - Real-time constraints can be met with a combination of dedicated hardware and software tailored to the system requirements.
- Embedded systems are typically expected to run continuously for years without maintenance
  - More careful attention is paid during software development and test than is done for desktop PC applications
  - Embedded systems avoid mechanical or moving parts as much as possible because these are unreliable
  - Human intervention may be impossible so a system must be able to restart itself even if catastrophic data corruption occurs

Evolution in the Embedded World

- In the past, "embedded" referred to relatively small systems with few characteristics of a "computer"
  - Mostly hardware product with tiny amounts of memory and small but very efficient programs
  - May not need to communicate with other programmed devices
  - If there was an OS at all, it was lean and not very feature-rich
- Embedded projects today are more sophisticated, both in terms of features and underlying hardware
  - Most embedded systems require some type of connectivity, such as Ethernet or USB
  - Cheaper / faster processors & memory are available
  - Trying to write everything from scratch is not feasible so software reuse is more necessary than ever
  - Many of these components now exist in the public domain and in the open source community

Cost Considerations

- For high volume systems, cost usually dominates the system design
  - Use a CPU that is “good enough” for these secondary functions
  - Intentionally simplify system to keep costs as low as possible
  - Each component (including software) is selected and targeted to minimize system cost
- For low-volume systems, personal computers can often be used
  - A design choice might still be to limit the programs installed or replace the operating system with a real-time operating system
  - Special purpose hardware that is a requirement in a high volume system may be replaced by one or more high performance CPUs
- Some embedded systems may require both high performance CPUs, special hardware, and large memories to accomplish a required task.

Popular RTOS choices

- For embedded systems that use an operating system, we have more choices available today than ever before
  - There are more than we can list here, but to name a few:
    - Embedded Linux
    - VxWorks
    - QNX
    - Windows CE
    - Windows NT Embedded
    - LynxOS
    - NetBSD / OpenBSD
    - Palm OS
    - DOS
    - Inferno
    - eCos
- About 20% of embedded systems use Linux today & growing

Why Embedded Linux?

- Linux first attracts designers because it is free to download, it comes with the source, and is compatible with a wide range of processors
- A Linux operating system gives immediate access to many useful features that would be difficult to provide with a lower order OS
  - by choosing the right packages, it is possible to easily add such features as TCP/IP, SNMP, TFTP, HTTP, VoIP and others
  - there is a rich set of drivers and other features which add to the value package

Embedded Linux Advantages

- Choosing embedded Linux brings several advantages compared to other embedded operating systems:
  - Open source kernel provides mature well tested code base with a reputation for reliability
  - A modular architecture makes it possible to customize system to meet application requirements
  - No up-front purchase costs
  - No run-time royalty costs
  - A standard programming interface
    - Inexpensive development seats or zero cost development tools
  - A worldwide support community
  - Reasonably small footprint
    - Windows CE takes 21MB compared to 2MB for Embedded Linux

Embedded Linux Disadvantages

- Embedded Linux has a few disadvantages compared to other embedded operating system choices:
  - Has to be customized / optimized for specific application to obtain minimum footprint and maximum benefit
    - Designers must have certain knowledge or skills not widely available
  - Open source
    - No common platform to rally around – which distribution to choose?
  - Concerns about general-public-license contamination
  - Not generally a hard real-time solution

Memory Requirements

- A Linux kernel, combined with a few other free software utilities, can fit into the limited hardware space available in an embedded device
  - Linux micro-kernel requires only about 100 K on a Pentium CPU, including virtual memory and all core operating system functions
  - Add in the networking stack and basic utilities, a complete Linux system could still fit in 500 K of memory
- Total memory required is dictated by the applications selected for the given application, such as a Web server or SNMP agent
  - An embedded Linux system should be customized to user needs to minimize space requirements
  - A fully featured Linux kernel requires about 1 MB of memory
  - Typical installation of embedded Linux takes less than 2 MB

Linux is not a Real Time OS

- Before 2.6, the standard Linux application was not suitable for hard real-time systems for several basic architectural reasons
  - the basic Linux scheduler uses a fairness algorithm to guarantee even the lowest priority process some CPU time, even though a higher priority process may be waiting
  - when a process calls a kernel service, such as the scheduler or a device driver, this call disables interrupts and makes it impossible to pre-empt Linux until the service completes execution
  - Linux also relies on page swapping to move user code into and out of virtual memory, making timing unpredictable
- Standard Linux could still be used in embedded systems design, but just not if if the application has real-time response constraints
  - Something needed to be done ...

Adapting Linux for Real Time

- To use the 2.4 kernel in real time applications, the Linux community used patches and workarounds to deliver deterministic performance
  - One approach was to add pre-emption points within the kernel to reduce process latency but still protect critical code sections
  - A modified scheduler also ensures that the processor executes the highest priority tasks
  - Another approach was to add a small real-time kernel to handle the high-priority tasks while Linux runs as the lowest priority to schedule the remaining non-real-time tasks
    - RTAI (Real Time Application Interface) and RTLinux are two open-source projects based on this dual-kernel approach
- These patches are imperfect, but they offer a drastic improvement over the standard 2.4 Linux kernel
  - Main drawback is that they are nonstandard and require unique support and reimplementation for each kernel update

Linux as a Real Time OS

- Many real-time concepts applied to the 2.4 kernel are a permanent part of Linux in 2.6 and are standard build options
  - Numerous pre-emption points allowing the scheduler to suspend an active task and initiate a higher priority process were added
  - The kernel can be built without a page-swapping virtual-memory system that wreaks havoc on process timing
  - The process-scheduler algorithm has been re-written to speed task switching in multitasking applications
  - Improvements were made to the Linux Posix (Portable Operating System Interface for Unix) implementation
- 2.6 has updates to benefit embedded, desktop, and server systems
  - uClinux was incorporated into the kernel, allowing it to run on low cost micro-controllers without memory management
    - Yields a smaller footprint build for applications with no user interface

2.4 Kernel vs 2.6 Kernel

- So – with all of that, why not move to 2.6 kernel for embedded?
- The 2.6 kernel has the following disadvantages:
  - Slower to build: takes 30 - 40% longer to compile than 2.4
  - Slower to boot: takes 5-15% longer to boot to multi-user mode
  - Slower to run: context switches up to 96% slower, file system latencies up to 76% slower, local communication latencies up to 80% slower, local communication bandwidth less than 50% in some cases
  - Bigger FLASH memory footprint: the compressed kernel image is 30-40% bigger
  - Bigger RAM memory footprint: the kernel needs 30-40% more RAM
- For our current embedded developments we have chosen to remain with the patched 2.4 kernel rather than move to 2.6
  - Future projects might require 2.6 for other reasons (ex. IPv6)

When to choose Linux?

- Embedded developers need to ask what they want from an OS
  - First consider - does this application even need an OS?
    - Do you just need a round robin processing loop?
  - What processing hardware will be used?
    - Will it be a larger device with networking etc?
  - Do you need threading?
  - Do you need device drivers for chip peripherals?
- Linux is best suited to embedded applications that require access to higher order features (network features, SNMP, web server, etc)
  - Systems with multiple processes & interprocess communication or networking
  - Systems with mixture of real-time and non real-time requirements
- With modifications to the kernel Linux can be used for hard-real time systems

The Licence Issue

- Discussion of Linux is incomplete without considering licence issues
- Legal ambiguity surrounding the GPL potentially impedes choice of Linux for some embedded developments
- Embedded system design is especially susceptible to GPL concerns
- The GPL as clarified by Linus Torvalds. The Linux kernel is licensed under the GPL, but with this clarification:
  - “This copyright does *not* cover user programs that use kernel services by normal system calls -- this is merely considered normal use of the kernel, and does *not* fall under the heading of 'derived work.' ”
- This statement clarifies that user-space programs are not considered to be derived from Linux
  - It is also interpreted as allowing proprietary kernel modules

Practical Example

- We have built an embedded telecoms system with Ethernet, TFTP download, SNMP management interfaces and 3 layers of processor hierarchy
  - Based on a standard 2.4.24 kernel with RTAI patches added
  - Process switching measured at 15 microsecond (100 worst case) on Freescale MPC852 processor
- With static libraries, the load size still less than 4 MB as follows:
  - Kernel 0.5 MB
  - Selected GCC libraries ~1 MB
  - SNMPD 1.2 MB
  - TRAPD 1 MB

Conclusion

- Embedded Linux development is evolving rapidly
  - A designer has the option to choose from a variety of options for everything from the bootloader and distribution to the filesystem and GUI
  - Given this remarkable freedom of choice and a very active Linux community, embedded development on Linux has reached new levels of acceptance and usage
  - Many new handheld and embedded devices are being delivered as open boxes
- Hopefully this introductory overview of the embedded Linux space has whet your appetite and answered some of your questions
  - To aid you further in your projects, The Business Accelerators are able to help with more in-depth information on embedded Linux or help with your design efforts

The Business Accelerators

Creating Competitive Advantage

www.businessaccelerators.ca