Embedded Linux

Olivier Gruber

Professeur

Université Joseph Fourier

Projet SARDES

(INRIA et IMAG-LSR)

Embedded Linux

- Part One
  - Introduction to embedded systems
  - Fundamentals of a linux system
  - Virtual Machine Monitor technology
- Part Two
  - Minimal linux system
  - Boot and install process
  - Minimal kernel
  - OSGi platform

Embedded Systems

- What is an embedded system?
  - Dedicated software running in an industrial or consumer product
    - What matters is the complete product (hardware+software)
    - The software has no value by itself
  - Almost everything is board dependent
    - Sometimes there is not even an operating system
    - Everything is done by hand, hard-coded
    - The hardware choice drives everything else
      - E.g. satellite software
        - Weight and dimensions are imposed
      - E.g. low-end phones
        - 10 cents per phone for more memory
        - 10M euros for 100M phones
        - Total software of a phone could be about 1 euro

Embedded Systems

- Major characteristics of embedded systems
  - **Dedicated software**
    - A washing machine never brews coffee...
    - An MP3 player is not about playing other formats
  - **Reliable and secure**
    - Blue screens or segmentation faults are not an option
      - Airbus software must be reliable
      - ABS software must work when needed
    - Security must be a reality
      - Car electronics must be secure, not helping car thiefs
      - High-security buildings

Embedded Systems

- Major characteristics of embedded systems
  - **Maintainable**
    - Product lifespans are in tens of years
      - Suggests maintainable software
    - A typical car life is 10 years or more
      - 70% of car problems are electronic-related
      - Most are software bugs
        - Some only require to reset the overall system
        - Some require patching the embedded software
      - Sometimes it is a hardware failure
        - But hardware components have a shorter lifetime
        - No one wants to throw their car away because of this...
      - Suggests a modular approach
        - At the level of small embedded systems
        - A car could contain 4 networks and about 70 systems

Embedded Systems

- Major characteristics of embedded systems
  - **Optimized**
    - Hardware constraints are high
      - Small memory footprints
        - As low as a few kilobytes
        - A few mega-byte memory is huge
      - Slow processors
        - 8bit or 16bit processors are still around
          - Like an ARM7 on Atmel boards for instance
        - Slow 32-bit processors
          - Such as 33MHz 486 on PC-104 boards
          - Or 40MHz Dragonball (68K)
      - High-end PDA or smart phones
        - About 16 or 32 MB or even 64MB
        - Up to 400MHz 32-bit processors

Embedded Systems

- Major characteristics of embedded systems
  - **Why so optimized?**
    - A matter of price... because of large volumes
    - A matter of weight and dimensions
    - A matter of consumption
      - Despite power-efficient new processors
      - Battery life is the challenge of most mobile embedded systems
    - Examples
      - An early Palm had a battery life of 3-4 weeks
      - A PocketPC survived barely two days
      - Think of your digital camera...

Embedded Systems

- Major characteristics of embedded systems
  - **Specific**
    - All geared to the targeted board and the functionality
    - Form-factors are widely different
      - No GUI or just a small LCD
      - No mouse or keyboard, may be a touch screen
      - Sometimes no human interface at all
        - ABS in cars
        - Electric meters
    - Tools and environments are also specific
      - Compilers and debuggers are often specific to a board
      - Operating systems are also very diverse and not ubiquitous
      - Many different processors (ARMxx, PowerPC, x86, DSPs, others)
      - Overall development environment is crude and harsh

Embedded Systems

- A bit of philosophy...
  - **More powerful hardware... a chance or a curse?**
    - In fact both... we needed more powerful computers
  - But definitively **a cancer for software engineering**
    - Just developed so much bad habits for developers
      - Today, an empty library could be 300KB !
      - Early personal computers
        - ZX-81 with 1KB RAM, 8KB ROM, no disk, no floppy but a tape
        - Apple II with 48KB RAM, 16KB ROM, 5 ¼ floppy
      - Today's PCs
        - 256KB cache, 2GB RAM, 200GB of disk
        - 40M LOC for Windows
        - Typical software install over 100MB easy

Embedded Systems

- A bit of philosophy...
  - The problem is all across the board
    - Operating systems and compilers
    - Middleware frameworks, including verbose code generators
    - High-level language libraries (e.g. Java) and application developers
  - Just a comparison
    - Sun's Sparc (1987) 512KB RAM, 70MB hard drive
      - With Unix, gcc, gdb, X11, latex, emacs, etc...
      - With Smalltalk that was an entire environment
    - Eclipse 3.2
      - Just an IDE for Java over 130MB...
      - Plus a JRE of 16MB

Embedded Systems

- A bit of philosophy...
  - **Results in a profound schism**
    - Between traditional and embedded systems
  - Lack of skills
    - Only a few are still understanding low-level systems
  - Going back in history
    - Dedicated operating systems and tools
      - For small boards
    - Linux is too fat and complex for most embedded systems
      - But it can be tailored down a bit
        - We are still talking dozens of MB still
      - Suited for a whole range of hardware configuration
        - Routers, PDAs, smart phones, GPS, etc.

Embedded Systems

- Slowly evolving
  - Better modularity and reusability
    - Towards reusable components
    - Starting with operating systems
    - Continues with middleware systems
    - Talking about software components like OSGi
  - Not a reality today
    - Everything is done by hand
      - The hardware choice drives everything else
    - But software costs are unmastered
      - Assembly language is still used in many products
      - C is the default choice, sometimes C++
      - Java is trying to impose itself

Embedded Systems

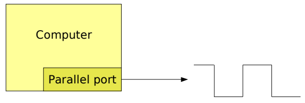
- Another schism: real time or not
  - Traditional operating systems are time-sharing systems
    - No real time constraints for correctness
    - Most often, scheduling favors overall throughput
  - Real-time systems
    - There is no single definition of what a real-time system is
    - But time constraints become part of the correctness definition

Embedded Systems

- Soft real time
  - Accuracy in time constraints is around 500ms
  - Examples
    - Video streaming where missing a few frames is acceptable
    - Most physical sensors such as wind, speed or temperature
- Hard real time
  - Each processing is defined with a time constraint
    - That **must** be respected
    - Under all loads
  - Hard real time usually also implies
    - Deterministic behavior with high availability and dependability
  - Examples
    - Embedded systems for cars, trains, planes, satellites or nuclear plants

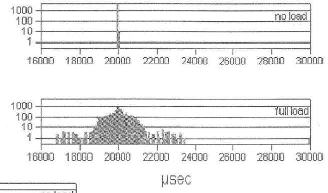
Embedded Systems

- Small experience
  - Signal generation at a given frequency
    - We use the PC parallel port and a small program
    - To generate a certain frequency (25Hz, so about a half-period of 20ms)
  - We compare on the same hardware
    - A standard linux
    - A real-time system

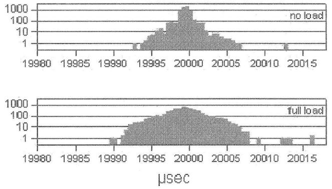


Embedded Systems

- Comparing soft and hard real time



Timings from standard linux are heavily influenced by load from **22Hz to 29Hz**
Most within +/- 1 seconds in full load and some within +/- 3 seconds



Timings from realtime linux are slightly influenced by load from **24.98Hz to 25.01Hz**
Most within +/- 10ms in full load.

Embedded Systems

- Discussion
  - Standard linux and other operating systems cannot be used for real-time
    - Some patches exist to improve the situation
    - But they are not making Linux a hard real-time system
  - Predictability versus performance
    - Predictability usually means less performance on equal hardware
      - A real-time system does not perform better
      - It is more predictable
    - Because some performance is lost to predictability

Embedded Systems

- Memory sizes
  - Typical sizes are given in the table
    - The number dates back to 2002 or so
    - Note:
      - Didn't change for the smaller embedded systems
      - Have doubled or so for the larger ones (up to 128MB)
  - The ROM is opposed to the RAM
    - But often most or all of the ROM is in reality some FLASH

Systems	Large	Medium	Typical	Deeply embedded
RAM	32-8 MB	8-2 MB	4-0.1 MB	Less than 0.1 MB
ROM	32-8 MB	8-2 MB	2-0.5 MB	0.5-0.1 MB
Processor	32bit	32-16bit	16-8bit	8bit

Embedded Systems

- System footprint
  - We must consider both static and dynamic footprint
    - Static footprint is the usual one we talk about
      - Easier to measure
        - Just look at the code sizes
      - Although pay attention to dynamically loaded code
        - Shared libraries
        - Kernel modules
    - Dynamic footprint is the real measure
      - Harder to measure
        - Need tools but also it may depend on the working set
      - But this is what need to fit in memory!

Embedded Systems

- Programming languages
  - Assembly language is still by far the language of choice
    - Enables to control both footprint and performance
    - But increased software development costs as software complexity grows
    - Lacks portability across the increasing families of processors
      - Strong ARM, SH3, PowerPC, Intel IA-32 and others
  - C language
    - Becoming the preferred language
      - If performance and footprint considerations allow to use C
      - Learn and use compiler options (like GCC -Ox for performance and -Os for size)
    - Heavy use of macros for adapting software
      - Adapting software is a major step in embedded system programming
        - Many constants are hardware dependent
        - Many low-level APIs are hardware dependents
      - Powerful macro languages are used to support the necessary software adaptation

Embedded Systems

- Programming languages
  - Considering Java
    - The portability of Java is extremely interesting to preserve software investments
    - But standard Java is not an option
      - JVMs are too fat
      - Libraries are poorly programmed and too fat
      - No hardware specifics are not reified
    - Sun's is promoting J2ME profiles
      - Smart card
      - CDLC and CDC
      - Foundation and personal
  - Several efforts exist
    - Savaje or Esmertec for example

Embedded Systems

- Programming languages
  - Industrial Software Technologies (IST)
    - A French technological leader in deeply embedded Java
      - Bare metal virtual machines as small as 32KB
        - Up to 64KB with a MIDP profile
      - From 8bit processors up to 32bit processors
        - Runs on many different processors and reifies many board specifics
    - Accelerator technology
      - IceTea language, a derivative of Java
      - Produces faster code than most platform C compilers!
    - Tailored virtual machines
      - For each hardware and each product line
      - Full control over the safety/speed/footprint challenge
    - Advanced debugging and testing environments
      - Hardware simulators with full assert modes
      - Discover 95% of problems, leaves only 5% on the actual board

Embedded Systems

- Operating systems
  - Evolution
    - Embedded systems used to be bare-metal standalone program
      - Simple functionality and complete control over the hardware
    - But this is no longer an option in many products
      - Software complexity is rising fast
      - Heterogeneity of hardware is also increasing rapidly
      - Multi-function embedded systems
    - All this suggests to use an operating system
      - Supports multi-programming
      - Virtualizes the hardware specifics through device drivers
      - Provide standardized Application Binary Interfaces (ABI)
      - Provides a basis for security and isolation

Embedded Systems

- Operating systems
  - Some examples
    - VxWorks from WindRiver (www.windriver.com)
      - The number one real-time OS in the embedded industry
      - WindRiver also acquired recently the pSOS real-time kernel
      - Technology
        - Provides strong network connectivity
        - TCP/IP stack natively integrated in the kernel
        - Comes with tools for cross-compiling and testing
      - Licenses are expensive
    - QNX (www.qnx.com)
      - Canadian Unix-like real-time OS
        - POSIX-compliant
        - Graphical interface is Photon, close to X Window System
        - Uses GNU tools
      - Can be used for free for non commercial applications

Embedded Systems

- Operating systems
  - Some examples
    - LynxOS
      - Another real-time POSIX-compliant operating system
      - Developed by LynuxWorks (www.lynuxworks.com)
    - µC/OS and µC/OS II (www.ucos-ii.com)
      - Created by Canadian Jean J. Labrosse for microcontrollers
      - Targeted originally the Motorola 68HC11 but it is now available for others
      - Provides a minimal TCP/IP stack (µC/IP)
      - Can be used for free for non commercial applications

Embedded Systems

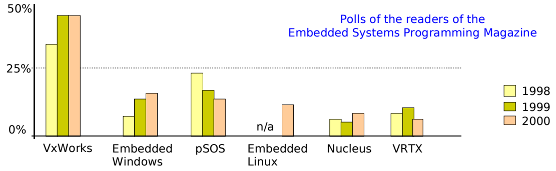
- Operating systems
  - More examples
    - Windows CE
      - US Microsoft targets embedded markets
        - About personal devices such as Smart phones or Personal Digital Assistants
        - A matter of long-term survival for Microsoft
      - Too fat and too slow for most embedded systems
        - Requires 400MHz processors and 64MB of RAM...
        - But attractive because of its compatibility and integration with desktops
      - Information at www.microsoft.com/windows/embedded

Embedded Systems

- Operating systems
  - More examples
    - Nucleus
      - Developed by Accelerated Technology Inc. (www.accelerated-technology.com)
      - Real-time operating system
        - TCP/IP connectivity
        - User interfaces through a Graphix library,
        - Web browser (WebBrowse)
        - HTTP server (WebServer)
      - Sources are provided and there is no royalties for redistribution
    - eCOS (Embeddable Configurable Operating System)
      - Initially from Cygnus, acquired by Red Hat Software
      - Real-time operating system for tiny memory footprints
      - Based on POSIX and GNU tools
      - Provides TCP/IP connectivity
      - Available under a license close to GNU GPL
      - Sources at http://ecos.sourceware.org

Embedded Systems

- Embedded Linux
  - Used to be not a player for embedded systems...
    - Just too fat, embedded devices too small and too slow
    - Embedded systems were about assembly language and dedicated software...
  - By 2000
    - Smaller Linux-based distributions are available
    - Medium to large embedded systems are powerful enough



Polls of the readers of the
Embedded Systems Programming Magazine

Embedded Systems

- Embedded Linux
  - Problems with proprietary operating systems
    - Often made by small companies
      - Difficult to keep up with hardware evolutions
        - A processor lifetime is about 12 to 24 months
      - Real risk of a embedded software company going under
        - Usually requires to by expensive source licenses
    - License costs are already too high
      - Profit margins are decreasing
      - Software must almost be free
    - Software development costs
      - Tool chains are specific and expensive
      - Difficult to find qualified people (not taught in Universities)
      - Trainings are expensive

Embedded Systems

- Embedded Linux
  - Open source advantages
    - Free to use and free is good!
    - Source availability
      - Both for tomorrow's bug and legacy processor support
      - But also for the ability to develop derivative work
    - A dynamic community
      - Finding help on line
        - Through FAQs, mailing lists, newsgroups, etc.
      - Software that is evolving rapidly
        - With respect to new processors or devices
        - With respect to fixing bugs
    - Licenses allowing commercial usages
      - GPL and LGPL licenses
      - MIT or BSD licenses
      - Apache or Eclipse licenses

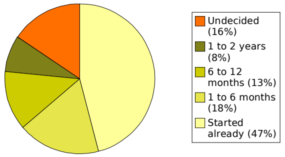
Embedded Systems

- Embedded Linux
  - Open source challenges
    - Diversity and heterogeneity of Linux distributions
      - Somewhat the same but still different enough to require an important ramp up
    - The licensing fears
      - Not quite a reality for building commercial systems
      - Even with the GPL, what was open source must remain so
      - But one can use Linux in a commercial routers or PDA
    - Lack of support and guarantees
      - Just a very different model for the embedded world
        - One has to believe in the community rather than on a contract
        - In reality, both work and both don't...
      - Many businesses are built around providing support for OSS

Embedded Linux

- So where do we stand with embedded Linux?
  - A few numbers from a study from Venture Development Corporation (VDC)
    - $28M in 2000, $55M in 2001, projected to be $305M in 2005
    - 59% of the embedded industry players had never used Linux in 2001
    - 19% had used it on one project and 22% used it on several projects
  - **Linux usage 2002** (source: CNET Networks Inc.)

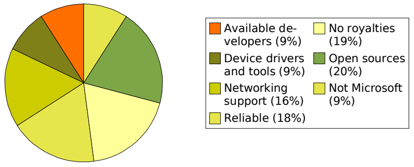
When do you plan to use Linux
for an embedded system project?



Embedded Linux

- Linux usage (source: CNET Networks Inc.)

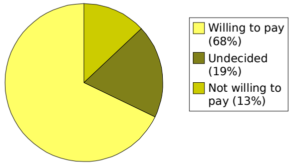
Why are you considering or using Linux?
(**2002**)



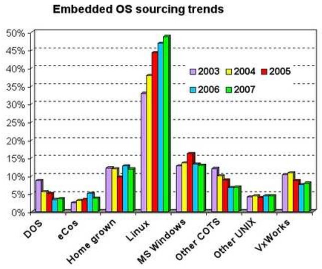
Embedded Linux

- Linux usage (source: CNET Networks Inc.)
  - All-free Open Source Software (OSS)
    - For a commercial product, paying for support is an important opportunity to ease Linux adoption and reduce development times

Would you be willing to pay for support?
(**2002**)



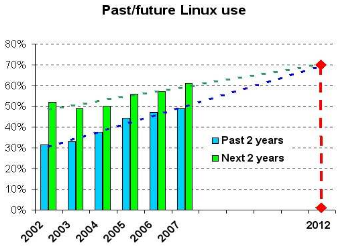
Embedded Linux



Which OSes have been in your (company's) embedded designs during the past two years?

Source: http://www.linuxdevices.com

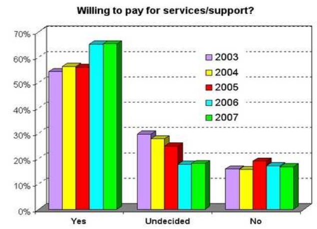
Embedded Linux



Actual and planned Linux use may converge by 2012

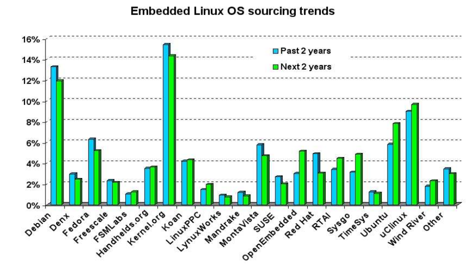
Source: http://www.linuxdevices.com

Embedded Linux



Source: http://www.linuxdevices.com

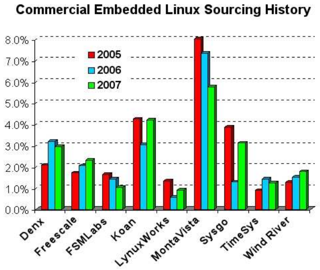
Embedded Linux



This suggests that Linux continues to deliver on the promise of vendor neutrality and absence of vendor lock-in, and that embedded Linux technology remains adequately decoupled from the fortunes or failings of any single company or organization.

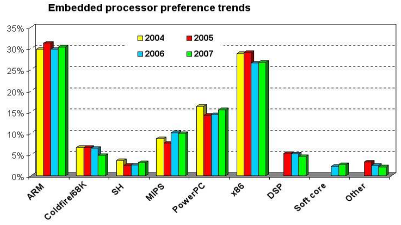
Source: http://www.linuxdevices.com

Embedded Linux



Source: http://www.linuxdevices.com

Embedded Linux



Source: http://www.linuxdevices.com

Embedded Systems

- Some embedded Linux distributions
  - MontaVista Linux
    - From http://www.mvista.com
    - Leader for commercial embedded Linux systems
    - Introduced the soft-real-time features in the Linux kernel 2.6
    - Supports a large number of processors
  - BlueCat Linux
    - From LynuxWorks (www.lynuxworks.com)
    - BlueCat version 5.0 is based on the Linux kernel 2.6

Embedded Systems

- Some embedded Linux distributions
  - µClinux
    - From http://www.uclinux.com
      - Pronounce (u-see-linux)
      - Targets processors with no MMU (Memory Management Unit)
    - Available on many processors
      - ColdFire Motorola, 68xxx, ARM, Intel i960, Axis ETRAX, etc.
      - Fast inclusion of new 2.6 kernel versions
    - A commercial support is available from Arcturus Networks

Embedded Systems

- Some embedded Linux distributions
  - RTAI Linux
    - From http://www.rtai.org
      - Adding a real-time kernel as loadable module
      - Linux is considered as the low-priority task
    - Was developed from RTLinux
      - But it is now independent and has an active community
  - EDLK
    - From a German corporation (http://www.denx.de)
      - Open source but no real time support
    - Excellent quality for cross-compiling to x86, PowerPC and ARM

Embedded Systems

- Some embedded Linux distributions
  - PeeWee Linux (http://www.peeweelinux.org)
    - Kernel 2.2 without real-time support
    - Easy-to-use tool to build the overall system image
      - Somewhat like make menuconfig for the kernel
    - Supports DiskOnChip Flash memories
  - Somewhat obsolete though

Embedded Systems

- Embedded Linux
  - When to not use Linux?
    - Target system does not need network connectivity or other existing drivers or protocols
    - Target system does not need to evolve (short lifetime for example)
    - Target system is too small
      - Minimal Linux kernel is at least 400KB compressed
      - Average Linux kernel is usually over 1MB compressed
      - Dynamic footprint is at least 4MB
    - GPL/LGPL is unacceptable to you, your boss or your specific needs
  - What to do then?
    - Look at using eCos or µC/OS
    - Bare metal tiny Java virtual machines like IST
    - Do your own development

Embedded Linux

- Case study
  - Build a minimal kernel for running an OSGi platform on a JVM
- Practical knowledges
  - Understand the Linux kernel boot process
  - Understand how to tailor a Linux kernel and a distribution
  - Understand how to install from scratch
  - Understand how to use Virtual Machine Monitors

Embedded Linux

- Pre-requisite
  - Be root on your machine
  - Virtual Machine Monitor
    - Download VirtualBox from www.virtualbox.org
  - Linux kernel sources
    - Download Linux kernel sources, suggested version 2.6.23.9
    - From http://www.kernel.org/
    - Or ftp://ftp.free.fr/mirrors/ftp.kernel.org/linux/kernel
  - Grub loader
    - Download Grub loader, version 0.97
    - From ftp://alpha.gnu.org/gnu/grub/