SIEMENS

The Internet of Things and Life beyond Linux

Embedded Linux Conference Europe 2016

Prof. Dr. Wolfgang Mauerer
Siemens AG, Corporate Research and Technologies
Smart Embedded Systems
Corporate Competence Centre Embedded Linux

Overview

Outline

SIEMENS

Introduction

Target Audience

- Linux is your standard development target
- Your embedded boards come with Linux/Android
- Open Source is default choice
- “Standard embedded engineer”

TA Check

- Linux *application* development?
- Embedded Linux *system* development?
- *Deeply embedded systems* development?

Introduction

Target Audience

- Linux is your standard development target
- Your embedded boards come with Linux/Android
- Open Source is default choice
- “Standard embedded engineer”

TA Check

- Linux *application* development?
- Embedded Linux *system* development?
- *Deeply embedded systems* development?

Introduction

Target Audience

- Linux is your standard development target
- Your embedded boards come with Linux/Android
- Open Source is default choice
- “Standard embedded engineer”

TA Check

- Linux *application* development?
- Embedded Linux *system* development?
- *Deeply embedded systems* development?

SIEMENS

Introduction

Target Audience

- Linux is your standard development target
- Your embedded boards come with Linux/Android
- Open Source is default choice
- “Standard embedded engineer”

TA Check

- Linux *application* development?
- Embedded Linux *system* development?
- *Deeply embedded systems* development?

SIEMENS

Internet of Things: What is it about?

Internet of Things

- Wireless sensor networks, home control
- Ubiquitous connectivity
  - Novel communication approaches (non-IP mesh networks)
  - Not covered in this presentation
- 2020: 25-30 billion devices
- Hardware costs *extremely* important

Hardware for IoT I

Infinite Ressources

- Supermarket class smartphone: 2GiB RAM, 2-4 cores
- Raspberry Pi: 1GiB RAM, 4 cores

Deeply Embedded: Cortex-M class

- NXP: $\approx$ 200 devices, TI: $\approx$ 400 devices
- On-Board memory, 100s of KiB
- Too large for bare metal programming, too small for Linux
- Available during the last 20 years
- Likely not going away any time soon

Hardware for IoT II

	Networked Node	Embedded Cntrl.	Embedded Comp.	Embedded Server
ARM offerings	M0/M0+/M3/M4	M4/7,A9,R4/5/7	ARM A9/A35,R7	ARM A53/A72
Intel offerings	Quark MCU	Quark SoC	Atom	Core, Xeon
Architecture, clock	32-bit, <500 MHz	32-bit, <1 GHz	32/64-bit, <2 GHz	64-bit, >2 GHz
non-volatile storage	MiBs	GiBs	GiBs	TiBs
RAM	< 8 MiB	< 1 GiB	< 4 GiB	> 4 GiB
HW ref. platform	Arduino class board	RPI class board	SoC-FPGA (Zync,...)	Industrial PC
application examples	Sensor, field device	control systems	special purpose & server based controllers	
	PLC, IoT node	gateways	multi-purpose controllers	

Shrinking Linux?

Network maintainer's point of view

What parts would you remove to get the foot print down for a 2MB single purpose machine?
I wouldn't use Linux, end of story.
Maybe two decades ago, but not now, those days are over.

(Response to net diet patch series)

SIEMENS

Alternatives

Linux Weltanschauung

“Linux has a lot more longevity and generality than most embedded OSes. Most such OSes are proprietary. All of them lack the range of capabilities, drivers, and general level of code quality and review found in Linux. Most have far smaller communities (or no communities at all).”

tiny.wiki.kernel.org/faq

SIEMENS

Weltanschauung and Veracity

- Many parts of Linux: Very high quality
- Tremendous complexity. Necessary?
- “Corner Cases” like real-time: Community?

Consequences I

SIEMENS

Perception

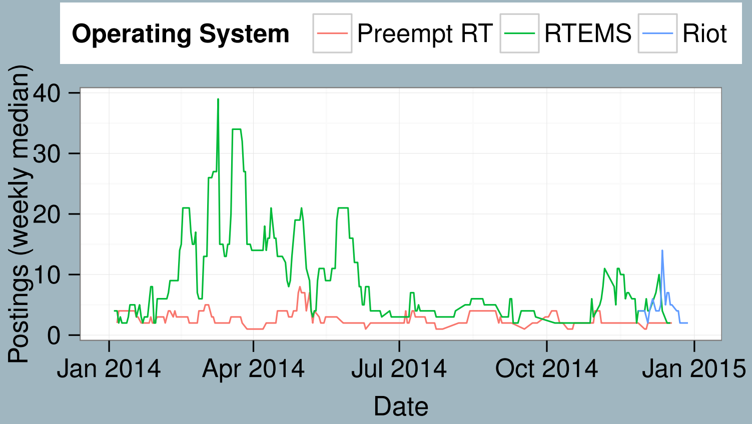
"A growing kernel makes it hard for the people who are trying to build tiny systems, forcing them to go to a proprietary real-time operating system instead." lwn.net

Alternative RTOSes

- RTEMS, eCos, Contiki, RIOT, mbed, FreeRTOS, uclinux, threadX. . .
- ⟨Favourite proprietary OS⟩

Consequences II

No communities at all?



- Biased, naturally: Specific feature vs. complete OS

Outline

SIEMENS

Δ(IoT, Linux): Address Spaces & Execution

Kernel+User//Syscalls//Threads vs. processes//Stacks

Δ(IoT, Linux): Scheduling

RT scheduling//determinism//scheduling choices//pre-computed schedules//time vs. event based

Δ(IoT, Linux): Building Appliances

App packaging//highly configurable kernels

SIEMENS

IoT vs. Linux: Legal Caveats

IoT vs. Linux: Legal Caveats

- Linux: Transition Kernel ⇔ Userland: *license barrier*
- IoT: Kernel + "Userland" in single address space
- Code (statically) linked together
- Some licenses: Implications on payload code, up to inheriting OS license!

IoT vs. Linux: Commonalities

IoT and Linux: Commonalities

- Toolchain (Cross Building), Build System, Version Control
- Debugging Mechanisms
- Standard C/C++ programming techniques
- Non-system libraries: Custom API
  - POSIX is overrated!
  - IoT system libraries: Yet another library. . .

Outline

SIEMENS

High-Level Comparison

System	POSIX	Maturity	VM	Archs	Drivers	Ressources	Docs
FreeRTOS	✗	high	✓	high	high	low	good
RTEMS	✓	very high	✗	high	very high	avg	very good
µclinux	✓	avg	✗	avg	high	avg	poor
mbed	✗	high	✗	low	low	avg	very good
Zephyr	✗	high	✗	low	avg	low	avg

RTEMS

RTEMS – Real-Time Executive for Multiprocessor Systems

- Extensive support for CPU architectures
- Comprehensive board support
- Commercial Vendors involved in development
- POSIX/Berkeley sockets support
- FreeBSD and LWIP networking stacks
- MPU, but no MMU support

Embedded With RTEMS www.rtems.org

- `www.rtems.org`
- $\approx 250 KiB$

FreeRTOS

FreeRTOS

- Virtual address space support
- Dynamic task creation
- Very small community
- Multiple TCP/IP stacks; Berkeley sockets API

- `www.freertos.org`
- A few KiB onwards

mbed

mbed

- Restricted to ARM targets (Cortex-M)
- Ease of use for (first-time) developers: Web-based dev environment
- Very easy deployment of applications (w/ suitable HW)
  - Upload binary to mass storage device
  - HW target support required (mbed HDK)
- High level C++ SDK

ARM® mbed™

- www.mbed.org
- ≈ 512 KiB

Zephyr

Zephyr

- Linux Foundation Community Project
- Build system and programming style very similar to Linux kernel
- Nano- and Microkernel with different APIs/capabilities
  - Nanokernel: Single task + ISR + fibres (non-preemptive; cooperative multitasking)
  - Microkernel: Multiple tasks (+ ISR + fibres); preemptive multitasking

ARM® mbed™

- www.zephyrproject.org
- A few KiB onwards

SIEMENS

Outline

SIEMENS

RTEMS example application I

```
#include <stdio.h>
#include <stdlib.h>

void *POSIX_Init(
  void *argument
)
{
  print("Hello, world");
  exit(0);
}

/* configuration information */

#include <bsp.h>

#define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER

#define CONFIGURE_POSIX_INIT_THREAD_TABLE
#define CONFIGURE_MAXIMUM_POSIX_THREADS 1

#define CONFIGURE_INIT
#include <rtems/confdefs.h>
```

Building RTEMS systems

Building a complete *system*

1. Download sources (links: see published slides)
2. Define target environment
3. Define build environment details

Technical Details

```
wget http://url/of/component.tar.bz2
```

Building RTEMS systems

Building a complete *system*

1. Download sources (links: see published slides)
2. Define target environment
3. Define build environment details

Technical Details

```
export ARCH=arm
export BSP=raspberrypi
```

Building RTEMS systems

Building a complete *system*

1. Download sources (links: see published slides)
2. Define target environment
3. Define build environment details

Technical Details

```
export ARCH=arm
export BSP=stm32f4
```

Building RTEMS systems

Building a complete *system*

1. Download sources (links: see published slides)
2. Define target environment
3. Define build environment details

Technical Details

```
export ARCH=i386
export BSP=pc386
```

SIEMENS

Building RTEMS systems

Building a complete *system*

1. Download sources (links: see published slides)
2. Define target environment
3. Define build environment details

Technical Details

```
export TARGET=${ARCH}-rtems4.11
export TOOLDIR=${HOME}/rtems-bin
export JOBS=8
export PATH=${TOOLDIR}/bin:${PATH}
export RTEMS_MAKEFILE_PATH=\
     ${TOOLDIR}/rtems/bsps/4.11/${ARCH}-rtems4.11/${BSP}
export RTEMS_ROOT=${TOOLDIR}/rtems/bsps/4.11/share/rtems4.11/
```

SIEMENS

Building RTEMS systems

Building a complete *system*

1. Build binutils
2. Build the initial C compiler/standard C library/final C compiler
3. Build the RTEMS kernel

Technical Details

```
mkdir binutils-build; cd binutils-build
../binutils-version/configure --target=${TARGET} \
                              --prefix=${TOOLDIR}
make -j${JOBS} && make install
```

Building RTEMS systems

Building a complete *system*

1. Build binutils
2. Build the initial C compiler/standard C library/final C compiler
3. Build the RTEMS kernel

Technical Details

```
mkdir gcc-build; cd gcc-build
../gcc-4.9.1/configure --target=${TARGET} --without-headers \
                       --with-gnu-as --with-newlib \
                       --enable-threads --prefix=${TOOLDIR} \
                       --enable-languages="c,c++"
make all-gcc -j${JOBS} && make install-gcc
```

Building RTEMS systems

Building a complete *system*

1. Build binutils
2. Build the initial C compiler/standard C library/final C compiler
3. Build the RTEMS kernel

Technical Details

```
mkdir newlib-build; cd newlib-build
../newlib/configure --target=${TARGET} --prefix=${TOOLDIR}
make -j${JOBS} && make install
```

Building RTEMS systems

Building a complete *system*

1. Build binutils
2. Build the initial C compiler/standard C library/final C compiler
3. Build the RTEMS kernel

Technical Details

```
cd gccbuild
../gcc-4.9.1/configure --target=${TARGET} --with-gnu-as \
                       --with-newlib --enable-threads \
                       --enable-languages="c,c++" \
                       --prefix=${TOOLDIR}
make -j${JOBS} && make install
```

Building RTEMS systems

Building a complete *system*

1. Build binutils
2. Build the initial C compiler/standard C library/final C compiler
3. Build the RTEMS kernel

Technical Details

```
cd rtems; ./bootstrap
cd ..; mkdir rtems-build; cd rtems-build
../rtems/configure --target=${ARCH}-rtems4.11 \
                   --enable-rtemsbsp=${BSP} \
                   --enable-networking --enable-posix \
                   --prefix=${TOOLDIR}/rtems/bsps/4.11
make && make install
```

SIEMENS

Building the payload application

Building the payload application

- Collect application sources (libraries: separate build, of course)
- Makefile templates: `rtems/make/Template`
- Build process delivers binaries in `o-optimized/`
  - `file.exe`: ELF executable. Symbol information, sections etc.
  - `file.ralf`: *RTEMS Application Loadable File.* “Core dump” of the binary. ($\approx$ `objdump -O binary`) – execute on *raw* hardware (debugger, flash tool, bootloader)

Running pc386 inside qemu

System in emulated machine

```
qemu-system-i386 -no-reboot -serial stdio -monitor null \
                 -nographic -m 2 \
                 -append "--console=com1" \
                 -s -kernel file.exe
```

SIEMENS

Running pc386 inside qemu

System in emulated machine + debugging

```
qemu-system-i386 -no-reboot -serial stdio -monitor null \
                 -nographic -m 2 \
                 -append "--console=com1" \
                 -s -kernel file.exe
```

- `gdb file.elf`
- `(gdb) target remote localhost:1234`
- ...as easy as debugging a simple Linux userland application

SIEMENS

Running pc386 inside qemu

System in emulated machine + debugging + networking

```
qemu-system-i386 -no-reboot -serial stdio -monitor null \
                 -nographic -m 4 \
                 -append "--console=com1 --ne2k-irq=9" \
                 -device ne2k_isa,netdev=usernet \
                 -netdev user,id=usernet \
                 -redir tcp:24742::24742 \
                 -s -kernel file.exe
```

Networked appliance with dynamic language

Requirements

- Payload: No deeply embedded experts required
  - Port lua to device (essentially: adapt `Makefile`)
  - Run standard C applications (RT etc.) in parallel
- Standard networking (configuration option!)
- Linux-like interactive development
- Works well with $\approx$ 0.5 MiB of RAM
- See link in published slides

Thanks for your interest!