

Real-Time Applications with Xenomai

Real Time Operating Systems and Middleware

Luca Abeni

luca.abeni@unitn.it

Xenomai

- Xenomai: real-time OS based on Linux
 - Before Xenomai 3: dual kernel system
 - Xenomai 3 provides 2 options
 - `cobalt` → dual kernel systems, as in previous versions
 - `mercury` → real-time in Linux user space (using Preempt-RT)
- *Skins* providing different kinds of real-time APIs

Dual-Kernel Approach

- Xenomai can use a dual kernel approach (before Xenomai 3, or using `cobalt`)
 - Based on the I-Pipe concept
- Real-Time applications: Linux kernel modules
- Can use one of the APIs provided by Xenomai
- Cannot use the standard C library, etc...
- Need Linux kernel headers to be compiled
- Started by inserting a module in the kernel

Example of Xenomai Application

- Focus on dual kernel
- POSIX API (use the POSIX skin provided by Xenomai)
- Old Kernel and Xenomai versions
 - Why? To save disk space! And time...
 - Can be easily repeated with more modern kernel and Xenomai versions
- Simple example: periodic threads
- Based on the ARM architecture

Example - Goals

- Learn how to compile and use kernel modules
- See a dual kernel system in action
- Cross-compile kernel code for ARM
- Familiarize with a real-time system used in practice

Before Starting

- The ARM Cross-Compiler will be used
 - Uncompress it somewhere, and adjust the `PATH`
- The example application needs some Kernel Headers to build
- Then, some binary files (I-Pipe modified kernel, Xenomai modules, ...) are needed to execute the test
- As usual, we test with QEMU (`qemu-system-arm`)
- Find all the stuff here:

`http://disi.unitn.it/~abeni/RTOS/Xenomai`

Building the Xenomai Test - 1

- Building a Linux Kernel module...

- Proper Makefile

```
1 EXTRA_CFLAGS=-Iinclude/xenomai/posix -Iinclude/xenomai
2 obj-m=posix-test.o
3 posix-test-objs=periodic_tasks.o periodic-thread.o task_bodies.o
```

- `posix-test` is the name of the application
- `posix-test-objs=` specifies the compilation units composing it
- Uses the **kbuild** scripts from Linux kernel!
 - Again: need Linux sources to build (uncompress in `/tmp/l-head`)

Building the Xenomai Test - 2

- `make -C /tmp/l-head M=$(pwd) ARCH=arm
CROSS_COMPILE=arm-unknown-linux-gnu-`
- “`-C /tmp/l-head`”: tell make where kbuild and the Linux headers are
- “`M=$(pwd)`”: tell kbuild where the module to be compiled is (`pwd`: current working directory)
- “`ARCH=arm
CROSS_COMPILE=arm-unknown-linux-gnu-`”:
cross-compilation stuff
- A file named “`posix-test.ko`” is created

Running the Xenomai Application - 1

- Cross-compiled for ARM → test in a VM
 - Build a ramfs image
(`initramfs-scripts/mkfs.sh`)
 - Copy `posix-test.ko` in it
 - Copy the modules from
`binaries-xeno.tar.bz2`
(`Xeno/lib/modules`) in it
- In the VM, insert the modules!
 - From
`/lib/modules/2.6.19.7/kernel/kernel/xenomai/...`

Running the Xenomai Application - 2

- Inserting a Linux module: `insmod` or `modprobe`
 - Using `modprobe` in the test VM is complex
 - `insmod` → full path to modules
- Modules to be inserted: Xenomai, POSIX skin and application
 - `insmod nucleus/xeno_nucleus.ko`
 - `insmod skins/posix/xeno_posix.ko`
 - `insmod /posix-test.ko`