

Xenomai 3 installation notes



Paulo Pedreiras
pbrp@ua.pt

DETI/IT/University of Aveiro

Real-Time Operating Systems Course
Oct/2021

Xenomai 3

- Xenomai 3 has a new architecture
- Can run:
 - Cobalt: Seamlessly side-by-side with Linux as a co-kernel system
 - Best performance, but
 - Implies patching and compiling the kernel
 - Mercury: Natively over mainline Linux kernels.
 - The mainline kernel should be patched with the PREEMPT-RT patch to improve timeliness
 - Poorer performance
 - Smaller installation burden/complexity
 - Good for taking a “taste” of Xenomai or for applications with large periods/jitter tolerance/...

Installation

- Xenomai follows a split source model, decoupling the kernel space support from the user-space libraries.
 - Kernel components available under the kernel/ sub-tree
 - User-space components under lib/ sub-tree.
 - Other top-level directories, such as scripts/, testsuite/ and utils/, provide additional scripts and programs to be used on either the build host, or the runtime target.
- The kernel/ sub-tree which implements the in-kernel support code is seen as a built-in extension of the Linux kernel. Therefore, the standard Linux kernel configuration process should be used to define the various settings for the Xenomai kernel components.
- The lib/ sub-tree contains the various user-space libraries exported by the Xenomai framework to the applications.
 - This tree is built separately from the kernel support.
 - Libraries are built in order to support the selected core, either Cobalt or Mercury.

Installation - Mercury

- Does not require compiling a kernel
- Kernel should provide high resolution timers support (CONFIG_HIGH_RES_TIMERS)
- PREEMPT_RT advisable for shorter latency/jitter

Installation - Mercury

- Download tarbal
- Extract tarbal to path **[PATH_SOURCE]**
 - `tar xvjf xenomai-3.1.tar.bz2`
- Create a build folder, configure and install (Mercury libraries natively for a x86_64/SMP system, enabling shared multi-processing support)
 - `mkdir xenomaibuild`
 - `cd xenomaibuild/`
 - `[PATH_SOURCE]/xenomai-3.1/configure --with-core=mercury --enable-smp --enable-pshared`
 - `sudo make install`
- After this step the installation root should be populated with the libraries, programs and header files needed to build Xenomai-based real-time applications. The default directory path is `/usr/xenomai`.
- Note: be careful with the folder names. Certain characters can cause problems. Keep the path simple. E.g. `/home/USER_NAME/xenomaibuild/`

Installation - Mercure

- *Time to test the installation*
 - `$cd usr/xenomai/demo`
 - `$/altency` or
 - `$sudo ./cyclictest`
- *If the test programs work properly everything is set*
- *Those programs provide good indications about the RT performance of the platform.*

Xenomai application development

- PATH and LD_LIBRARY_PATH must be set properly.
- If the applications fail to compile/execute do:
 - `$export PATH=$PATH:/usr/xenomai/bin/`
 - `$export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/xenomai/lib/`
- For a permanent solution configure the “.profile” or “.bash_profile” (individual users) or “/etc/environment” (system-wide) files accordingly.

Note: the procedure herein reported was tested with Ubuntu 20.04 LTS and Xenomai 3.1

Info Sources

- Overview and Architecture
 - https://gitlab.denx.de/Xenomai/xenomai/-/wikis/Introducing_Xenomai_3
 - <https://elinux.org/images/7/76/Kiszka.pdf>
- Installation
 - <https://xenomai.org/documentation/xenomai-3/html/README.INSTALL/>
- Application development
 - <https://xenomai.org/documentation/xenomai-3/html/xeno3prm/index.html>
 - https://gitlab.denx.de/Xenomai/xenomai/-/wikis/Building_Applications_For_Xenomai_3
 - <https://www.ashwinnarayan.com/post/xenomai-realtime-programming/>
 - <https://www.ashwinnarayan.com/post/xenomai-realtime-programming-part-2/>
- Other
 - <https://www.diva-portal.org/smash/get/diva2:1251188/FULLTEXT01.pdf>