Philippe Gerum, Paolo Mantegazza, Bruno Rouchouse, Robert Schwebel

The RTAI 3.0 Reference Guide

Contents

Part I

Introduction

1 Preface

1.1 About this Guide

RTAI has become one of the most sophisticated frameworks for Hard Realtime (HRT) on Linux. But what's the best way to use it for *your* application? Until now there was no "unified" documentation for RTAI which could be used as a tutorial to learn how RTAI works *and* as a reference for all the different sub packets and functions the framework provides.

The *RTAI 3.0 Reference Guide* tries to solve this problem. It is an attempt to unify all the existing documentation of the RTAI project, which, until now, existed in several places like the sourcecode, the web site, some READMEs and other locations.

1.2 RTAI History

(FIXME: add earlier RTAI history here)

During all this time the policy of the maintainers was to add almost every piece of contributed code to the RTAI packet, because the idea was that if something was useful for somebody it could be useful for somebody else as well. RTAI was organized after the Bazaar development model described by Eric Raymond [**?**]. Although this model worked quite fine for several years it turned out over the time that this model resulted in lots of features existing in several, incompatible implementations which where *almost* identical but not really. And, after all, the "organic grow" of the code resulted in large pieces of code which were less and less maintainable and could not easily be separated into independend modules. It turned out that in the future, user space hard realtime would more and more replace the traditional design of HRT in kernel modules in combination with userland tools for the handling of non-realtime tasks. This trend was followed with with the LXRT, LXRT-Informed and New-LXRT implementations of userspace realtime, but it was clear that the future could only be based on a better structured code base.

On the Hardware Abstraction Layer, the lowlevel interface between the hardware and the basic interrupt mechanisms of RTAI, "RTHAL" was used for most of the time. The RTHAL mechanism was basically the same technique which was used by RT-Linux: a deferred interrupt scheme. Although the mechanism worked quite fine it was a target of large and unfruitful discussions, triggered by the infamouse RT-Linux patent, filed by Victor Yodaiken in 1997 [**?**]. To end these contraproductive discussions and to base RTAI on a more flexible technology Karim Yaghmour published an article about "ADEOS - Adaptive Domains for Operating Systems" in 2000 (? FIXME) in which he described a technology which could be used for more than hard realtime. Using a technology which is used by *more* people is generally a Good Thing (TM), because the more people use a piece of code the better it's quality would be. As ADEOS could also be used to make several instances of Linux run on a machine at the same time, or to make glueless kernel debuggers, it was considered to be a candidate for the hardware abstraction layer of the Next Generation RTAI. And, last but not least, as the technology used by ADEOS is known and published for quite a long term this would finally end the uggly patent discussion with all it's fear, uncertaincy and doubt.

In 2002 the RTAI team started to think about a large redesign to make RTAI fit for the "Next Generation" of realtime software. It all started with Philippe Gerum implementing the ADEOS technology described by Karim Yaghmour. The next move was that the existing code base of RTAI was split into several logical units which could be maintained as independend packets. Until now, people had to install the whole RTAI packet, no matter if they needed everything or not. It was difficult to find out how things worked together and it was difficult for people to find their way into the RTAI development team. All this should become better

and easier with the next major release. Philippe Gerum started to reorganize the RTAI tree with help of the development team. At the end of 2003 the first release of the ”new” RTAI was released.

1.3 Participating in the RTAI Team

RTAI has always been a community effort, and this will not change with the ”new” RTAI. Although we try to put a better ”structural design” behind the software the team still follows Paolo Mangegazza's idea of a community effort. Things that are useful for somebody will still find it's way into the package, although the maintainers try to make the ”big plan” behind RTAI more visible to the outside.

It is most important that people do not only *work* with RTAI but *actively participate* in the RTAI development team. It is easier than ever to do this today: the ”RTAI-Dev” mailing list was opened up for the general public so that everybody can see what the developers are doing and how RTAI continues evolving. Who is interested in joining the team can simply subscribe the RTAI-Dev mailing list and help making RTAI+Linux the best realtime operating system which ever existed. One of the most important properties of the RTAI team is that it is a team – people try to achieve a goal by working together. It always has been fun for the team members to develop RTAI in a cooperative and constructive way, and we hope that this will not change in the near future.

People who want to participate in the RTAI development should first subscribe the mailing lists. Further details can be found on the website.

2 Introduction

2.1 What is Realtime?

2.2 The RTAI Concept

2.2.1 Technology

2.2.2 Notes on the RTAI Release Numbering Scheme

2.2.3 Architecture

2.2.4 RTAI's Services

2.3 Downloading RTAI

2.3.1 Downloading Official Releases

2.3.2 Accessing CVS

2.4 Configuring and Building RTAI

2.4.1 The Configuration System

2.4.2 Making

2.5 The Makefile System

2.5.1 Making Extensions for RTAI

2.5.2 Writing Separate Makefiles

3 Development Fundamentals

3.1 Kernel Module Basics

3.1.1 Building a Kernel Module

3.1.2 Exporting Symbols

3.1.3 Passing Parameters to Modules

Part II

The RTAI Core

Part III

RTAI Services

4 Interprocess Communication

4.1 FIFOs

4.1.1 Overview

In a well designed RTAI application you usually have a clear separation between time critical and non-critical parts of the code. This means that you have to transport data from realtime land to userland and vice versa.

RTAI offers several interprocess communication mechanisms for this task. One of the easiest method is the use of FIFOs. After a FIFO was created you can simply ”put” some data into it from one environment (e. g. from userland) and read it later – in the same order – from the other envirionment (e. g. from kernel space).

4.1.2 FIFO creation

To create a FIFO you can use this function:

```
rtf_create (unsigned int minor, int size)
```

NAME

rtf_create (unsigned int minor, int size)

minor: & minor device number of the fifo size: & buffer size for fifo data; this can be resized later with rtf_resize()

RETURN VALUE

0 on success -ENODEV illegal fifo minor number -ENOMEM out of memory

4.1.3 Handlers

4.1.4 Signal Interface

4.1.5 Select/Poll and Blocking

4.1.6 Named FIFOs

4.1.7 API Reference

4.1.8 Implementation

4.2 Semaphores

4.2.1 Overview

4.2.2 API Reference

4.3 Shared Memory

4.3.1 Overview

4.3.2 Implementation

4.3.3 Build and Insert Shared Memory

4.3.4 Using Shared Memory in User Space

4.3.5 Using Shared Memory in Kernel Space

4.3.6 Proc

4.3.7 API Reference

4.4 Mailboxes

4.4.1 Overview

4.4.2 Implementation

4.4.3 API Reference

4.5 Extended Messages

4.5.1 Overview

4.5.2 API Reference

4.6 Remote Procedure Call

4.6.1 Overview

4.6.2 API Reference

5 Distributed Interprocess Communication

5.1 Overview

5.2 API Reference

6 Posix - RTAI/fusion

6.1 The New POSIX Approach

In older RTAI versions POSIX compatiblity has been achieved by writing several modules which implemented the standard POSIX APIs in RTAI modules. This approach has turned out to be problematic and difficult to maintain, so a new scheme has been invented.

Instead of rewriting a complete POSIX layer usable in a real-time context, a way has been found to move the existing user-space Linux support seamlessly into the RTAI realm. This basically allows to call regular Linux syscalls synchronously from RTAI tasks running in user-space, while keeping the scheduling priority of the caller unaffected by the domain migration.

This approach offers a number of advantages:

- Conflicts (source-wise and execution-wise) between the RTAI-specific and native POSIX interfaces (Linuxthreads/NPTL) cannot occur, since there is no need for any RTAI-specific interface in the first place.
- Since user-space thread interfaces use kernel syscalls to perform thread synchronization and other management, interface upgrades should have very little impact on the RTAI POSIX support. Conversely, RTAI should benefit from POSIX interface upgrades at low or even no cost.
- All user-space services are now automatically RTAI-enabled, i.e. real-time compatible, provided they are thread-safe in the first place. Obviously, this does not mean that Linux services miraculously become time-bounded with this extension, but at least, they are now callable from any RTAI task context. It is still up to the user to evaluate the suitability of calling such code in a real-time context though.

6.1.1 How it works

LXRT is able to migrate user-space tasks back and forth between the Linux and RTAI domains, depending on the requested operating mode.

Hard real-time mode is obtained by stealing the current task from the Linux's scheduler runqueue and readying it for the LXRT scheduler. Since the latter also reinstates the MMU-context of any incoming task, the net result is that we have memory protected tasks solely scheduled by RTAI. After this transition, the migrated/stolen task left in `TASK_UNINTERRUPTIBLE` state disappears from Linux's radar, just like any suspended task, assuming that some code somewhere will care for resuming it when it's due. At this point, the RTAI scheduler can pick this task and reinstate its MMU-context when it switches it back in.

Migration from Linux to RTAI needs a helper task controlled by Linux (namely, `kthread_b`) which is resumed by RTAI, and whose job is to move any given real-time task to RTAI's ready task queue, then trigger a fast real-time rescheduling to make such task resume into the RTAI domain immediately. The action of waking up this helper task after the migrating task has been suspended (state == `TASK_UNINTERRUPTIBLE`) conveniently ensures that the Linux runqueue correctly reflects the new situation.

The following pseudo-code example shows how it works:

```
T1 runs in soft real-time mode...
        T1 calls make_hard_realtime()
                T1 Linux state is set TASK_UNINTERRUPTIBLE
```

```
                kthread_b() kernel thread is woken up
kthread_b() runs... (T1 is now out of Linux's runqueue)
        kthread_b() fast schedules back T1
T1 immediately resumes in hard real-time mode...
```

Migration from RTAI to Linux is somewhat simpler, only requiring a RTAI service request to be scheduled so that `wake_up_process()` is called on the migrating task as soon as the Linux kernel gets back in control.

6.1.2 Side-effect of a soft-mode transition in LXRT

Soft-mode transition is caused by an explicit call to `make_soft_realtime()`, or implicitely done by LXRT when a Linux syscall is caught from a hard real-time task. LXRT also makes sure that such task goes back to hard real-time as soon as the Linux syscall has completed.

The first effect of the soft-mode transition is to lower the calling task's priority down to the Linux placeholder task (i.e. `rt_linux_task`), representing the idle – non real-time fallback context. This is an accepted side-effect that such lowering also affects the real-time priority handling, de facto excluding the soften task from the real-time realm.

6.2 Issues and solutions

The following issues are raised when it comes to integrate the Linux General Purpose Operating System (GPOS) kernel within the execution domain of a RTOS like RTAI:

- Linux internal design promotes fairness and throughput, at the expense of determinism. The net effect is that many code paths in the vanilla 2.4 Linux kernel are not preemptible for serving interrupts and/or rescheduling tasks.

By increasing the overall preemptibility of the Linux kernel, Montavista's "kpreempt" extension partly solves this issue, by limiting the non-preemptible sections to those protected by a spinlock, and calling the rescheduling procedure on exit of such sections if needed. Coupling Andrew Morton's low latency patch further reduces the length of such non-preemptible sections.

- Fine grained Linux kernel preemptibility does not help prioritizing the interrupt load from a real-time point of view, which means that low-priority bottom-halves can still preempt high-priority real-time Linux tasks (SCHED_FIFO) under this model, thus introducing unbounded latencies.

By implementing a technique called "interrupt shielding" using ADEOS capabilities, we can control the interrupt flow so that Linux IRQ handlers are not fired while real-time RTAI tasks tread on Linux kernel code. This is obtained by inserting an intermediate ADEOS domain between RTAI and Linux, and stalling/unstalling this stage at the proper time. This way, whatever Linux actions are regarding its own interrupt state, ADEOS guarantees that the effective interrupt control for the GPOS kernel is left in the hands of the shielding domain above Linux in the pipeline.

```
i.e. IRQ -> [RTAI] -->-- [Shield] -->-- [Linux] (std mode)

     IRQ -> [RTAI] -->-- [Shield] --/   [Linux] (rt mode)
```

- Interrupt shielding has one drawback: RTAI tasks which end up suspending themselves on a Linux kernel resource waiting for a Linux interrupt to occur (e.g. I/O or timing wait) will be woken up only after the shield is deactivated, i.e. when no other RTAI task is running into the Linux kernel domain. This fact is a source of priority inversions, since low priority RTAI tasks could prevent hi-priority ones to wake up until they relinquish the processor. Additionally, Linux's periodic timer resolution is not suitable for hi-frequency real-time processing.

Timing syscalls (`gettimer()/setitimer()` and `nanosleep()`) used by the POSIX layer are transparently intercepted by RTAI/fusion, and converted to their RTAI-based hard real-time counterparts. This conversion has two positive effects: RTAI's timer resolution is much higher, and timed RTAI tasks are suspended into the RTAI domain, thus cannot be shielded from timer interrupts by the intermediate domain.

However, RTAI tasks waiting on I/O events only handled by the Linux kernel are still subject to priority inversions. For now, the best way to prevent this undesirable side-effect is to have those tasks handled by RTAI I/O drivers instead of relying on Linux drivers, since wakeups in the RTAI domain are performed immediately by the RTAI interrupt handler.

- RTAI currently provides determinism *aside* of Linux by adding a preemptive and concurrent kernel. Services requests from RTAI to Linux then cause the requested services to run outside the real-time execution domain. RTAI tasks can only send asynchronous requests (i.e. srqs) to perform Linux services which will be processed when no real-time tasks are runnable. Moreover, the service inherits the scheduling priority of the last Linux task which happened to have been preempted by RTAI. This situation currently prevents the Linux kernel code to be part of the real-time execution domain.

By tracking the priority of the RTAI task running the Linux kernel code and applying it dynamically to the Linux placeholder task controlled by the RTAI scheduler (i.e. rt_linux_task), the position of the RTAI task within the real-time priority hierarchy is kept unchanged, even when it executes into the Linux domain.

Additionally, the priority of the Linux task coupled to the real-time RTAI task is set to a value preserving the overall RTAI priority scheme. This way, we have two schedulers (i.e. RTAI and Linux) cooperating for the execution of real-time tasks inside a single priority scheme across both execution domains.

Example:

```
RTAI domain   -- RTAI (pseudo-)task "T0" scheduled by rtai_sched
        (pri = 10)

Linux domain  -- Linux task "foo" (mapped to T0) scheduled by Linux
        (pri = sched_getpriority_max(SCHED_FIFO) + 1 + pri(T0))
```

- When migrating tasks from the RTAI execution domain to the Linux domain, a time-critical transition must occur, during which RTAI delegates the control of this task to the Linux scheduler. This phase must be fast and perfectly synchronized, to guarantee a safe reentrancy into the Linux kernel code. Additionally, since the only safe transition point is the Linux rescheduling procedure, the delay incurred to reach the next rescheduling point inside any preemptible portion of the Linux kernel must be bounded and as short as possible.

Transitions from RTAI to Linux are obtained by sending a virtual ADEOS interrupt from the RTAI domain to the Linux domain. The interrupt handler into the Linux kernel wakes up the process as requested.

If the Linux kernel code was last preempted by RTAI outside of a critical section, a Linux rescheduling takes place immediately on the return path of the virtual interrupt. Otherwise, the preemptible kernel will ensure that a rescheduling takes place as soon as the critical section is exited. With the additional benefit of the low latency patch, those sections are kept as short as possible, enforcing lock-breaking preemption points as needed.

6.3 Practical Assumptions, Limits and Guarantees

The new scheme has some practical assumptions and limits:

- A vast majority of RTAI-based applications in user-space would rely on POSIX services mainly for thread synchronization and hi-resolution timing. Thread synchronization services would only pay the additional cost of RTAI/Linux migrations - needed to re-enter the Linux kernel from a RTAI task for

performing the system call - when the access to some resource is effectively contended by more than a single POSIX thread. Provided this cost is at most a handful of microseconds, it is assumed that the advantages of RTAI-enabling the Linux system calls instead of rewriting specific APIs outweight this single drawback, especially since most of the Linux syscalls issued for synchronization by the native POSIX layer tend to put the caller to sleep. In any case, very time-critical services could still be substituted with a mere RTAI add-on if needed.

- "In the POSIX interface we trust". IOW, it is assumed that the standard POSIX interface used to program the real-time applications is efficient as far as thread concurrency is concerned. As a consequence of this, and aside of the kernel time consumed in executing syscalls for its threads, rewriting another dedicated RTAI-specific interface should not be significantly more efficient performance-wise.
- "In the Linux kernel we trust". IOW, the syscalls internally used by the standard POSIX interface for common operations are assumed to be efficiently written and would not charge unacceptable CPU cost to any application which currently targets user-space real-time support. The preemptible and low latency kernel patches are expected to provide for limited and bounded _scheduling_ latencies inside the Linux kernel.
- Running a RTAI task into the Linux domain with an effective real-time priority must not impact on the preemptability inside the RTAI domain. The following assertions remain true at any point in time:
  - Adeos's pipelined interrupt model ensures that any unmasked interrupt is immediately dispatched to the RTAI domain if unstalled, regardless of the currently running domain (RTAI, shielding domain or Linux).
  - Interrupt shielding ensures that Linux targeted interrupts are not propagated to the Linux handlers when a real-time task executes into the Linux kernel.
- The Linux kernel provides for a `SCHED_FIFO` scheduling policy enforcing static real-time priorities among the tasks it controls. This is especially important as Linux's scheduling decisions must not break the RTAI priority hierarchy when real-time tasks are operating into the Linux domain.

RTAI/fusion will be adapted for running on the Linux 2.6 kernel in the future, but as of now, please put on your 2.4 glasses when reading this document.

6.4 Implementation

6.4.1 Current Implementation

RTAI/fusion is built upon the implementation of a mutable priority scheme for the Linux placeholder task. As a remainder, this pseudo-task currently represents all the non real-time Linux tasks, and acts as a common fallback schedulable object to pick when no real-time RTAI tasks are ready to run.

The test code is built upon the Xenomai nucleus. Xenomai provides the scheduling of kernel-based and user-space threads, using a technology derived from LXRT, and incorporating the implementation details listed below (see 7). This nucleus includes a real-time, FIFO-based, static priority scheduler.

Like LXRT, the Xenomai scheduler defines a Linux placeholder called the "root" thread which gets scheduled as a fallback option when no other Xenomai threads are runnable into the RTAI domain. This causes the RTAI domain to suspend itself Adeos-wise, yielding control to the next domain down the pipeline.

Each Xenomai thread capable of executing in user-space has one kernel-based TCB called a "shadow", along with its standard Linux-controlled task structure. Real-time scheduling of shadow threads is done using those TCBs, but the register state is exclusively saved into the Linux task structures by both Linux and Xenomai schedulers during context switching. In Xenomai's terminology, a shadow thread is the real-time control block of a Linux task also schedulable by Xenomai. Both schedulers control the task in a mutually exclusive manner.

The priority of the root thread is mutable, and can be changed dynamically to reflect the position of the Linux kernel activity within the real-time priority hierarchy.

A Linux-controlled gatekeeper task is in charge of resuming a Xenomai thread migrating from Linux to the RTAI domain. This thread is the equivalent of LXRT's kthread_b() helper task.

Mutable priority works like this:

In a system defining T1(prio=1,domain=RTAI), T2(prio=2,domain=RTAI), and ROOT(prio=0,domain=Linux), we would have the following transition:

```
T2 runs in domain RTAI...
   T2 migrates to the Linux domain (for executing a syscall)
       ROOT inherits priority(T2)
                T2 executes the syscall on behalf of ROOT
                (at this point, interrupts are no more propagated to
                the Linux kernel)
   T2 migrates back to the RTAI domain (after the syscall completes)
        T2 resumes into the RTAI domain
        ROOT priority is lowered back (to 0 if no other real-time
        threads are runnable into the Linux domain, or inherited from the
        next real-time thread to execute in this domain). As soon as the
        root thread recovers its base priority (0), the interrupt flow is
        restarted toward the Linux kernel.
T2 suspends...
T1 resumes...
```

As shown, T2 migration would not cause the preemption of the root thread by T1 despite its lower initial priority, since ROOT's priority would not allow it after it has been dynamically raised.

In practice, T2 would not be forced to exit the Linux domain right after the syscall completes, but only when it issues a RTAI syscall. The migration back to RTAI has been described for illustration purposes.

Obviously, ROOT never migrates to RTAI, and is always runnable.

6.4.2 Implementation details with regard to priority mutation

When informed by Adeos that a new Linux task is about to be switched in (ADEOS_SCHEDULE_HEAD event), we check in the event handler whether a shadow thread is paired with the incoming task. In such a case, the root thread priority is raised to the priority of the shadow, otherwise, this priority is set to the "idle" priority level (i.e. non-realtime priority).

In the RTAI code controlling the migration of the current thread from RTAI to the Linux domain, the priority of the root thread is inherited from the migrating thread.

In the RTAI code controlling the migration from the current Xenomai thread from Linux to the RTAI domain, the priority of the gatekeeper thread is inherited from the migrating thread.

6.5 Hardware/Software pre-requisites

The experimental RTAI/fusion subsystem currently runs on the following configuration:

- Linux 2.4.21/ia32
- ADEOS combo patch (Adeos + Kpreempt + Lolat)
- Xenomai nucleus from the RTAI vesuvio branch.

A simple demo is available for download here:

http://savannah.gnu.org/download/xenomai/fusion/

6.6 Programming with POSIX API

6.6.1 Threads

6.6.2 ...

6.6.3 API Reference

6.7 Interprocess Communication

6.7.1 Overview

6.7.2 Mutexes

6.7.3 Conditional Variables

6.7.4 Queues

6.7.5 API Reference

7 Floating Point Operation

8 LXRT

8.1 Task Creation

8.2 Interprocess Communication

9 The Common API

9.1 Why a Common API

9.2 Tasks

10 Perl

10.1 Intention

10.2 API Reference

11 Watchdogs

11.1 Overview

11.2 API Reference

Part IV

Development Strategies

12 Debugging

12.1 Introduction

12.2 Debugging Kernel Modules

12.2.1 rt_printk and dmesg

12.2.2 kdbg

12.2.3 r2d2

12.2.4 LTT

12.3 Debugging LXRT

Part V

RTAI Internals

13 The RTAI Makefile System

13.1 Description of the Build System

13.1.1 Motivations

The RTAI build system is a merge of Linux's 2.5 Kconfig with autoconf/automake/libtool. The following requirements led to such mixed system:

- GUI-based configuration.
- Cross-compilation support.
- Compilation of C and C++ kernel modules.
- Full shared library support (C/C++).
- C++ compilation of hosts programs.
- Compilation of GUI (X-based).
- Script-based configuration (for automatic mode).
- Initial support for profile-based configurations.
- Reliable (host) feature detection framework.

13.1.2 Dynamics

The dynamics of the system is as follows:

=¿ Kconfig/Menuconfig-based configuration (GUI-aided)

$ make menuconfig,xconfig,gconfig

— — 1. Produces rtai-core/config/kconfig/.rtai_config (Kconfig format: CONFIG_RTAI_FEATURE=¡value¿ or unset) — 2. Runs ./configure –with-kconfig-file=rtai-core/config/kconfig/.rtai_config v

=¿ Script level configuration (automatic or manual). The configure script can either be controlled by individual options passed to it, or fed with a configuration file in Kconfig format.

Automatic mode (spawned right after kconfig): ./configure –with-kconfig-file=rtai-core/config/kconfig/.rtai_config

Manual mode (from the user CLI): ./configure –enable-feature-x –with-param-y=n

— — 1. Applies a set of reasonable defaults for unset parameters. — 2. Auto-detects host features as required. — 3. Produces config.h and Makefiles. v

=¿ Compilation

$ make

13.1.3 Bootstrapping

RTAI needs to be configured prior to being built, but the configuration tool cannot rely on autoconf-generated Makefiles to build itself. Therefore, the GUI-aided configuration system which depends on dynamically built programs is bootstrapped by a simple "makefile" (note the small caps) at the root of the source tree.

The bootstrap makefile traps the targets needed to build and run the Kconfig tool (make menuconfig—xconfig—gconfig), and invokes the self-contained rtai-core/config/kconfig/Makefile.kc makefile which in turn builds and run the configuration GUI as required. Upon return of the GUI-based configuration session, the bootstrap makefile (re)runs autoconf's "configure" script as needed to take the configuration changes into account.

Once the configuration script has been run successfully at least once, there is no more need to use the bootstrap makefile located in the source tree, since the top-level autoconf-generated Makefile will be available in the build tree, and handle the menuconfig, xconfig, gconfig and other targets appropriately (actually, menuconfig, xconfig and gconfig are silently passed to the boostrap makefile by the autoconf-generated one so that we do not duplicate rather complex rules uselessly).

13.1.4 Dependencies

The build system is based on:

- Kconfig as extracted from Linux 2.5.69 development version. The X-based configuration front-ends either needs Qt (xconfig) or GTK2 (gconfig), and the dialog-based one depends on ncurses.
- Autoconf ¿= 2.57
- Automake ¿= 1.7.3
- Libtool ¿= 1.4.3

13.1.5 Configuration Variables

The following variables are substituted by autoconf into each automake-controlled Makefile.

@RTAI_TARGET_ARCH@ The canonical name of the target architecture for which RTAI is built. Currently, "i386" is the only supported architecture.

@RTAI_BUILTIN_MODLIST@ A white-space separated list of (otherwise modular) features to be integrated into the RTAI scheduler(s). Each name found in this list is canonicalised relatively to the rtai-core/ directory, and can be one of:

- trace (LTT support)
- math (In-kernel libm support)
- ipc/bits (event flags support)
- ipc/fifos (FIFOS support)
- ipc/netrpc (NETRPC support)
- ipc/tbx (Typed mailboxes support)
- ipc/shmem (Shared memory support)
- rtmem (Dynamic memory management)
- tasklets (Tasklets support)
- usi (User-space interrupts support)
- watchdog (Task watchdog support)

@RTAI_KMOD_CFLAGS@ Basic CFLAGS used to compile kernel modules written in C.

@RTAI_KMOD_CXXFLAGS@ Basic CFLAGS used to compile kernel modules written in C++.

@RTAI_USER_CFLAGS@ Basic CFLAGS used to compile user-space programs.

@RTAI_FP_CFLAGS@ Additional CFLAGS used to compile objects including math operations. This variable's value depends on CONFIG_RTAI_FPU_SUPPORT to determine whether we should ask GCC to use hardware-assisted or software-emulated floating-point support.

@RTAI_COMPAT_CFLAGS@ When the compatibility mode is activated, this variable contains additional directives that basically help finding compatibility headers. Otherwise, this variable is empty.

@RTAI_MVM_LDADD@, @RTAI_MVM_CFLAGS@ @RTAI_MVM_CXXFLAGS@ @RTAI_MVM_GCC_TARBALL@ Specific flags to compile and link the Xenomai simulator. These are not used outside of the virtual machine environment.

14 Ports

14.1 Porting RTAI to New Architectures

14.2 x86

14.3 ARM

14.4 PowerPC

14.5 MIPS

14.6 m68k

14.7 cris

Part VI

Appendix

A Licenses

A.1 The RTAI Licenses

FIXME: write some introduction text.

A.2 GPL

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

¡one line to give the program's name and a brief idea of what it does.¿ Copyright (C) ¡year¿ ¡name of author¿

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a ”copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

¡signature of Ty Coon¿, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

A.3 LGPL

GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author’s reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

¡one line to give the library's name and a brief idea of what it does.¿ Copyright (C) ¡year¿ ¡name of author¿

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

¡signature of Ty Coon¿, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

B Bibliography