# Real-time POSIX over RTAI/fusion

LSM – July 6, 2005

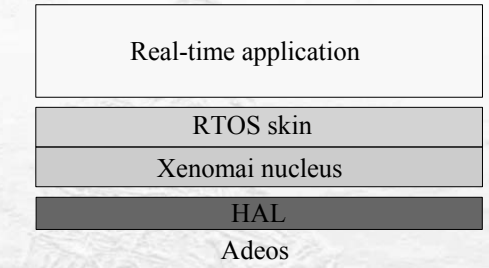
Philippe Gerum, rpm@openwide.fr



# Current development state

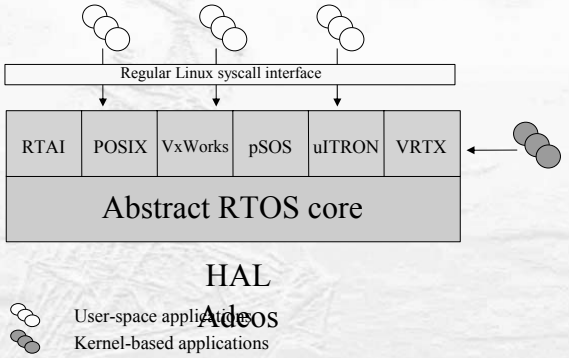
- RTAI/fusion
  - Linux 2.6 and above (x86, ppc32, ia64)
  - Portable to any Adeos-enabled architecture
  - Pervasive hard RT support in user-space
  - Nanokernel technology
  - Traditional RTOS emulation
- v0.8.2 released, July 2005
  - v0.9 due by August (ppc64 port)

# Xenomai-based architecture

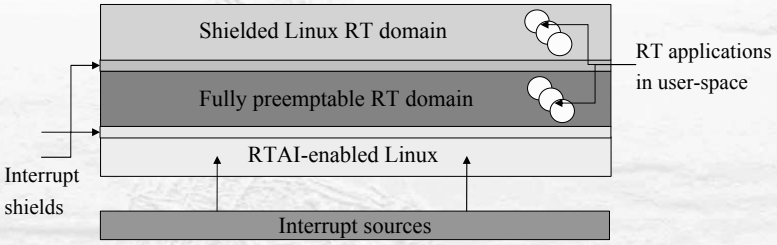





# API skins in RTAI/fusion

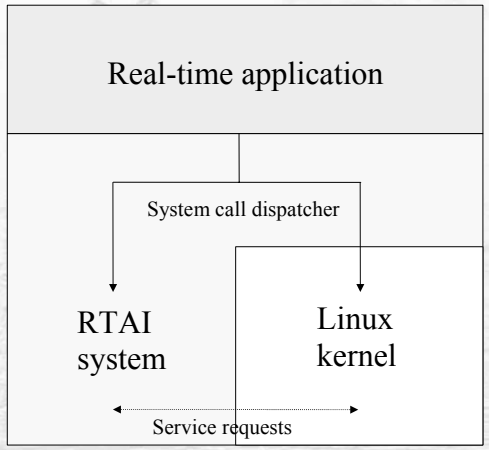


# Real-time execution domains






# Seamless RTAI services integration



# Orthogonal real-time approaches

- PREEMPT_RT (Linux domain)
  - Access to standard device drivers
  - Large fully preemptable code base
- RTAI/fusion (RTAI domain)
  - Lightweight
  - Stringent determinism
  - Not impacted by “rogue” code



# PREEMPT_RT in brief

- Fully preemptible kernel
  - multiple preemption models
  - real-time aware synchronization objects
  - threaded interrupt model
- Major kernel overhaul
  - fixing latency sources (VM, RCU, net, drivers etc.)
  - adapting to new synchronization semantics

# Advantages in combining

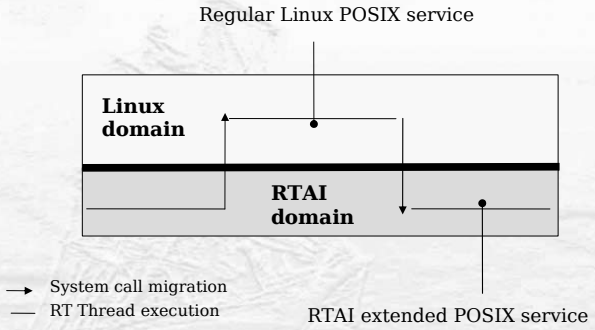
- Broader hardware spectrum
  - RTAI/fusion: low-end to mid-range
  - PREEMPT_RT: mid-range to high-end
- Broader performance spectrum
  - RTAI/fusion: low-latency guaranteed
  - PREEMPT_RT: low-latency best effort
- Broader application spectrum
  - RTAI/fusion: machine control
  - PREEMPT_RT: audio / video systems



# Mixed POSIX support

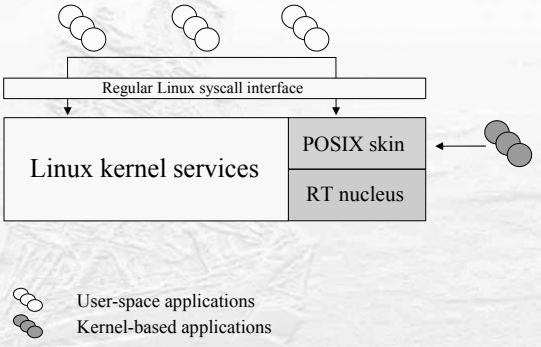
- Mixed POSIX interfaces
  - RTAI/fusion: time-critical PSE51 sub-set
  - PREEMPT_RT: access to regular sub-systems
- Shadowed POSIX calls
  - Fusion/POSIX wrapper library
  - Regular execution of non-shadowed calls
- Dynamic service dispatching
  - System call interposition
  - Execution domain migration

# Fusion thread migration






# Hardened POSIX support



# Hardened POSIX calls

- pthread management
  - creation, deletion, scheduling
- mutexes
  - all operations
- counting semaphores
  - all operations
- message queues
  - all operations
- clock management
  - timed waits and clock status



# Conclusion

- Broad real-time application spectrum
  - No “one-fits-it-all” real-time Linux approach
- Varying per-application requirements
  - Different per-thread constraints
  - Need for accessing “vanilla” Linux services
- User-space programming model
  - Seamless integration
  - Memory protection
  - Easy access to common tools (e.g. GDB)

# Thank you for attending.