Writing Linux Real-Time Applications

John Ogness

`<john.ogness@linutronix.de>`

Embedded Recipes 2025

Linux can do hard real-time!

Linux can do hard real-time!

(missed deadline = critical failure)

Linux can do hard real-time!

(missed deadline = critical failure)

… and since 6.12 it is mainline!

It is really time to celebrate!
25th anniversary of RTLWS
20th anniversary of Preempt-RT

Behaviors to consider for real-time

- memory in physical RAM
- real-time policies and priorities
- synchronization and notification
- cyclic tasks
- task and interrupt CPU affinities
- real-time networking

Linux Behavior

Memory in Physical RAM

Linux Behavior

Memory in Physical RAM

Linux Maps Memory on Demand

- heap(s) and stack(s)
- allocation via mmap()
- text and data segments

Linux Recycles Memory

- swapping pages to disk
- reclaiming pages it can recover from disk
- reclaiming unused heap space

Solution

- `mlockall(2)`, pre-faulting, glibc tuning with `mallopt(3)`

Linux Behavior

Real-Time Policies and Priorities

Linux Behavior

Real-Time Policies and Priorities

Policies

- `SCHED_FIFO`
- `SCHED_RR` (same prio = round robin)

Priorities

- 99 = high priority
- 1 = low priority

API

- `chrt(1)`
- `sched_setscheduler(2)`

Linux Behavior

Real-Time Policies and Priorities

Policies

- `SCHED_FIFO`
- `SCHED_RR` (same prio = round robin)

Priorities

- 99 = high priority
- 1 = low priority

API

- `chrt(1)`
- `sched_setscheduler(2)`

`SCHED_DEADLINE` also exists, but it is complex to combine with priority-based scheduling.

Linux Behavior

Synchronization and Notification

Linux Behavior

Synchronization and Notification

Locking

- `pthread_mutex_t`
- `PTHREAD_PRIO_INHERIT` (mutex protocol)

Conditional Variables

- `pthread_cond_t`
- see also `librtpi` (efficient ownership transfers)

Linux Behavior

Cyclic Tasks

Linux Behavior

Cyclic Tasks

Beware of RT-Unsafe APIs!

- **timerfd** (deferred interrupt handling)
- **POSIX timers** (deferred interrupt handling, based on signals)

Correct Implementation

- **dedicated thread**
- `clock_nanosleep(2)` (wakes from hardware interrupt handler)
- `CLOCK_MONOTONIC` (immune to time setting)
- `TIMER_ABSTIME` (avoids variance and drift)

Linux Behavior

Task and Interrupt CPU Affinities

Linux Behavior

Task and Interrupt CPU Affinities

Isolating and Pinning CPUs

- `cpuset(7)` **(cgroups)**
- `isolcpus` **boot argument (deprecated)**

API

- `taskset(1)`
- `sched_setaffinity(2)`
- `cgroups(7)`
- `/proc/irq/IRQ_NUMBER/smp_affinity`

Linux Behavior

Real-Time Networking

Linux Behavior

Real-Time Networking

TSN Hardware Support

- PTP
- 802.1Qav
- 802.1Qbv
- Tx Launch Time
- Multi-Queue

Isolated CPU for RT Networking

- hardware interrupt
- interrupt kthread
- NAPI instance kthread
- RT network application

Linux Behavior

Real-Time Networking

TSN Hardware Support

- PTP
- 802.1Qav
- 802.1Qbv
- Tx Launch Time
- Multi-Queue

Isolated CPU for RT Networking

- hardware interrupt
- interrupt kthread
- NAPI instance kthread
- RT network application

Patches currently in review to remove isolated CPU requirement.

Linux Behavior

Real-Time Networking

Linux Real-Time Communication Testbench

RTC-Testbench

`https://github.com/Linutronix/RTC-Testbench`

A real-time and non-real-time traffic validation tool for converged ethernet networks with and without utilization of TSN mechanisms.

Design

RT Application Design Considerations

Design

RT Application Design Considerations

Event Flow

- transition from higher to lower priorities
- hardware interrupt highest ”priority” in an event chain

Real-Time Work

- only use real-time priority for real-time work
- do not abuse priorities for ”scheduler tuning”

RT-Safe Sleeping

- consider the waker in all scenarios
- the waker is the parent in an event flow

So now we will all go out and write perfect real-time applications, right?

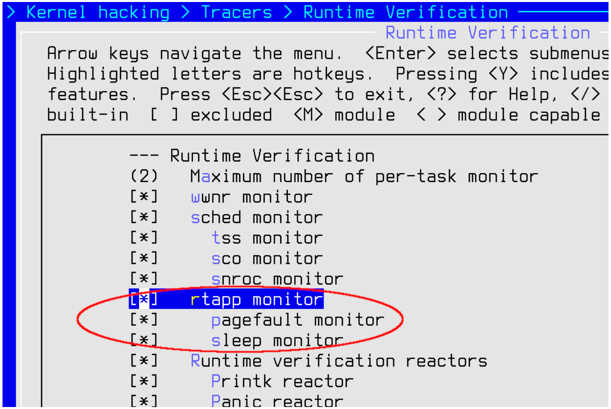
So now we will all go out and write perfect real-time applications, right?

Wrong!

- Linux provides many interfaces and the implementation details are often unknown to userspace developers
- not all subsystems/drivers use the same semantics
- libraries may not be real-time safe (even the C library)
- This is all a lot to watch out for!

RV Monitor rtapp

```
Subject: [PATCH v8 00/22] RV: Linear temporal logic monitors for
         RT application
Date: Mon, 12 May 2025 12:50:43 +0200
Message-ID: <cover.1747046848.git.namcao@linutronix.de>
```



```
> Kernel hacking > Tracers > Runtime Verification ──────
                                   Runtime Verification ─
  Arrow keys navigate the menu.  <Enter> selects submenus
  Highlighted letters are hotkeys.  Pressing <Y> includes
  features.  Press <Esc><Esc> to exit, <?> for Help, </>
  built-in  [ ] excluded  <M> module  < > module capable

          --- Runtime Verification
          (2)   Maximum number of per-task monitor
          [*]   wwnr monitor
          [*]   sched monitor
          [*]     tss monitor
          [*]     sco monitor
          [*]     snroc monitor
          [*]   rtapp monitor
          [*]     pagefault monitor
          [*]     sleep monitor
          [*]   Runtime verification reactors
          [*]     Printk reactor
          [*]     Panic reactor
```

RV Monitor rtapp

Usage

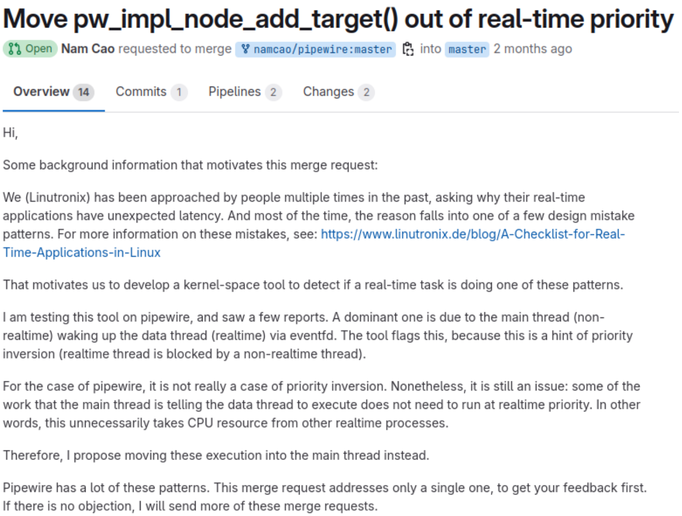
```
# echo 1 > /sys/kernel/debug/tracing/rv/monitors/rtapp/enable

# echo printk > /sys/kernel/debug/tracing/rv/monitors/rtapp/reactors

# perf record -g --call-graph dwarf -a \
              -e rv:error_sleep -e rv:error_pagefault

# perf script
pipewire  1234 [000] 10829.141465: rv:error_sleep: pipewire[1234]: violat
    ffffffff93e89649 ltl_validate+0x3d9 ([kernel.kallsyms])
    ffffffff93e89649 ltl_validate+0x3d9 ([kernel.kallsyms])
    ffffffff93e89b3a handle_sched_set_state+0x9a ([kernel.kallsyms])
    ffffffff93d12323 __trace_set_current_state+0x63 ([kernel.kallsyms])
    ffffffff940b7f42 do_epoll_wait+0x2a2 ([kernel.kallsyms])
    ffffffff940b96a1 __x64_sys_epoll_wait+0x61 ([kernel.kallsyms])
    ffffffff948cceca do_syscall_64+0x8a ([kernel.kallsyms])
    ffffffff93a0012f entry_SYSCALL_64_after_hwframe+0x76 ([kernel.kallsym
              108ee6 epoll_wait+0x56 (/usr/lib/x86_64-linux-gnu/libc.so.6
               1710f impl_pollfd_wait+0x3f (/usr/lib/x86_64-linux-gnu/spa
                8d4a loop_iterate+0xaa (/usr/lib/x86_64-linux-gnu/spa-0.2
               4803b do_loop+0xcb (/usr/lib/x86_64-linux-gnu/libpipewire-
               891f4 start_thread+0x304 (/usr/lib/x86_64-linux-gnu/libc.s
              108aff clone+0x3f (inlined)
```

PipeWire



Move pw_impl_node_add_target() out of real-time priority

Open **Nam Cao** requested to merge `namcao/pipewire:master` into `master` 2 months ago

Overview 14 | Commits 1 | Pipelines 2 | Changes 2

Hi,

Some background information that motivates this merge request:

We (Linutronix) has been approached by people multiple times in the past, asking why their real-time applications have unexpected latency. And most of the time, the reason falls into one of a few design mistake patterns. For more information on these mistakes, see: https://www.linutronix.de/blog/A-Checklist-for-Real-Time-Applications-in-Linux

That motivates us to develop a kernel-space tool to detect if a real-time task is doing one of these patterns.

I am testing this tool on pipewire, and saw a few reports. A dominant one is due to the main thread (non-realtime) waking up the data thread (realtime) via eventfd. The tool flags this, because this is a hint of priority inversion (realtime thread is blocked by a non-realtime thread).

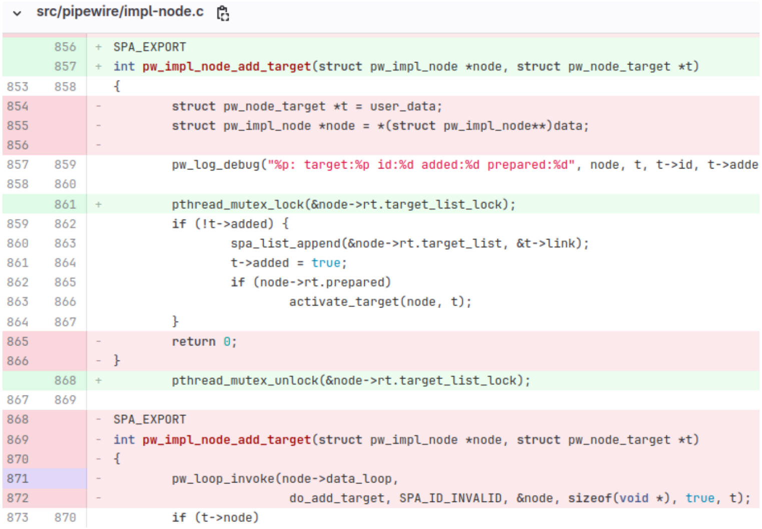
For the case of pipewire, it is not really a case of priority inversion. Nonetheless, it is still an issue: some of the work that the main thread is telling the data thread to execute does not need to run at realtime priority. In other words, this unnecessarily takes CPU resource from other realtime processes.

Therefore, I propose moving these execution into the main thread instead.

Pipewire has a lot of these patterns. This merge request addresses only a single one, to get your feedback first. If there is no objection, I will send more of these merge requests.

https://gitlab.freedesktop.org/pipewire/pipewire/-/merge_requests/2285

PipeWire



src/pipewire/impl-node.c

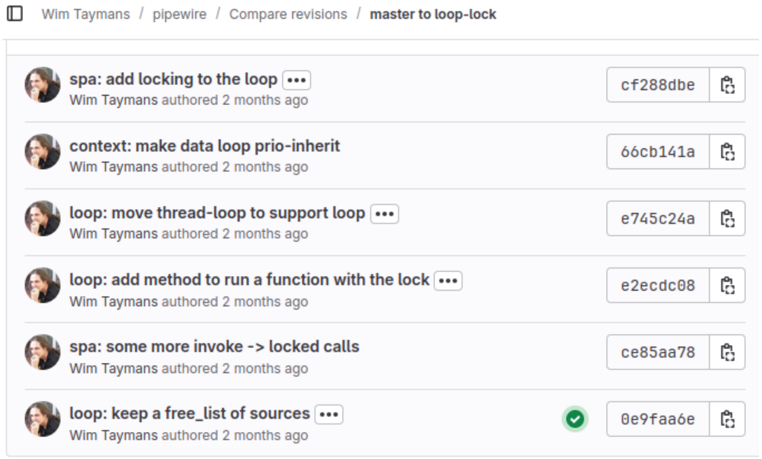
```
+ SPA_EXPORT
+ int pw_impl_node_add_target(struct pw_impl_node *node, struct pw_node_target *t)
  {
-         struct pw_node_target *t = user_data;
-         struct pw_impl_node *node = *(struct pw_impl_node**)data;
-
          pw_log_debug("%p: target:%p id:%d added:%d prepared:%d", node, t, t->id, t->adde

+         pthread_mutex_lock(&node->rt.target_list_lock);
          if (!t->added) {
                  spa_list_append(&node->rt.target_list, &t->link);
                  t->added = true;
                  if (node->rt.prepared)
                          activate_target(node, t);
          }
-         return 0;
- }
+         pthread_mutex_unlock(&node->rt.target_list_lock);

- SPA_EXPORT
- int pw_impl_node_add_target(struct pw_impl_node *node, struct pw_node_target *t)
- {
-         pw_loop_invoke(node->data_loop,
-                         do_add_target, SPA_ID_INVALID, &node, sizeof(void *), true, t);
          if (t->node)
```

Do not post non-real-time work to real-time loop.

PipeWire


Wim Taymans / pipewire / Compare revisions / master to loop-lock

spa: add locking to the loop •••
Wim Taymans authored 2 months ago — cf288dbe

context: make data loop prio-inherit
Wim Taymans authored 2 months ago — 66cb141a

loop: move thread-loop to support loop •••
Wim Taymans authored 2 months ago — e745c24a

loop: add method to run a function with the lock •••
Wim Taymans authored 2 months ago — e2ecdc08

spa: some more invoke -> locked calls
Wim Taymans authored 2 months ago — ce85aa78

loop: keep a free_list of sources •••
Wim Taymans authored 2 months ago — 0e9faa6e

https://gitlab.freedesktop.org/wtaymans/pipewire/-/tree/loop-lock?ref_type=heads

PipeWire

Other issues found with PipeWire

- many more sites of assigning non-real-time work to real-time loop
- page faults
- timerfd usage

Questions / Comments

Thank you for your attention!

John Ogness <john.ogness@linutronix.de>

Nam Cao <namcao@linutronix.de>