ELISA
Enabling Linux in Safety Applications

WORKSHOP

NASA Goddard

Real-Time Linux Update

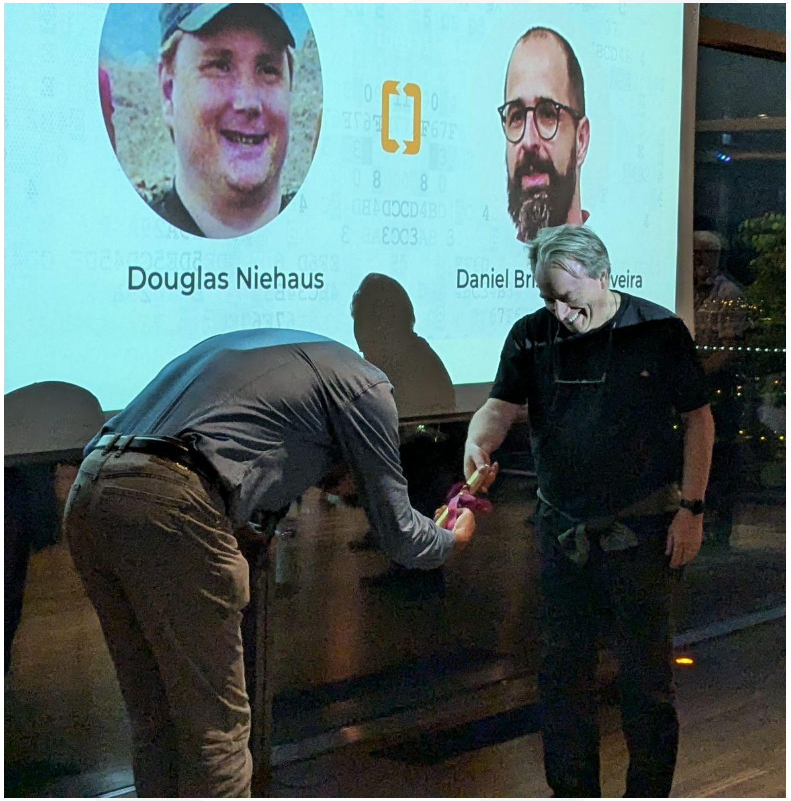
Real-Time Linux Update

IT’S DONE!

ELISA Enabling Linux in Safety Applications

WORKSHOP

September 19, 2024



ELISA Enabling Linux in Safety Applications

WORKSHOP

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS

ELISA Enabling Linux in Safety Applications

WORKSHOP

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary
  - No unbounded priority inversion

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary
  - No unbounded priority inversion
  - Missed deadlines are bugs!

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary
  - No unbounded priority inversion
  - Missed deadlines are bugs!
- Deterministic Operating System

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary
  - No unbounded priority inversion
  - Missed deadlines are bugs!
- Deterministic Operating System (DOS!)

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary
  - No unbounded priority inversion
  - Missed deadlines are bugs!
- Deterministic Operating System (DOS!)
  - May impact average performance (not “fast”)

What is Real-Time Linux?

- Started in 2004 with the mandate to make Linux itself into a true RTOS
  - No micro-kernels!
  - Runs normal Linux programs as RT (no special system calls)
- Hard Real-Time Designed
  - All needed paths must have a theoretical worst case boundary
  - No unbounded priority inversion
  - Missed deadlines are bugs!
- Deterministic Operating System (DOS!)
  - May impact average performance (not “fast”)
  - Fastest worst case scenarios

What was made to make Linux Real-Time?

- Turned hard interrupt handlers into threads

What was made to make Linux Real-Time?

- Turned hard interrupt handlers into threads
  - Allows interrupt handlers to be preempted

What was made to make Linux Real-Time?

- Turned hard interrupt handlers into threads
  - Allows interrupt handlers to be preempted
  - Can prioritise interrupt handlers, one interrupt handler can preempt another

What was made to make Linux Real-Time?

- Turned hard interrupt handlers into threads
  - Allows interrupt handlers to be preempted
  - Can prioritise interrupt handlers, one interrupt handler can preempt another
  - Better control of interrupt processing on which CPU

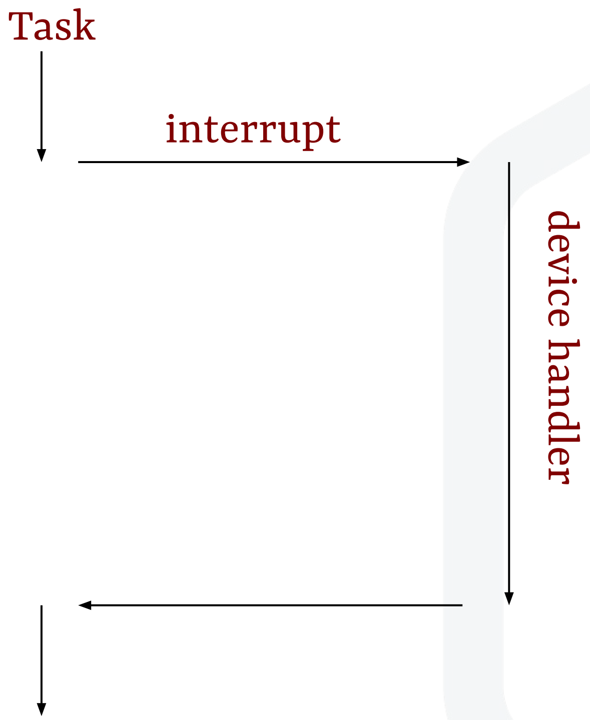
What was made to make Linux Real-Time?

- Turned hard interrupt handlers into threads
  - Allows interrupt handlers to be preempted
  - Can prioritise interrupt handlers, one interrupt handler can preempt another
  - Better control of interrupt processing on which CPU
  - Hard interrupt may go off on one CPU, but processing can be on another

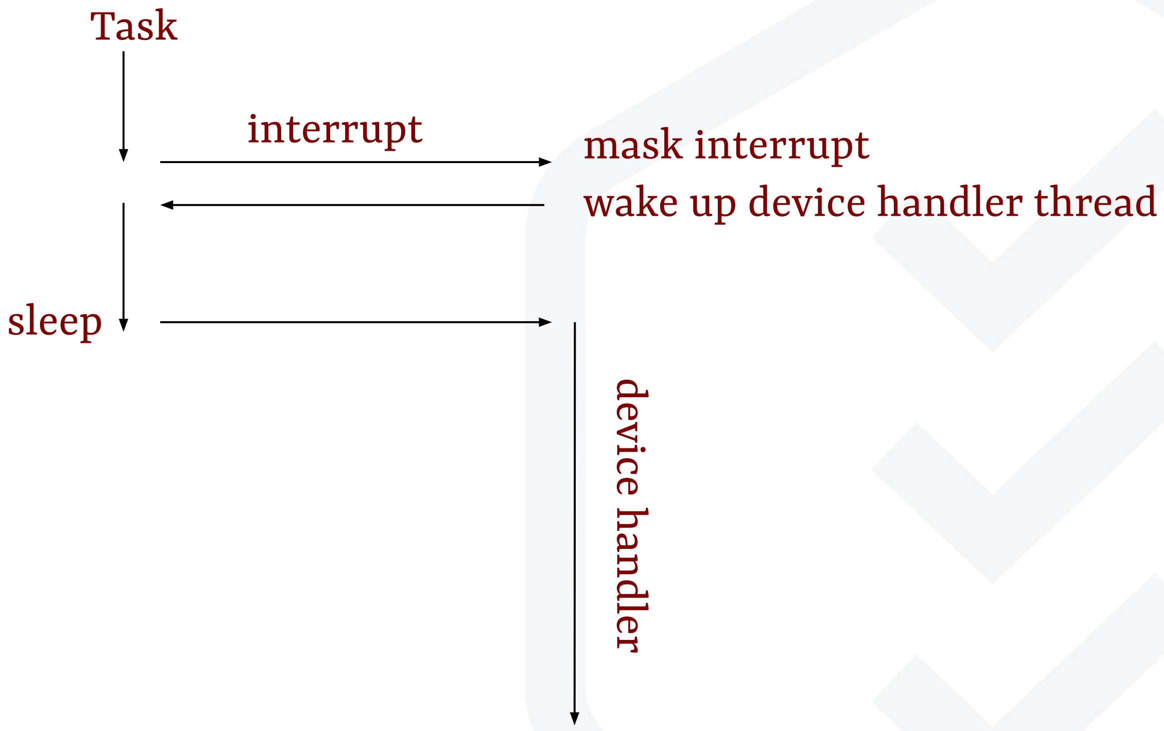
What was made to make Linux Real-Time?

- Turned hard interrupt handlers into threads
  - Allows interrupt handlers to be preempted
  - Can prioritise interrupt handlers, one interrupt handler can preempt another
  - Better control of interrupt processing on which CPU
  - Hard interrupt may go off on one CPU, but processing can be on another
- Added to mainline Linux in v2.6.39 (2011)

Normal hard interrupt handlers



Threaded interrupt handlers



What was made to make Linux Real-Time?

- Soft interrupts run by the thread who needs them

What was made to make Linux Real-Time?

- Soft interrupts run by the thread who needs them
- Soft interrupts are interrupts that can be interrupted by hard interrupts
  - Networking processing
  - Timer processing
  - Block devices
  - RCU
  - Etc

What was made to make Linux Real-Time?

- Soft interrupts run by the thread who needs them
- Soft interrupts are interrupts that can be interrupted by hard interrupts
  - Networking processing
  - Timer processing
  - Block devices
  - RCU
  - Etc
- Non-RT, runs like normal interrupts (preempts current task)

What was made to make Linux Real-Time?

- Soft interrupts run by the thread who needs them
- Soft interrupts are interrupts that can be interrupted by hard interrupts
  - Networking processing
  - Timer processing
  - Block devices
  - RCU
  - Etc
- Non-RT, runs like normal interrupts (preempts current task)
- Ksoftirqd - runs when too many softirqs are queued on a CPU

What was made to make Linux Real-Time?

- In PREEMPT_RT, softirqs execute in the task context that raised them

What was made to make Linux Real-Time?

- In PREEMPT_RT, softirqs execute in the task context that raised them
- If a task raises a softirq, most likely the softirq is running for that task

What was made to make Linux Real-Time?

- In PREEMPT_RT, softirqs execute in the task context that raised them
- If a task raises a softirq, most likely the softirq is running for that task
- Softirqs raised by interrupt threads, run as the context of that interrupt thread
    - Networking and RCU

What was made to make Linux Real-Time?

- In PREEMPT_RT, softirqs execute in the task context that raised them
- If a task raises a softirq, most likely the softirq is running for that task
- Softirqs raised by interrupt threads, run as the context of that interrupt thread
  - Networking and RCU
- Ksoftirq can still run (but runs as RT priority 1)

What was made to make Linux Real-Time?

- Spin locks turned into mutexes (sleeping spin locks)
  - Spin locks are per CPU and spin while waiting for other CPUs
  - Can not be preempted when held
  - Can be held for a long time
  - RT converts them to a “mutex” or sleeping lock

What was made to make Linux Real-Time?

- Spin locks turned into mutexes (sleeping spin locks)
  - Spin locks are per CPU and spin while waiting for other CPUs
  - Can not be preempted when held
  - Can be held for a long time
  - RT converts them to a “mutex” or sleeping lock
- Requires threaded interrupts (as interrupt handlers take spin_locks)

What was made to make Linux Real-Time?

- Spin locks turned into mutexes (sleeping spin locks)
  - Spin locks are per CPU and spin while waiting for other CPUs
  - Can not be preempted when held
  - Can be held for a long time
  - RT converts them to a “mutex” or sleeping lock
- Requires threaded interrupts (as interrupt handlers take spin_locks)
- There are some spin_locks that can not be converted
  - These spin locks are called “raw_spin_lock”
  - They are called when preemption is disabled or in real interrupt context

What was made to make Linux Real-Time?

- Spin locks turned into mutexes (sleeping spin locks)
  - Spin locks are per CPU and spin while waiting for other CPUs
  - Can not be preempted when held
  - Can be held for a long time
  - RT converts them to a “mutex” or sleeping lock
- Requires threaded interrupts (as interrupt handlers take spin_locks)
- There are some spin_locks that can not be converted
  - These spin locks are called “raw_spin_lock”
  - They are called when preemption is disabled or in real interrupt context
- Added to mainline Linux in v6.12 (2024)

What was made to make Linux Real-Time?

- Priority inheritance
    - All mutex (including sleeping spin locks) have priority inheritance
    - Prevents unbounded priority inversion

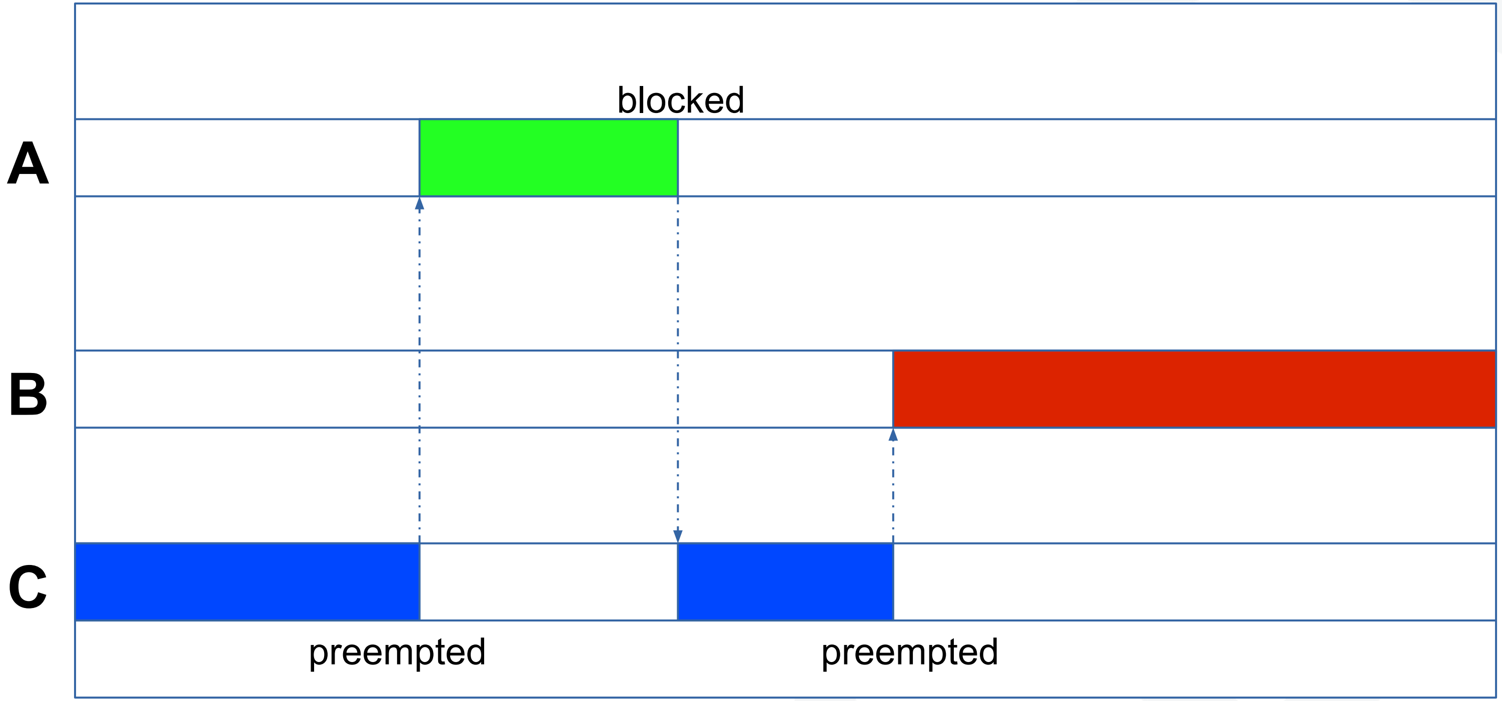
What was made to make Linux Real-Time?

- Priority inheritance
  - All mutex (including sleeping spin locks) have priority inheritance
  - Prevents unbounded priority inversion
- Added to mainline Linux for user space futex in v2.6.18 (2006)

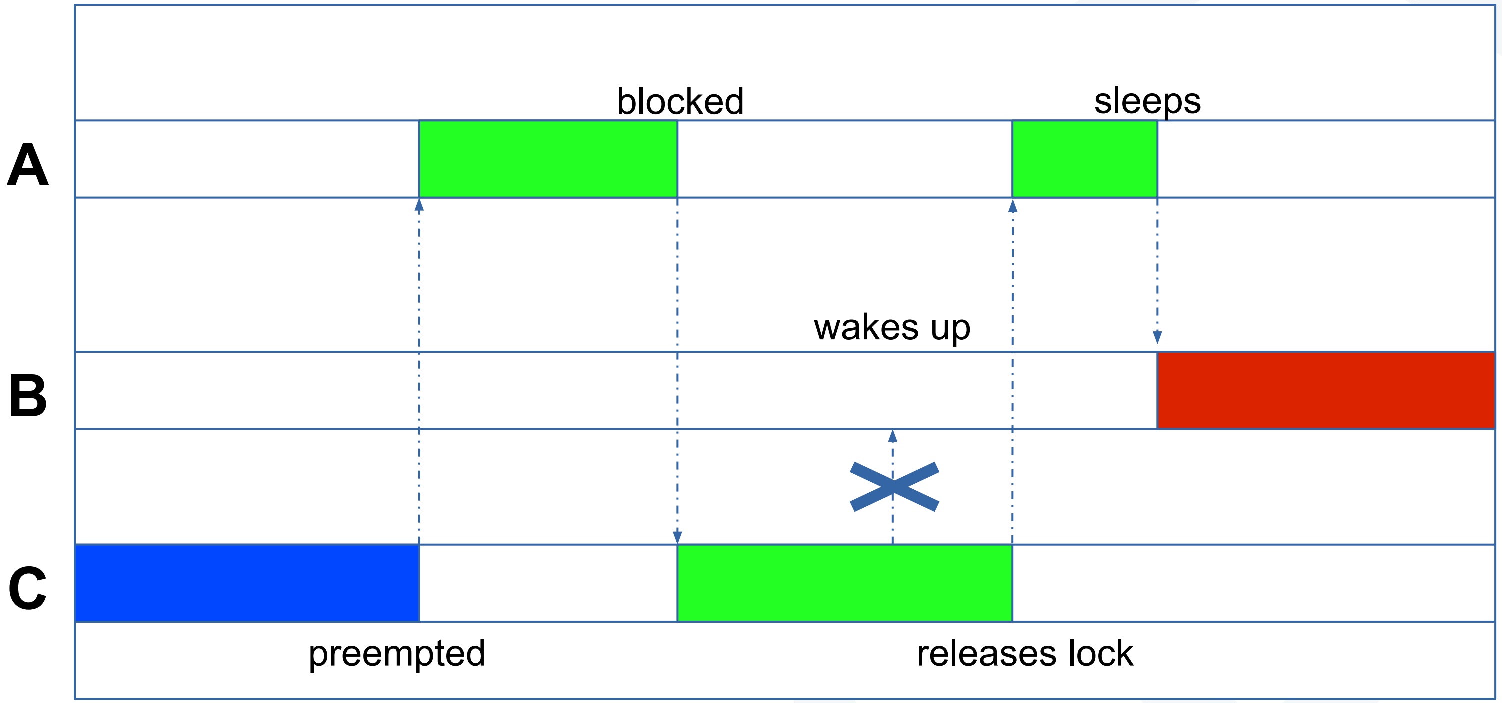
What was made to make Linux Real-Time?

- Priority inheritance
  - All mutex (including sleeping spin locks) have priority inheritance
  - Prevents unbounded priority inversion
- Added to mainline Linux for user space futex in v2.6.18 (2006)
- Added to mainline Linux for kernel mutex in v6.12 (2024)

Unbounded priority inversion



Bounded priority inversion



Status of Linux Real-Time

Status of Linux Real-Time

IT'S DONE!

Status of Linux Real-Time

IT’S DONE!

Not really, but almost!

ELISA Enabling Linux in Safety Applications

WORKSHOP

Status of Linux Real-Time

- PREEMPT_RT finally made it into v6.12

Status of Linux Real-Time

- PREEMPT_RT finally made it into v6.12
  - But don’t use that kernel if you want RT (use at least v6.13)

ELISA Enabling Linux in Safety Applications WORKSHOP

Status of Linux Real-Time

- PREEMPT_RT finally made it into v6.12
    - But don’t use that kernel if you want RT (use at least v6.13)
    - It is missing NEED_RESCHED_LAZY

Status of Linux Real-Time

- PREEMPT_RT finally made it into v6.12
  - But don’t use that kernel if you want RT (use at least v6.13)
  - It is missing NEED_RESCHED_LAZY
- NEED_RESCHED_LAZY added in v6.13
  - Does not preempt sleep “spin lock” for non-RT tasks
  - Improves performance of non-RT tasks

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6.13.y-rt

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

Status of Linux Real-Time

Module loading

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linu

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

Serial Console

ELISA Enabling Linux in Safety Applications

WORKSHOP

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6.13.y-rt

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

Basic cleanups

ELISA Enabling Linux in Safety Applications

WORKSHOP

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6.13.y-rt

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

General i915 Hacks

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6.13.y-rt

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_loc
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

General ARM Hacks

ELISA Enabling Linux in Safety Applications

WORKSHOP

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6.13.y-rt

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_irq_save()
96edfd863ffa powerpc/pseries: Select the generic memory allocator.
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation for PREEMPT_RT
b35d95e74643 powerpc/stackprotector: work around stack-guard init from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

General PPC Hacks

Status of Linux Real-Time

```
$ git log --pretty=oneline --abbrev-commit --reverse v6.13-rc1..linux-6.13.y-rt

d16c00faab9a preempt: Move PREEMPT_RT before PREEMPT in vermagic.
a22caa611be1 serial: 8250: Switch to nbcon console
acabbf65e61d serial: 8250: Revert "drop lockdep annotation from serial8250_clear_IER()"
75c8428eb7e9 module: Use complete RCU protection instead a mix of RCU and RCU-sched.
389b918f9807 preempt: Add a generic function to return the preemption string.
db7099a324c0 drm/i915: Use preempt_disable/enable_rt() where recommended
acba58824ae5 drm/i915: Don't disable interrupts on PREEMPT_RT during atomic updates
aaff26226481 drm/i915: Don't check for atomic context on PREEMPT_RT
ee6e450fbfc1 drm/i915: Disable tracing points on PREEMPT_RT
7ad03555be9e drm/i915/gt: Use spin_lock_irq() instead of local_irq_disable() + spin_lock()
55bb5062738c drm/i915: Drop the irqs_disabled() check
920f664b8d7b drm/i915/guc: Consider also RCU depth in busy loop.
f92553ce51ee Revert "drm/i915: Depend on !PREEMPT_RT."
aa334412d84f arm: Disable jump-label on PREEMPT_RT.
b80d79ad9c95 ARM: enable irq in translation/section permission fault handlers
9a4b05f4abe9 arm: Disable FAST_GUP on PREEMPT_RT if HIGHPTE is also enabled.
3d9c8e74c1fc ARM: Allow to enable RT
c22fc883cd84 powerpc/pseries/iommu: Use a locallock instead local_ir           )
96edfd863ffa powerpc/pseries: Select the generic memory allocato
d01c57fb0df5 powerpc/kvm: Disable in-kernel MPIC emulation f           T_RT
b35d95e74643 powerpc/stackprotector: work around stack          t from atomic
728e044a97a3 POWERPC: Allow to enable RT
527711180912 sysfs: Add /sys/kernel/realtime entry
```

Is the kernel RT?

Current Stable RT

Version	Maintainer	EOL
6.13	Sebastian A. Siewior	Maintainer
6.6	Clark Williams	Dec 2026
6.1	Clark Williams	Dec 2026
5.15	Joseph Salisbury	Oct 2026
5.10	Luis Claudio R. Goncalves	Dec 2026
5.4	Tom Zanussi	Dec 2025
4.19	Daniel Wagner	Dec 2024

Status of Linux Real-Time

IT'S DONE!

Status of Linux Real-Time

IT'S DONE!

Not really, but almost!

What’s more to do?

- Growing pains

What's more to do?

- Growing pains
    - The dog that finally caught the car!

What’s more to do?

- Growing pains
  - The dog that finally caught the car!
- How to handle bugs that are only for RT?
  - No MAINTAINERS file
  - Who do the users contact? linux-rt-users@vger.kernel.org

ELISA Enabling Linux in Safety Applications WORKSHOP

What's more to do?

- Growing pains
  - The dog that finally caught the car!
- How to handle bugs that are only for RT?
  - No MAINTAINERS file
  - Who do the users contact? linux-rt-users@vger.kernel.org
- New mailing list: linux-rt-devel@vger.kernel.org

What’s more to do?

- Growing pains
    - The dog that finally caught the car!
- How to handle bugs that are only for RT?
    - No MAINTAINERS file
    - Who do the users contact? linux-rt-users@vger.kernel.org
- New mailing list: linux-rt-devel@vger.kernel.org
- Documentation, documentation, documentation!!!!

Questions?

Licensing of Workshop Results

All work created during the workshop is licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) [https://creativecommons.org/licenses/by/4.0/] by default, or under another suitable open-source license, e.g., GPL-2.0 for kernel code contributions.

You are free to:

- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

ELISA Enabling Linux in Safety Applications

WORKSHOP