`rtprio(2)` and POSIX(.1b) Priorities, and Their FreeBSD Implementation: A Deep Dive (and Sweep)

Olivier Certner

Kumacom SARL

AsiaBSDCon 2024

Who am I?

Olivier Certner

- 
- CS professional for ~20 years
- Worked in the CAD and finance sectors
- CTO of small startups
- PhD in many-core parallel programming models
- Language expert and polyglot
  - Notably, C, Common Lisp, Ada and C++

Why am I here?

BSD use

- FreeBSD user since 2004
- Using it everywhere
- Maintaining private changes (ports, userland, kernel).

Community involvement

- Sporadic bug reports and mails on lists (since ~20 years)
- Gradual increase in involvement since ~4 years
  - Maintaining a few ports
  - Reporting bugs in base and submitting patches
  - Contractor for the FreeBSD Foundation since 2023/09
  - Committer since ~4 months (`olce@`)

Project Goals

Rationalize scheduling priorities

- Fix scheduling APIs bugs
  - Behavior
  - Security
- Decouple implementation and interfaces
- Better POSIX compliance
  - In effect or in spirit
  - Except when poor or non-sensical
- Extend usefulness
  - Make timesharing's priority levels useful
  - Improve priority reporting

Paper content

- Provide an API reference for `rtprio(2)` and POSIX(.1b)
  - History of POSIX standardization (of scheduling)
- Mention differing platform's behaviors
- Why did I get into that?
- Expose old and new design choices
- Report on progress

This talk

1. Scheduling policies
   - Background
   - `rtprio(2)`
   - POSIX(.1b)

This talk

1. Scheduling policies
   - Background
   - `rtprio(2)`
   - POSIX(.1b)
2. Impacting changes
   - Timesharing priority levels
   - Privilege checks
   - More queues for schedulers

Background

Scheduling

Decide

- Which runnable thread
- Runs on which CPU/core
- At which moment
- For which duration

Priority-based models

Common alternatives

- Multi-level queue
  - Fixed-level assignment
- Multi-level feedback queue
  - Dynamic level changes
  - Based on behavior

FreeBSD internal model

ULE scheduler's classes

- Realtime
  - Multi-level queue
  - Includes:
    - Interrupt (kernel) threads
    - Realtime user threads
    - Regular kernel threads
- Timeshare
  - Multi-level feedback queue
  - With a twist: "circular" queue
- Idletime
  - Multi-level queue

rtprio(2)

API overview

System calls

`rtprio(2)`	Operate on some process (PID) or the current thread (0)
`rtprio_thread(2)`	Operate on some thread (TID)

Modes

`RTP_LOOKUP`	Retrieve settings
`RTP_SET`	Set settings

Settings

type	Scheduling type/class
prio	Priority level within the class

- Higher number means lower priority

Scheduling types/classes 1

From highest to lowest priority:

- Interrupt threads

  type `RTP_PRIO_ITHD`
  prio ?

- Realtime user threads

  type `RTP_PRIO_FIFO` or `RTP_PRIO_REALTIME`
  prio 0–31 with a caveat...

- Regular kernel threads

  type `RTP_PRIO_NORMAL`
  prio 0

Scheduling types/classes 1

From highest to lowest priority:

- Interrupt threads

  type `RTP_PRIO_ITHD`
  prio Implementation dependent

- Realtime user threads

  type `RTP_PRIO_FIFO` or `RTP_PRIO_REALTIME`
  prio 0–31

- Regular kernel threads

  type `RTP_PRIO_KERNEL`
  prio Implementation dependent

Scheduling types/classes 2

From highest to lowest priority:

- Timesharing threads

  type `RTP_PRIO_NORMAL`
  prio Implementation dependent... and varying!

- Idle threads

  type `RTP_PRIO_IDLE`
  prio 0–31 with a caveat...

Scheduling types/classes 2

From highest to lowest priority:

- Timesharing threads

  type `RTP_PRIO_NORMAL`
  prio 0–40

- Idle threads

  type `RTP_PRIO_IDLE`
  prio 0–31

POSIX(.1b)

The standard

Differences to `rtprio(2)`

- Priority levels "reversed"
- Non-negative priority numbers
- Absolute priority scale
- Process vs. thread scheduling settings
  - Effect depends on scheduling contention scope

Reality check 1

- Priority levels “reversed”

Reality check 1

- Priority levels "reversed"

  But not on:

  - HP-UX

Reality check 1

- Priority levels "reversed"

  But not on:

  - HP-UX
- Non-negative priority numbers

Reality check 1

- Priority levels "reversed"

  But not on:

  - HP-UX
- Non-negative priority numbers

  But not on:

  - illumos

Reality check 2

- Absolute priority scale

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD
  - OpenBSD

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD
  - OpenBSD
  - NetBSD

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD
  - OpenBSD
  - NetBSD
  - illumos

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD
  - OpenBSD
  - NetBSD
  - illumos
  - Linux

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD
  - OpenBSD
  - NetBSD
  - illumos
  - Linux
- Process vs. thread scheduling settings
  - Support for system contention scope only
  - Process settings should have no effect

Reality check 2

- Absolute priority scale

  Well, supposedly... but not on:

  - FreeBSD
  - OpenBSD
  - NetBSD
  - illumos
  - Linux
- Process vs. thread scheduling settings
  - Support for system contention scope only
  - Process settings should have no effect
  - But, in surveyed variants, they are mapped to:
    - Either the "main" thread, or the calling thread
    - Or all process' threads

Outline

Timesharing priority levels

Dynamic levels

```
$ ./set_rtprio 0 NORMAL 10
Current priority:  0.
RT prio:  Type:  RTP_PRIO_NORMAL, prio:  10.
```

Dynamic levels

```
$ ./set_rtprio 0 NORMAL 10
Current priority:  0.
RT prio:  Type:  RTP_PRIO_NORMAL, prio:  10.
$ ./prio
Current priority:  0.
RT prio:  Type:  RTP_PRIO_NORMAL, prio:  0.
```

Dynamic levels

```
$ ./set_rtprio 0 NORMAL 10
Current priority:  0.
RT prio:  Type:  RTP_PRIO_NORMAL, prio:  10.
$ ./prio
Current priority:  0.
RT prio:  Type:  RTP_PRIO_NORMAL, prio:  0.
$ ./prio
Current priority:  0.
RT prio:  Type:  RTP_PRIO_NORMAL, prio:  1.
```

Other problems

- Tied to internal priority levels
  - Actual range: 0 – `PRI_MAX_TIMESHARE-PRI_MIN_TIMESHARE`
  - Changed in FreeBSD 14, may change again.
- `rtprio(2)` and POSIX(.1b) inconsistency
  - `rtprio(2)` tries to set the internal priority
  - POSIX(.1b) just completely ignores the passed level

Switch to fixed values

- But which ones?

Switch to fixed values

- But which ones?
- Readily available: Nice values

Switch to fixed values

- But which ones?
- Readily available: Nice values
- Need to be per-thread

Switch to fixed values

- But which ones?
- Readily available: Nice values
- Need to be per-thread
- POSIX compliance?
  - Absolute priority levels
  - Mapping to a process' nice value

Privilege checks

MAC framework reminder

- Fine-grained privileges
  - Constants starting with `PRIV_`
  - Some examples: `PRIV_VFS_READ`, `PRIV_VFS_WRITE`
- Root has all privileges
  - Except in jails
- See full API description at `priv(9)`
  - `priv_check_cred()`
  - `priv_check()`

Scheduling privileges

Initial list

- PRIV_SCHED_SETPRIORITY
- PRIV_SCHED_RTPRIO
- PRIV_SCHED_IDPRIO
- PRIV_SCHED_SETPOLICY
- PRIV_SCHED_SET
- PRIV_SCHED_SETPARAM

Scheduling privileges

New list

- PRIV_SCHED_SETPRIORITY, PRIV_SCHED_RAISEPRIO
- PRIV_SCHED_RTPRIO
- PRIV_SCHED_IDPRIO
- PRIV_SCHED_SETPOLICY
- PRIV_SCHED_SET
- PRIV_SCHED_SETPARAM

mac_priority(4)

- Users in group `realtime` can use realtime classes
  - Group grants `PRIV_SCHED_RTPRIO` and `PRIV_SCHED_SETPOLICY`
  - Will soon also grant `PRIV_SCHED_RAISEPRIO`
- Users in group `idletime` can use the idletime class
  - Group grants `PRIV_SCHED_IDPRIO`

More queues for schedulers

POSIX XSH 2.8.4

Conforming implementations shall provide a priority range of at least 32 priorities for this policy.

(In both the `SCHED_FIFO` and `SCHED_RR` sections.)

Complying?

- `RTP_PRIO_MIN` is 0

Complying?

- `RTP_PRIO_MIN` is 0
- `RTP_PRIO_MAX` is 31

Complying?

- `RTP_PRIO_MIN` is 0
- `RTP_PRIO_MAX` is 31
- New static assertion

Complying?

- `RTP_PRIO_MIN` is 0
- `RTP_PRIO_MAX` is 31
- New static assertion
- What could possibly go wrong?

Priority levels conflation

Multi-level queue

- Has only 64 distinct levels
- Priority P mapped to queue number P mod 64
  - 4 priorities per queue
- All threads of a single level not treated differently
- 1-to-1 mapping to internal levels...

Priority levels conflation

Multi-level queue

- Has only 64 distinct levels
- Priority P mapped to queue number P mod 64
  - 4 priorities per queue
- All threads of a single level not treated differently
- 1-to-1 mapping to internal levels...

ULE

- Actually uses 1 queue per internal class
- Spreads out timesharing processes
  - Internal range currently is 136–223 (88 values)
  - Mapped to 64 queues
- Queries each queue in turn

One queue per level

- Stop playing unnecessary tricks

One queue per level

- Stop playing unnecessary tricks
- Make ULE's policy really work as intended

One queue per level

- Stop playing unnecessary tricks
- Make ULE's policy really work as intended
- Allow giving even more levels to timesharing

One queue per level

- Stop playing unnecessary tricks
- Make ULE's policy really work as intended
- Allow giving even more levels to timesharing
- Ensure real 32 levels for realtime

Other achievements

- Factorize and fix priority translation
- Kernel drives almost everything
- Align `rtprio(2)` and POSIX(.1b) interfaces
  - `SCHED_IDLE`
  - Easy to add more
- Linuxulator included

Work in progress

- Switch `RTP_PRIO_NORMAL` levels as nice values
  - Implies a nice value per thread
  - Possibly the same for `SCHED_OTHER`
- Adoption of `SCHED_BATCH`
- Reporting based on `rtprio(2)`
  - `ps(1)`
  - `top(1)`

Possible future work

- Query `rtprio(2)` priority ranges
  - Smoother evolutions
- Absolute priority levels (POSIX)
- Per-process priority limit
  - Unprivileged users could raise to it
  - May obsolete `mac_priority(4)`
- Runaway processes mitigations
  - Downgrade `SCHED_FIFO` threads to `SCHED_RR` by default
  - Allocate time slots to threads in lower priority classes

Code status

- For now, only minor stuff committed
- Bulk needs external reviews
- 256 queues still a WIP
- Some other WIP not published yet

`https://github.com/OlCe2/freebsd-src`
Branch: `oc-rtprio_sched`

Thanks

To you

Thanks

To you

And the FreeBSD Foundation

The end!

Questions?

Thoughts?