INTRODUCING

STORAGE INSTANTIATION DAEMON

PETER RAJNOHA <PRAJNOHA@REDHAT.COM>
DEVCONF.CZ, JANUARY 27 2019, BRNO
UEVENTS

OVERVIEW
UEVENTS
OVERVIEW

USER SPACE

KERNEL SPACE
UEVENTS
OVERVIEW

USER SPACE

KERNEL SPACE

NETLINK
UEVENTS
OVERVIEW
UEVENTS
OVERVIEW
UEVENTS
OVERVIEW
UEVENTS
UEVENTS

- uevents are event notifications that userspace can monitor
UEVENTS

- uevents are **event notifications** that userspace can monitor
- both kernel and userspace can cause **uevents** to get generated
  - *kernel multicast* uevents
    - genuine
    - synthesized (writing to /sys/.../uevent file)
  - *userspace multicast* uevents
  - *userspace unicast* uevents
UEVENTS

- uevents are event notifications that userspace can monitor
- both kernel and userspace can cause uevents to get generated
  - kernel multicast uevents
    - genuine
    - synthesized (writing to /sys/.../uevent file)
  - userspace multicast uevents
  - userspace unicast uevents
- uevent environment in KEY=VALUE text format
  - ACTION, DEVPATH, SUBSYSTEM, SEQNUM
  - more variables added by driver core, subsystems, drivers...
**UEVENTS**

- uevents are *event notifications* that userspace can monitor
- both kernel and userspace can cause uevents to get generated
  - *kernel multicast* uevents
    - genuine
    - synthesized (writing to /sys/.../uevent file)
  - *userspace multicast* uevents
  - *userspace unicast* uevents
- uevent *environment in KEY=VALUE text format*
  - ACTION, DEVPATH, SUBSYSTEM, SEQNUM
  - more variables added by driver core, subsystems, drivers...
- **8 uevent action types:**
  - ADD, CHANGE, REMOVE, MOVE
  - ONLINE, OFFLINE, BIND, UNBIND
**UEVENTS**

- uevents are *event notifications* that userspace can monitor
- both kernel and userspace can cause uevents to get generated
  - `kernel multicast` uevents
    - genuine
    - synthesized (writing to `/sys/.../uevent file`)
  - `userspace multicast` uevents
  - `userspace unicast` uevents
- uevent environment in `KEY=VALUE` text format
  - ACTION, DEVPATH, SUBSYSTEM, SEQNUM
  - more variables added by driver core, subsystems, drivers...
- 8 uevent action types:
  - ADD, CHANGE, REMOVE, MOVE
  - ONLINE, OFFLINE, BIND, UNBIND
- all uevents sent through *netlink socket*
UDEV
UDEV

- udev daemon in userspace to support dynamic device management
UDEV

- udev daemon in userspace to support dynamic device management
- monitoring netlink socket for uevents (*kernel* uevent type)
UDEV

- udev daemon in userspace to support dynamic device management
- monitoring netlink socket for uevents (kernel uevent type)
- processing udev rules
  - key=value matching/writing
  - sysfs property matching/writing
  - sysctl parameter matching/writing
  - tag matching/creation
  - executing builtin or external commands, collecting output
  - setting device node permissions
  - creating symlinks to device nodes
UDEV

- **udev daemon** in userspace to support dynamic device management
- **monitoring** netlink socket for uevents (*kernel* uevent type)
- **processing** udev rules
  - key=value matching/writing
  - sysfs property matching/writing
  - sysctl parameter matching/writing
  - tag matching/creation
  - executing builtin or external commands, collecting output
- **setting device node permissions**
- **creating symlinks to device nodes**

- **storing records in udev database**
  - records per device
  - subset of key=value environment sent with uevent
  - key=value pairs added by rules
UDEV

- **udev daemon** in userspace to support dynamic device management
- **monitoring** netlink socket for uevents (**kernel** uevent type)
- **processing** udev rules
  - key=value matching/writing
  - sysfs property matching/writing
  - sysctl parameter matching/writing
  - tag matching/creation
  - executing builtin or external commands, collecting output
  - **setting device node permissions**
  - **creating symlinks to device nodes**
- **storing** records in udev database
  - records per device
  - subset of key=value environment sent with uevent
  - key=value pairs added by rules
- **regenerating uevents** including key=value pairs resulted from udev rule processing (**udev** uevent type)
UDEV

- **udev daemon** in userspace to support dynamic device management
- **monitoring** netlink socket for uevents *(kernel uevent type)*
- **processing** udev rules
  - key=value matching/writing
  - sysfs property matching/writing
  - sysctl parameter matching/writing
  - tag matching/creation
  - executing builtin or external commands, collecting output
  - **setting device node permissions**
  - **creating symlinks to device nodes**
- **storing** records in udev database
  - records per device
  - subset of key=value environment sent with uevent
  - key=value pairs added by rules
- **regenerating uevents** including key=value pairs resulted from udev rule processing *(udev uevent type)*
- **others able to monitor** kernel and/or udev uevents
UEVENTS + UDEV
UEVENTS + UDEV

USER SPACE

KERNEL SPACE
UEVENTS + UDEV
UEVENTS + UDEV
UEVENTS + UDEV

User Space
/sys/.../uevent

Kernel Space
synthesized or genuine kernel uevent

UDEV

NETLINK
UEVENTS + UDEV

Diagram showing the relationship between UDEV and UDEVD, with USER SPACE, USER SPACE -> UDEV, UDEV -> UDEV WORKER, and KERNEL SPACE with /sys/.../uevent and NETLINK.
UEVENTS + UDEV
UEVENTS + UDEV

EXTERNAL COMMANDS

UDEV WORKER

BUILTIN COMMANDS
UDEV RULES

UDEV

USER SPACE

KERNEL SPACE

UBLIQUE COMMANDS
UDEV RULES

/\sys/.../uevent

NETLINK
UEVENTS + UDEV

Kernel Space

User Space

/sys/.../uevent

Netlink

Kernel Space

Udevd Worker

Udev DB
/run/udev

Udevd
UEVENTS + UDEV
STORAGE SPECIFICS
STORAGE SPECIFICS

- the ideal: one single-level device usable after ADD uevent
STORAGE SPECIFICS

- the ideal: one single-level *device usable after ADD uevent*
- the reality: *device usable after further actions*
  - initialization sequence
  - multistep activation scheme
  - grouping
  - layering
STORAGE SPECIFICS

- **the ideal:** one single-level device usable after `ADD` uevent
- **the reality:** device usable after further actions
  - initialization sequence
  - multistep activation scheme
  - grouping
  - layering
- **devices may contain** signatures/metadata/external configuration that define the next layer in the stack
  - `blkid` scan for the majority
  - `multipath -c` to detect multipath components
  - detached header location for LUKS encrypted devices
  - further additional scans by various subsystems
PROBLEMS WITH UDEV
WHILE HANDLING STORAGE DEVICES
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- **overloaded uevent** action type - just a *CHANGE* for lots of notifications
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a *CHANGE* for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded `uevent` action type - just a `CHANGE` for lots of notifications
- restricted `udev` rule language
- calling `external commands` to make (even simple) decisions
- all rules and keys are `global`, any rule can overwrite values for various keys
- `accessing udev database` from `udev rules` is clunky and error-prone
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a *CHANGE* for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a *CHANGE* for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
- udevd worker process timeout causes the process to get killed without further fallback
PROBLEMS WITH UDEV
WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/ usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
- udevd worker process timeout causes the process to get killed without further fallback
- scheduling separate work requires complex synchronization scheme
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a *CHANGE* for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as *ready/usable, public, private, temporarily private*
- amount of work done within udevd context may not be appropriate
- udevd worker process *timeout* causes the process to get killed without further fallback
- scheduling separate work requires complex *synchronization* scheme

**UDEV IS NOT PRIMARILY DESIGNED FOR THIS!**
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a *CHANGE* for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
- udevd worker process *timeout* causes the process to get killed without further fallback
- scheduling separate work requires complex *synchronization* scheme

**UDEV IS NOT PRIMARILY DESIGNED FOR THIS!**

**IT'S DESIGNED TO HANDLE NODES AND SYMLINKS IN */DEV AND THEIR PERMISSIONS**
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
- udevd worker process timeout causes the process to get killed without further fallback
- scheduling separate work requires complex synchronization scheme

UDEV IS NOT PRIMARILY DESIGNED FOR THIS!
IT’S DESIGNED TO HANDLE NODES AND SYMLINKS IN /DEV AND THEIR PERMISSIONS
WHICH IT DOES JUST FINE
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
- udevd worker process timeout causes the process to get killed without further fallback
- scheduling separate work requires complex synchronization scheme

UDEV IS NOT PRIMARILY DESIGNED FOR THIS!
IT'S DESIGNED TO HANDLE NODES AND SYMLINKS IN /DEV AND THEIR PERMISSIONS
WHICH IT DOES JUST FINE
WE NEED A BIT DIFFERENT APPROACH HERE FOR OUR NEEDS!
PROBLEMS WITH UDEV WHILE HANDLING STORAGE DEVICES

- overloaded uevent action type - just a CHANGE for lots of notifications
- restricted udev rule language
- calling external commands to make (even simple) decisions
- all rules and keys are global, any rule can overwrite values for various keys
- accessing udev database from udev rules is clunky and error-prone
- problems with identification of current state
- no direct support for grouping
- no standard on marking device as ready/usable, public, private, temporarily private
- amount of work done within udevd context may not be appropriate
- udevd worker process timeout causes the process to get killed without further fallback
- scheduling separate work requires complex synchronization scheme

UDEV IS NOT PRIMARILY DESIGNED FOR THIS!
IT'S DESIGNED TO HANDLE NODES AND SYMLINKS IN /DEV AND THEIR PERMISSIONS
WHICH IT DOES JUST FINE
WE NEED A BIT DIFFERENT APPROACH HERE FOR OUR NEEDS!
CHANGES

KERNEL SPACE
synthesized or genuine kernel uevent

USER SPACE
/sys/.../uevent

EXTERNAL COMMANDS

UDEV WORKER
BUILTIN COMMANDS
UDEV RULES

1:1 uevent
UDEV

NETWORK

NETLINK

UDEV DB
/run/udev

EXTERNAL COMMANDS

CHANGES

KERNEL SPACE

/sys/.../uevent

synthesized or genuine kernel uevent

USER SPACE

/.../uevent

EXTERNAL COMMANDS

UEDVD WORKER

BUILTIN COMMANDS

UEDV RULES

1:1 uevent

UEDVD

UDEV DB
/run/udev

NETLINK

EXTERNAL COMMANDS
CHANGES

EXTERNAL COMMANDS

KERNEL SPACE
synthesized or genuine kernel uevent

USER SPACE
/sys/.../uevent

NETLINK

UDEV WORKER

BUILTIN COMMANDS
UDEV RULES

UDEVD

UDEV DB
/run/udev

1:1 uevent

EXTERNAL COMMANDS
CHANGES

EXTERNAL COMMANDS

UDEV WORKER

BUILTIN COMMANDS
UDEV RULES

UDEV DB
/run/udev

1:1 uevent

UDEV

NETLINK

USER SPACE

KERNEL SPACE

/sys/.../uevent

synthesized or genuine kernel uevent
CHANGES

KERNEL SPACE

KERNEL SPACE

synthesized or genuine kernel uevent

USER SPACE

/user/.../uevent

SYSTABLE

UDEV WORKER

BUILTIN COMMANDS

UDEV RULES

UDEV DB

/run/udev

1:1 uevent

UDEV

NETLINK

SID
CHANGES
CHANGES

- synthesized or genuine kernel uevent
- /sys/.../uevent
- /run/udev
- UDEV DB
- UDEV WORKER
- UDEVD
- built-in commands
- UDEV RULES
- 1:1 uevent
- sid
- udev uevent
- SID
- USER SPACE
- KERNEL SPACE
CHANGES

KERNEL SPACE

USER SPACE

synthesized or genuine kernel uevent

UDDEV WORKER

BUILTIN COMMANDS

UDDEV RULES

SID

uevent

UDEV DB

/run/udev

UDEV WORKER

UDEV RULES

1:1 uevent

UDEVDB

UDEV RULES

BUILTIN COMMANDS

UEDEV
CHANGES

KERNEL SPACE

USER SPACE

KERNEL SPACE

ACTION UUID KEY=VALUE ...

synthesized or genuine kernel uevent

CHANGES
CHANGES
STORAGE INSTANTIATION
DAEMON AND COMPONENTS
STORAGE INSTANTIATION
DAEMON AND COMPONENTS

- *sid daemon*
  - layered on top of udev
  - executes storage-specific uevent handling and processing
  - keeps its own database
STORAGE INSTANTIATION

DAEMON AND COMPONENTS

- **sid daemon**
  - layered on top of udev
  - executes storage-specific uevent handling and processing
  - keeps its own database

- **udev builtin command**
  - bridge between udev and SID with subcommands:
    - **sid active**
      - returns active, inactive, incompatible
    - **sid identify**
      - relays uevent with environment to SID
      - requests execution of identification and related routines
      - returns KEY=VALUE results for use in udev rules or to store in udev db
    - **sid checkpoint** `<checkpoint_name> [<key>]` ...
    - **sid version**
STORAGE INITIATION
DAEMON AND COMPONENTS

- **sid daemon**
  - layered on top of udev
  - executes storage-specific uevent handling and processing
  - keeps its own database

- **udev built-in command**
  - bridge between udev and SID with subcommands:
    - **sid active**
      - returns active, inactive, incompatible
    - **sid identify**
      - relays uevent with environment to SID
      - requests execution of identification and related routines
      - returns KEY=VALUE results for use in udev rules or to store in udev db
    - **sid checkpoint** `<checkpoint_name> [ <key> ... ]`
    - **sid version**

- **library interface**
  - access SID's information store
  - subscribe to SID notifications
STORAGE INSTANTIATION
DAEMON AND COMPONENTS

- **sid daemon**
  - layered on top of udev
  - executes storage-specific uevent handling and processing
  - keeps its own database

- **udev builtin command**
  - bridge between udev and SID with subcommands:
    - **sid active**
      - returns active, inactive, incompatible
    - **sid identify**
      - relays uevent with environment to SID
      - requests execution of identification and related routines
      - returns KEY=VALUE results for use in udev rules or to store in udev db
    - **sid checkpoint** `<checkpoint_name> [ <key> ... ]`
    - **sid version**

- **library interface**
  - access SID's information store
  - subscribe to SID notifications

- **sidctl command line interface**
  - control and access SID and its information store
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES

SID

DB

SID

UDEV worker
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES

SID

DB

S I D

UDEV WORKER
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES
STORAGE INSTANTIATION
DAEMON IDENTIFY STAGES

SID WORKER
SID DB snapshot
SID

SID DB snapshot
SID

UDEVDD WORKER
sid identify

uevent

db snapshot
SID DB
SID
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES

STAGE "A"

SID WORKER

SID DB
snapshot

db
snapshot

SID

DB

UDEVd WORKER

sid
identify

uevent
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES

STAGE "A"

db snapshot
SID DB snapshot
SID WORKER
SID

S I D  D B
S I D  W O R K E R

sid identify
UDEVD WORKER

uevent
STORAGE INSTANTIATION
DAEMON IDENTIFY STAGES

STAGE "A"

SID WORKER

SID DB snapshot

db snapshot

SID DB

SID

UDEVVD WORKER

sid identify

sid checkpoint

uevent
STORAGE INSTANTIATION
DAEMON IDENTIFY STAGES

STAGE "A"

SID WORKER

SID DB snapshot

db snapshot

SID DB

db master sync

db write

UDEV WORKER

sid identify

sid checkpoint

uevent
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES

STAGE "A"

SID WORKER

SID DB snapshot

db snapshot

SID DB

sid identify

sid checkpoint

UDEVWD WORKER

db master sync

db write

SID DB

udevent

uevent
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "A"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
IDENTIFY - STAGE "B"
SID DAEMON
DATABASE
SID DAEMON

DATABASE

- key-value (KV) database with various backends
SID DAEMON DATABASE

- key-value (KV) database with various backends
- value types
  - simple
  - vector
SID DAEMON DATABASE

- **key-value (KV) database** with various backends
- **value types**
  - simple
  - vector
- **snapshot separation**
SID DAEMON DATABASE

- key-value (KV) database with various backends
- value types
  - simple
  - vector
- snapshot separation
- delta synchronization of vector values
SID DAEMON
DATABASE

- **key-value (KV) database with various backends**
- **value types**
  - simple
  - vector
- **snapshot separation**
- **delta synchronization of vector values**
- **separate key namespaces**
  - `KV_NS_UDEV` (import/export from/to udev)
  - `KV_NS_GLOBAL` (visible globally)
  - `KV_NS_MODULE` (visible only in specific module)
  - `KV_NS_DEVICE` (visible only when processing specific device)
SID DAEMON
DATABASE

- **key-value (KV) database with various backends**
- **value types**
  - simple
  - vector
- **snapshot separation**
- **delta synchronization of vector values**
- **separate key namespaces**
  - `KV_NS_UDEV` (import/export from/to udev)
  - `KV_NS_GLOBAL` (visible globally)
  - `KV_NS_MODULE` (visible only in specific module)
  - `KV_NS_DEVICE` (visible only when processing specific device)
- **per-module protection flags**
  - `KV_PROTECTED` (originating module can read-write, others read-only)
  - `KV_PRIVATE` (originating module can read-write, others unable to access)
  - `KV_RESERVED` (originating module reserves, others can't take over)
SID DAEMON DATABASE

- **key-value** (KV) database with various backends
- **value types**
  - simple
  - vector
- **snapshot** separation
- **delta synchronization** of vector values
- separate **key namespaces**
  - `KV_NS_UDEV` (import/export from/to udev)
  - `KV_NS_GLOBAL` (visible globally)
  - `KV_NS_MODULE` (visible only in specific module)
  - `KV_NS_DEVICE` (visible only when processing specific device)
- **per-module** protection flags
  - `KV_PROTECTED` (originating module can read-write, others read-only)
  - `KV_PRIVATE` (originating module can read-write, others unable to access)
  - `KV_RESERVED` (originating module reserves, others can't take over)
- **persistence**
  - `KV_PERSISTENT` (persist record for next use)
QUESTIONS ?
**github:** https://github.com/prajnoha/sid

**freenode:** prajnoha on #lvm

**email:** prajnoha@redhat.com
THANK YOU!

redhat®