Block layer support for Persistent Reservations

The Linux kernel supports a user space interface for simplified Persistent Reservations which map to block devices that support these (like SCSI). Persistent Reservations allow restricting access to block devices to specific initiators in a shared storage setup.

This document gives a general overview of the support ioctl commands. For a more detailed reference please refer to the SCSI Primary Commands standard, specifically the section on Reservations and the "PERSISTENT RESERVE IN" and "PERSISTENT RESERVE OUT" commands.

All implementations are expected to ensure the reservations survive a power loss and cover all connections in a multi path environment. These behaviors are optional in SPC but will be automatically applied by Linux.

The following types of reservations are supported:

- PR_WRITE_EXCLUSIVE
  Only the initiator that owns the reservation can write to the device. Any initiator can read from the device.
- PR_EXCLUSIVE_ACCESS
  Only the initiator that owns the reservation can access the device.
- PR_WRITE_EXCLUSIVE_REG_ONLY
  Only initiators with a registered key can write to the device, Any initiator can read from the device.
- PR_EXCLUSIVE_ACCESS_REG_ONLY
  Only initiators with a registered key can access the device.
- PR_WRITE_EXCLUSIVE_ALL_REGS

  Only initiators with a registered key can write to the device, Any initiator can read from the device. All initiators with a registered key are considered reservation holders. Please reference the SPC spec on the meaning of a reservation holder if you want to use this type.

- PR_EXCLUSIVE_ACCESS_ALL_REGS
  Only initiators with a registered key can access the device. All initiators with a registered key are considered reservation holders. Please reference the SPC spec on the meaning of a reservation holder if you want to use this type.

The following ioctl are supported:

1. IOC_PR_REGISTER

This ioctl command registers a new reservation if the new_key argument is non-null. If no existing reservation exists old_key must be zero, if an existing reservation should be replaced old_key must contain the old reservation key.

If the new_key argument is 0 it unregisters the existing reservation passed in old_key.

2. IOC_PR_RESERVE

This ioctl command reserves the device and thus restricts access for other devices based on the type argument. The key argument must be the existing reservation key for the device as acquired by the IOC_PR_REGISTER, IOC_PR_REGISTER_IGNORE, IOC_PR_PREEMPT or IOC_PR_PREEMPT_ABORT commands.

3. IOC_PR_RELEASE

This ioctl command releases the reservation specified by key and flags and thus removes any access restriction implied by it.

4. IOC_PR_PREEMPT

This ioctl command releases the existing reservation referred to by old_key and replaces it with a new reservation of type for the reservation key new_key.

5. IOC_PR_PREEMPT_ABORT

This ioctl command works like IOC_PR_PREEMPT except that it also aborts any outstanding command sent over a connection identified by old_key.

6. IOC_PR_CLEAR

This ioctl command unregisters both key and any other reservation key registered with the device and drops any existing reservation.

Flags

All the ioctls have a flag field. Currently only one flag is supported:

- PR_FL_IGNORE_KEY
  Ignore the existing reservation key. This is commonly supported for IOC_PR_REGISTER, and some implementation may support the flag for IOC_PR_RESERVE.

For all unknown flags the kernel will return -EOPNOTSUPP.