NAME

pass - CAM application passthrough driver

SYNOPSIS

device pass

DESCRIPTION

The **pass** driver provides a way for userland applications to issue CAM CCBs to the kernel.

Since the **pass** driver allows direct access to the CAM subsystem, system administrators should exercise caution when granting access to this driver. If used improperly, this driver can allow userland applications to crash a machine or cause data loss.

The **pass** driver attaches to every SCSI and ATA device found in the system. Since it attaches to every device, it provides a generic means of accessing SCSI and ATA devices, and allows the user to access devices which have no "standard" peripheral driver associated with them.

KERNEL CONFIGURATION

It is only necessary to configure one **pass** device in the kernel; **pass** devices are automatically allocated as SCSI and ATA devices are found.

IOCTLS

CAMIOCOMMAND union ccb *

This ioctl takes most kinds of CAM CCBs and passes them through to the CAM transport layer for action. Note that some CCB types are not allowed through the passthrough device, and must be sent through the xpt(4) device instead. Some examples of xpt-only CCBs are XPT_SCAN_BUS, XPT_DEV_MATCH, XPT_RESET_BUS, XPT_SCAN_LUN, XPT_ENG_INQ, and XPT_ENG_EXEC. These CCB types have various attributes that make it illogical or impossible to service them through the passthrough interface.

If the user would like the kernel to do error recovery, the CAM_PASS_ERR_RECOVER flag must be set on the CCB, and the retry_count field set to the number of retries.

CAMGETPASSTHRU union ccb *

This ioctl takes an XPT_GDEVLIST CCB, and returns the passthrough device corresponding to the device in question. Although this ioctl is available through the **pass** driver, it is of limited use, since the caller must already know that the device in question is a passthrough device if they are issuing this ioctl. It is probably more useful to issue this ioctl through the xpt(4) device.

CAMIOQUEUE union ccb *

Queue a CCB to the **pass** driver to be executed asynchronously. The caller may use select(2), poll(2) or kevent(2) to receive notification when the CCB has completed.

This ioctl takes most CAM CCBs, but some CCB types are not allowed through the pass device, and must be sent through the xpt(4) device instead. Some examples of xpt-only CCBs are XPT_SCAN_BUS, XPT_DEV_MATCH, XPT_RESET_BUS, XPT_SCAN_LUN, XPT_ENG_INQ, and XPT_ENG_EXEC. These CCB types have various attributes that make it illogical or impossible to service them through the passthrough interface.

Although the CAMIOQUEUE ioctl is not defined to take an argument, it does require a pointer to a union ccb. It is not defined to take an argument to avoid an extra malloc and copy inside the generic ioctl(2) handler.

The completed CCB will be returned via the CAMIOGET ioctl. An error will only be returned from the CAMIOQUEUE ioctl if there is an error allocating memory for the request or copying memory from userland. All other errors will be reported as standard CAM CCB status errors. Since the CCB is not copied back to the user process from the pass driver in the CAMIOQUEUE ioctl, the user's passed-in CCB will not be modified. This is the case even with immediate CCBs. Instead, the completed CCB must be retrieved via the CAMIOGET ioctl and the status examined.

Multiple CCBs may be queued via the CAMIOQUEUE ioctl at any given time, and they may complete in a different order than the order that they were submitted. The caller must take steps to identify CCBs that are queued and completed. The periph_priv structure inside struct ccb_hdr is available for userland use with the CAMIOQUEUE and CAMIOGET ioctls, and will be preserved across calls. Also, the periph_links linked list pointers inside struct ccb_hdr are available for userland use with the CAMIOQUEUE and CAMIOGET ioctls and will be preserved across calls.

If the user would like the kernel to do error recovery, the CAM_PASS_ERR_RECOVER flag must be set on the CCB, and the retry_count field set to the number of retries.

CAMIOGET union ccb *

Retrieve completed CAM CCBs queued via the CAMIOQUEUE ioctl. An error will only be returned from the CAMIOGET ioctl if the **pass** driver fails to copy data to the user process or if there are no completed CCBs available to retrieve. If no CCBs are available to retrieve, errno will be set to ENOENT.

All other errors will be reported as standard CAM CCB status errors.

Although the CAMIOGET ioctl is not defined to take an argument, it does require a pointer to a union ccb. It is not defined to take an argument to avoid an extra malloc and copy inside the generic ioctl(2) handler.

The pass driver will report via select(2), poll(2) or kevent(2) when a CCB has completed. One CCB may be retrieved per CAMIOGET call. CCBs may be returned in an order different than the order they were submitted. So the caller should use the periph_priv area inside the CCB header to store pointers to identifying information.

FILES

/dev/passn Character device nodes for the **pass** driver. There should be one of these for each device accessed through the CAM subsystem.

DIAGNOSTICS

None.

SEE ALSO

kqueue(2), poll(2), select(2), cam(3), cam_cdbparse(3), cam(4), cd(4), ctl(4), da(4), sa(4), xpt(4), camcontrol(8), camdd(8)

HISTORY

The CAM passthrough driver first appeared in FreeBSD 3.0.

AUTHORS

Kenneth Merry <*ken@FreeBSD.org*>