NAME

intro - introduction to devices and device drivers

DESCRIPTION

This section contains information related to devices, device drivers and miscellaneous hardware.

The device abstraction

Device is a term used mostly for hardware-related stuff that belongs to the system, like disks, printers, or a graphics display with its keyboard. There are also so-called *pseudo-devices* where a device driver emulates the behaviour of a device in software without any particular underlying hardware. A typical example for the latter class is */dev/mem*, a mechanism whereby the physical memory can be accessed using file access semantics.

The device abstraction generally provides a common set of system calls, which are dispatched to the corresponding device driver by the upper layers of the kernel. The set of system calls available for devices is chosen from open(2), close(2), read(2), write(2), ioctl(2), select(2), and mmap(2). Not all drivers implement all system calls; for example, calling mmap(2) on a keyboard device is not likely to be useful.

Aspects of the device abstraction have changed significantly in FreeBSD over the past two decades. The section *Historical Notes* describes some of the more important differences.

Accessing Devices

Most of the devices in FreeBSD are accessed through *device nodes*, sometimes also called *special files*. They are located within instances of the devfs(5) filesystem, which is conventionally mounted on the directory */dev* in the file system hierarchy (see also hier(7)).

The devfs(5) filesystem creates or removes device nodes automatically according to the physical hardware recognized as present at any given time. For pseudo-devices, device nodes may be created and removed dynamically as required, depending on the nature of the device.

Access restrictions to device nodes are usually subject to the regular file permissions of the device node entry, instead of being enforced directly by the drivers in the kernel. But since device nodes are not stored persistently between reboots, those file permissions are set at boot time from rules specified in devfs.conf(5), or dynamically according to rules defined in devfs.rules(5) or set using the devfs(8) command. In the latter case, different rules may be used to make different sets of devices visible within different instances of the devfs(5) filesystem, which may be used, for example, to prevent jailed subsystems from accessing unsafe devices. Manual changes to device node permissions may still be made, but will not persist.

Drivers without device nodes

Drivers for network devices do not use device nodes in order to be accessed. Their selection is based on other decisions inside the kernel, and instead of calling open(2), use of a network device is generally introduced by using the system call socket(2).

Configuring a driver into the kernel

For each kernel, there is a configuration file that is used as a base to select the facilities and drivers for that kernel, and to tune several options. See config(8) for a detailed description of the files involved. The individual manual pages in this section provide a sample line for the configuration file in their synopsis portions. See also the files */usr/src/sys/conf/NOTES* and */usr/src/sys/${ARCH}/conf/NOTES*.

Drivers need not be statically compiled into the kernel; they may also be loaded as modules, in which case any device nodes they provide will appear only after the module is loaded (and has attached to suitable hardware, if applicable).

Historical Notes

Prior to FreeBSD 6.0, device nodes could be created in the traditional way as persistent entries in the file system. While such entries can still be created, they no longer function to access devices.

Prior to FreeBSD 5.0, devices for disk and tape drives existed in two variants, known as *block* and *character* devices, or to use better terms, buffered and unbuffered (raw) devices. The traditional names are reflected by the letters "b" and "c" as the file type identification in the output of "ls -l". Raw devices were traditionally named with a prefix of "r", for example */dev/rda0* would denote the raw version of the disk whose buffered device was */dev/da0*. *This is no longer the case*; all disk devices are now "raw" in the traditional sense, even though they are not given "r" prefixes, and "buffered" devices no longer exist at all.

Buffered devices were accessed through a buffer cache maintained by the operating system; historically this was the system's primary disk cache, but in FreeBSD this was rendered obsolete by the introduction of unified virtual memory management. Buffered devices could be read or written at any byte position, with the buffer mechanism handling the reading and writing of disk blocks. In contrast, raw disk devices can be read or written only at positions and lengths that are multiples of the underlying device block size, and write(2) calls are *synchronous*, not returning to the caller until the data has been handed off to the device.

SEE ALSO

close(2), ioctl(2), mmap(2), open(2), read(2), select(2), socket(2), write(2), devfs(5), hier(7), config(8)

HISTORY

This manual page first appeared in FreeBSD 2.1.

AUTHORS

This man page has been rewritten by Andrew Gierth from an earlier version written by Jörg Wunsch with initial input by David E. O'Brien.