NAME

dev − device files in /dev

DESCRIPTION

Device files are the eyes and ears of the system. Through the device files one has access to the disks, terminals and other parts of the machine. Single bytes or disk blocks may be transferred to or from a device with ordinary **read**(2) or **write**(2) calls, byte positions set with **lseek**(2), or more complicated control functions performed with **ioctl(2).**

Device files as found in **/dev** have several attributes that must be considered. Here are two examples as **ls −l** shows them:

```
brw-rw-rw-  1 root     operator   2,   1 Jun 10  1995 fd1
crw--w----  1 kjb      tty        4,   0 May 11 09:41 console
```

Most attributes are the same as for a regular file and have the same function. The file type and the major and minor device numbers are special to devices.

Character devices are marked with a **c** as a file type letter. Any I/O on a character device is sent down to the device driver without any interpretation. This means that a process doing the I/O must know the characteristics of the device and deal with them appropriately.

Block devices provoke the file system server into buffering the data on those devices. Data read or written by processes is passed through the file system block cache. Unaligned bytes read or written are extracted or reassembled by the file server from or to whole blocks in the cache. The file server transfers data to or from the device driver as blocks to positions at block size boundaries. These blocks are Minix blocks of 1024 bytes, disk devices usually have a 512 byte block size. Only block devices can be mounted as part of the file system tree if they contain a Minix file system.

The major device number (2 for **fd1** and 4 for **console**) are used by FS to find the device driver that manages a device. The minor device number (1 for **fd1** and 0 for **console**) is passed to the driver to select a device among a number of related devices that are all managed by that driver. The device drivers are usually kernel tasks under Minix, small processes that are contained within the address space of the kernel. The following tasks and associated devices exist:

Memory (major 1)

The **ram**, **mem**, **kmem**, and **null** devices are managed by the memory task. The **ram** device is a block device for a chunk of memory that is the RAM disk. Any byte read from or written to the **ram** device is copied from or to that memory chunk. The **mem** device is a character device for the entire address space of the system, but **kmem** only for the kernel data area. These two devices allow programs like **ps**(1) to hunt around the system looking for interesting bits. The **null** device is a data sink. It happily swallows any bytes written to it, and returns nothing on a read.

Floppy disk (major 2)

The **fd0**, **fd0p0**, **fd0p1**, **fd0p2**, and **fd0p3** block devices are the first floppy disk and the four partitions that may exist on a that floppy disk. Likewise are **fd1** and **fd1p[0−3]** the device and partitions for the second floppy disk. The floppy disk devices are described in detail in **fd**(4). Partitioning in general is explained in **controller**(4).

Controller 0 (major 3)

The first hard disk on controller 0 can be accessed by block device **c0d0**. This device addresses the entire hard disk from the first to the last sector. A hard disk is normally partitioned in up to four primary partitions, **c0d0p0**, **c0d0p1**, **c0d0p2**, and **c0d0p3**. Each of these devices accesses a range of sectors on the **c0d0** device. It is customary to give each operating system on a disk a primary partition. So the Windows C: "drive" can be on **c0d0p0**, and Minix can be on **c0d0p1**. Minix wants to have several partitions on its own, so **c0d0p1** can be further subdivided into the subpartitions **c0d0p1s0**, **c0d0p1s1**, **c0d0p1s2**, and **c0d0p1s3**. **/dev** contains devices for the first and second hard disk (**c0d0** and **c0d1**), their primary partitions (**c0d[01]p[0−3]**) and subpartitions thereof (**c0d[01]p[0−3]s[0−3]**). More detail can be found in **controller**(4).

Terminals (minor 4)

The TTY driver manages the system console device, aptly named **console**, the serial lines, **tty00** and

tty01, and the pseudo ttys. Through the console device one can display characters on a screen attached to a monochrome, Hercules, color, or VGA adapter. The **ttyc1**, **ttyc2**, etc. devices are the so-called "virtual consoles" that share the one console display. One can select which virtual console is to be visible on the screen and take input from the keyboard. To allow remote login the devices with minor numbers of 128 or higher offer virtual terminals. These pseudo ttys come in tty, pty pairs that form a pipe between processes running under the tty, and a controlling process attached to the pty side. See also **console**(4), and **tty**(4).

Anonymous TTY (major 5)
This is just one device named **tty** that is a synonym for the controlling tty of a process. This device is not managed by any device driver, but is handled by FS itself. A process can get access to the terminal it is running under by using **/dev/tty**.

Line printer (major 6)
The **lp** device sends any bytes written to it to the printer.

TCP/IP (major 7)
The TCP/IP task is not a kernel task, but a server like MM and FS. It sits between FS and the DP8390 task that manages the ethernet boards. Together they implement the TCP/IP protocol. See also **ip**(4).

Controller 1 (major 8)
Like controller 0 (major 3), but managing a second controller with devices **/dev/c1***.

Controller 2 (major 10)
Like controller 0.

Controller 3 (major 12)
Like controller 0.

Audio (major 13)
The **audio** device can be used to produce or record air vibrations using a Soundblaster 16 type audio card. See **audio**(4).

Mixer (major 14)
The **mixer** device is used to control the audio driver.

FILES

/dev/* All Minix devices

SEE ALSO

read(2), **write**(2), **lseek**(2), **ioctl**(2), **console**(4), **fd**(4), **controller**(4), **ip**(4), **tty**(4), **MAKEDEV**(8).

DIAGNOSTICS

There are five prominent errors that processes accessing device files may provoke:

ENODEV − No such device
There is no driver managing the device class this device belongs to. Either the driver is configured out, or it is not loaded (inet).

ENXIO − No such device or address
This device is not available. Either the driver does not support it at all, or the hardware isn't available, i.e. accessing the second disk on a system with only one disk.

EACCES − Permission denied
This error may cause a lot of head scratching if **ls −l** shows a device file to be writable. The media you are trying to access is simply physically write protected!

EINVAL − Invalid argument
Devices may not like reads or writes that are not block multiples, or very big transfers, etc. The device manual page should list the limits.

EIO − I/O error
This may be a real I/O error, i.e. a read or write on the device failing due to a media error. But it may also be the result of an operation that a device can't do, or an empty tape drive, etc.

NOTES

Some devices are not present by default. The **MAKEDEV** script knows how to make them.

MS-DOS/Windows equivalents

The names of MS-DOS/Windows devices probably map to Minix devices as follows:

A:	fd0	
B:	fd1	
C:	c0d0p0	(usually the first partition)
D:	c0d1p0, c0d2p0	(if it's another disk)
D:	c0d0p1s0	(if it's an extended partition)
D:	c0d1, c0d2, c0d3	(if it's a CD-ROM)
CON	console	
COM1	tty00	(UNIX counts from 0)
LPT1	lp	

AUTHOR

Kees J. Bot (kjb@cs.vu.nl)