

Re: number of /dev/usb nodes

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2008-06/msg00103.html>

- *From:* Chuck Robey <chuckr@xxxxxxxxxxxx>
 - *Date:* Sun, 08 Jun 2008 12:52:59 -0400
-

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

Bernd Walter wrote:

On Sun, Jun 08, 2008 at 10:16:26AM -0400, Chuck Robey wrote:

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

Bernd Walter wrote:

On Sat, Jun 07, 2008 at 01:18:41PM -0400, Chuck Robey wrote:

-----BEGIN PGP SIGNED
MESSAGE-----

Hash: SHA1

I can't seem to find how many /dev/usbN bus devices there can be. I'm writing some code that scans them all looking for anything that has my device, but I while I know to start at usb0, just how high do I go? There seem to be 128 device minors, is that the number? (from dev/usb/usb.h)

There shouldn't be a limit anymore.

I can't see any definition of 128 in usb.h that limits the number of busses.

The major/minor differentiation is long time ago.

You must be looking at old code.

I was trying to find a good way to do scanning, whjen I create the files like /dev/usb0, how far to go in my loop? Does the lowest available device always get created? That would imply that as soon as I began to get "No such

Re: number of /dev/usb nodes

device"
errors, I could stop iterating. If the rules for picking device filenames are pretty loose, then I could (for instance) stop scanning, say, 4 numbers past the first "No such device" returnee.

This wouldn't work if a USB controller is remove – e.g. a pulling a cardbus card.

Any idea on this? I didn't see this in the code, but I just need some sane limit on what filenames to scan about in. I look for item info, and if the usb vendor and product look friendly, I just snag the filename involved, and use that. Like, a scan of the usb1 bus might yield me a uhid0 which might be my meat, whereupon I could drop the usb1 open, and replace it with a uhid0 open.
There's more than 1 place that my devices could show, depending on the user's kernel devices. I just want to have some sane limit on how many usb buses I open for my scanning.

I never had to deal with this, since writing a USB driver is simple and as a driver you get informed for each new device.
No need to scan the busses yourself.
But I would say that the most reliable way is to just scan /dev/ for usb...

Assumptions I never said I was writing a FreeBSD driver... I am writing what Xorg calls an input driver (Xinput). I could rely on the config file, I thought I would try to use a scan in case I can't find the dev the user passes me. I see no reason to write a FreeBSD driver when I can do everything I need within the uhid driver (at least so far, in my test prog).

There IS one caveat: I've posted to the FreeBSD-USB list that there is a part of the libusbhid that I can't yet get working. Writing a FreeBSD driver would allow me to use other available data marshalling code, like what's in the ums driver. If I can't interest anyone to comment about the libusbhid, I might be forced down that path, but I don't want to.

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v2.0.4 (FreeBSD)

Comment: Using GnuPG with Mozilla – <http://enigmail.mozdev.org>

iD8DBQFITA5rz62J6PPcoOkRARtGAJ938MKH9L1HRuIpaH3QCy38huJwkQCfUCeF
dE0dyEG+GZUMhi8fAIXNfRk=
=ddvg

-----END PGP SIGNATURE-----

freebsd-hackers@xxxxxxxxxxx mailing list

Re: number of /dev/usb nodes

Re: number of /dev/usb nodes

<http://lists.freebsd.org/mailman/listinfo/freebsd-hackers>

To unsubscribe, send any mail to "freebsd-hackers-unsubscribe@xxxxxxxxxxxxx"