

Re: [PATCH] USB: add USB test and measurement class driver – round 2

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-08/msg12935.html>

- *From:* Greg KH <greg@xxxxxxxxx>
 - *Date:* Fri, 29 Aug 2008 13:01:33 -0700
-

On Fri, Aug 29, 2008 at 06:41:15PM +0200, Marcel Janssen wrote:

On Friday 29 August 2008 16:39:02 Greg KH wrote:

The issue with using cat on the shell level is that it uses fread which has the (in this case) ugly behaviour of recalling the driver's read method until the full number of characters requested has been accumulated (or until zero characters are returned, indicating the end of file). With USBTMC instruments, this behaviour is bad because the retry will not just return zero characters, it will cause a timeout with the associated error condition in the device. So, to enable the use of echo/cat, I added some fread handling to the driver (which catches the retries). I believe this also has been removed, so I assume cat/fread will not work (?).

I do not know, but we do not do wierd things in the kernel just because of broken userspace programs. This logic should be done in userspace, and programs should look at the return value of read() and handle it properly. Otherwise it is a bug.

I don't think this is broken in user space. The problem is that when you issue a measurement command it is not known how many bytes it will return. This is probably due to ASCII output being very common in T&M devices instead of raw data (int, float etc). The ASCII formatting is done in the device and this returns just a string which may or may not be terminated by the term char. This is of course not true for all T&M devices, but the majority works this way.

I admit that the above produces a lot of overhead, but it's just a fact that

Re: [PATCH] USB: add USB test and measurement class driver – round 2

T&M devices work this way, including ours for most of their data processing (not all though).

How is this overhead in userspace, just do something like the following:

```
char big_buffer[16000]; /* bigger than any possible request */
size = read(file_desc, &big_buffer[0], sizeof(big_buffer));
```

and size is the amount of data you actually read from the device, i.e. one request.

I think the USBTMC spec is quite clear on how it should be implemented on messaging level. Basically when you issue the command "*IDN?" the device will process this and return the device ID string. The length of this string is set in the TransferSize of the 12 byte header that the device returns. The problem when you issue a read command is that the read command does not yet know how much data to expect. It should issue the REQUEST_DEV_DEP_MSG_IN first and set the TransferSize value high enough.

In the USBTMC_488 extension you find an example (chapter 3.3.1 page 7) that shows the REQUEST_DEV_DEP_MSG_IN TransferSize being set to 64 although the actual data being returned from the device is less (only 36 bytes).

What you do see in practice is that when someone would issue a read command and asking for less bytes than are available is that the user program may handle this as a warning telling the user that he did not request all available data.

Then that's a userspace bug, don't do that in your program that reads from these types of devices :)

Stefan's driver works exactly the way I would expect from a users point of view. Whether the implementation can be improved is another issue, but the behaviour is correct and compliant with other usbtmc drivers on other platforms.

But it's not compliant with the "standard" way of using a file descriptor in unix, and might break some POSIX requirements as well (I'm not a good POSIX follower, so I don't know for sure...)

As this is trivial to handle in userspace, and requires no additional wierd code in the kernel driver, I don't see this as something we should change the driver for.

thanks,

greg k-h

—

Re: [PATCH] USB: add USB test and measurement class driver – round 2

Re: [PATCH] USB: add USB test and measurement class driver – round 2

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>