Linux Fu: Eavesdropping On Serial

Originalartikel

Backup

<html> In the old days, if you wanted to snoop on a piece of serial gear, you probably had a serial monitor or, perhaps, an attachment for your scope or logic analyzer. Today, you can get cheap logic analyzers that can do the job, but what if you want a software-only solution? Recently, I needed to do a little debugging on a USB serial port and, of course, there isn’t really anywhere to easily tie in a monitor or a logic analyzer. So I started looking for an alternate solution. If you recall, in a previous Linux Fu we talked about pseudoterminals which look like serial ports but actually talk to a piece of software. That might make you think: why not put a piece of monitor software between the serial port and a pty? Why not, indeed? That’s such a good idea that it has already been done. When it works, it works well. The only issue is, of course, that it doesn’t always work. The software in question is intercepttty. You may have to build it from source, but there aren’t any oddball dependencies. Just: <pre>git clone https://github.com/geoffmeyers/interceptty.gitcd intercepttty./configuremakesudo make install # or however you like to install things like checkinstall, etc.</pre><h2>Back End, Front End</h2> The software uses the concept of a backend device and a frontend device. The back device is, usually, your normal serial port. The frontend device is something that interceptty creates. So the idea is that you connect the program to the backend, it creates the front end, and then you connect some other program to the front end. The program will log all the traffic between the program connected to the front end and the port on the back end. You can also use file descriptors, unix sockets, or TCP sockets as a front or back device. The backend can also be a running program. There is also a provision for connecting between two actual ports. You can find all the options on the program’s man page. The output is a bit awkward, but it is easy to parse by other programs including an example Perl program included with it. A character shows you the direction of the data then you see a character in both hex and ASCII. <h2>An Example</h2> It is probably easier to see an example. I have a small controller on

```
/dev/ttyACM0
```

. Here’s a sample command line. <pre>sudo interceptty /dev/ttyACM0 /dev/ttyCAP</pre> This creates the

```
/dev/ttyCAP
```

fake port. Once the program connects you can see the data it sends to the left and the responses (including the echo) to the right. <pre>< 0x3a (:) > 0x3a (:) < 0x78 (x) > 0x78 (x) < 0x38 (8) > 0x38 (8) < 0x79 (y) > 0x79 (y) < 0x38 (8) > 0x38 (8) < 0x69 (i) > 0x69 (i) < 0x31 (1) > 0x31 (1) < 0x30 (0) > 0x30 (0) < 0x21 (!) > 0x21 (!) < 0x0d ([CR]) > 0x06 ([ACK]) > 0x0d ([CR]) > 0x0a ([LF])</pre><h2>Problematic Software</h2> Unfortunately, not all software likes to work with ptys. In particular, the main program I wanted to use takes advantage of the Sigrok serial port library. It is a known issue that this library makes calls that don’t work well with ptys. However, if you

use just about any normal terminal program like picocom or tio, it works fine. Other serial libraries seem to be able to handle it, also. I thought about trying slsnif, but it works the same way, so I doubt it would be any better in that regard. One other note about the Sigrok library. Recent changes in the kernel have removed an ioctl that the library uses so if you are using software that uses libserialport, you won’t be able to find any ports. Until they fix it, you’ll have to build the library yourself and patch the configure.ac file: <pre>git clone git:*sigrok.org/libserialportcd libserialport./autogen.sh./configuremakesudo make install</pre> Be sure you get rid of old versions of the library or, at least, make sure your program is using your custom library. <h2>Other Options</h2> You can, if you like, try converting a serial port to a network socket and then you have lots of monitoring options. Of course, this has similar problems insomuch as not everything understands the fake serial port you create. I can report, though, that Tio seems well-behaved enough to work with these pty fake port. It would be nice if all serial software understood that they might be called on to talk to a pty or a named pipe. On the other hand, you may have access to the source code for naughty programs, so if you really had to, you could fix them. </html>