I wanted a way to access the serial console of various router, switches, and firewalls in my lab without putting them on the network. There are not many user friendly or accessible platforms that allow such for lab use so I set out to build my own. This is what I ended up with after some research and an afternoon of tinkering.

I am using a 4gb Raspberry Pi 5 and some StartTech USB to RS232 DB9 adapters with the Prolific PL2303GT chip set. I installed Raspberry Pi OS Lite 64-bit for this and have the Serial to USB adapters connected to the system.

Serial Device Setup

The following command will show you the current serial devices connected to the system.

```
$ ls -la /dev/ | grep “USB”
```

The following command can also be used to verify the currently connected serial devices.

```
$ dmesg | grep “ttyUSB”
```

USB devices are numbered based on the order they are connected to the system. This can change the name of devices across reboots. You can create rules that pin the serial adapters to specific names. This is done by taking an attribute unique each adapter and using that to identify and then name the device.

First use the following command to list all the attributes about a specific device.

```
$ udevadm info --name=/dev/ttyUSB0 --attribute-walk
```

With that output I am using the vendor id and serial number to uniquely identify each device on the system. Use the following commands to filter out the specific information needed:

```
$ udevadm info --name=/dev/ttyUSB0 --attribute-walk | grep "serial"
$ udevadm info --name=/dev/ttyUSB0 --attribute-walk | grep “idVendor"
```

The sting you want to create will look something like this:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="067b", ATTRS{serial}==“BFAPb137X02", SYMLINK+="SerialDevice1"
```

The SYMLINK field defines the name that you want configured for that specific device.

There may be two idVendor fields that have differing values in the `udevadm` output. The `lsusb` command can be used to identify the correct one:

```
Bus 001 Device 002: ID 067b:23a3 Prolific Technology, Inc. ATEN Serial Bridge
```

In my case the value of the idVendor field I need is *067b*.

Now you need to set the names you need to create a new file in the `/etc/udev/rules.d/` directory. I named my file `99-usb-serial.rules`. Then added the strings to the file. After rebooting and using the `ls -la /dev/` command you can verify that the devices have been renamed.

ser2net Installation

Now to install ser2net. This is the software that allows you to connect to a serial device over the network.

```
$ sudo apt install ser2net
```

The configuration is stored in a yaml configuration file named `/etc/ser2net.yaml`. This example configuration will use telnet to bridge a port to a particular serial connection.

```
connection: &con0001
       accepter: telnet,10.2.102.5,2001
       enable: on
       options:
         banner: *banner
         kickolduser: true
         telnet-brk-on-sync: true
       connector: serialdev,
                 /dev/SerialDevice1,
                 9600N81,local
```

This configuration can be copied for another device. The connection name, accepter port and, serial device will need to be changed for each serial device you have. Note here we are using the names for the serial adapters that we configured earlier. I also pinned the accepter to the static IP I configured on the Raspberry Pi.

When you change you need to restart the ser2net process. Use the following configuration to do so.

```
$ sudo systemctl restart ser2net.service
```

To check the status of the service and ensure there are no configuration errors use this command.

```
$ sudo systemctl status ser2net.service
```

Ser2net is extremely versatile and I implore you to read through the wiki to see all the different configuration methods that are possible.

Resources

https://github.com/cminyard/ser2net
https://manpages.debian.org/unstable/ser2net/index.html
https://deepwiki.com/cminyard/ser2net/1-overview