UNIX & LINUX

How to bind USB device under a static name?

Ask Question

36

17

I have an Arduino which sometimes gets bound to `/dev/ttyUSB0` and other times to `/dev/ttyUSB1`, making my script fail.

I do not want to enumerate all the possibilities of where my device could be, but I'd rather have it be bound somewhere static, e.g. `/dev/arduino`.

How do I achieve that?

linux arch-linux usb serial-port arduino

edited Oct 29 '18 at 16:55
Renan
14.7k ● 6 ● 56 ● 79

asked Mar 5 '13 at 7:38
k0pernikus
4,512 ● 15 ● 45 ● 67

4

uino to right devise by its VID & PID. – Eddy_Em Mar 5 '13 at 8:00

Check arch-wiki: wiki.archlinux.org/index.php/Udev#Writing_udev_rules – uzsolt Mar 5 '13 at 10:01

1

After you've changed the rules, see How to reload udev rules without reboot? – Gilles Mar 5 '13 at 22:00

5 Answers

36

✓

As suggested, you can add some udev rules. I edited the `/etc/udev/rules.d/10-local.rules` to contain:

```
ACTION=="add", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001",
SYMLINK+="my_uart"
```

You can check for the variables of your device by running

```
udevadm info -a -p  $(udevadm info -q path -n /dev/ttyUSB0)
```

There is a more in depth guide you can read on http://www.reactivated.net/writing_udev_rules.html

answered Mar 5 '13 at 10:53

Kotte
1,527 ● 13 ● 23

Worked like a charm. One question: How to exit `udevam`? And it is important to note that `my_uart` creates the symlink under `/dev/my_uart`. I first wrote `/dev/arduino` the first time and it failed whilst `arduino` is sufficient. – k0pernikus Mar 7 '13 at 19:11

`udevadm` should exit by itself when it's done. – Kotte Mar 8 '13 at 7:31

Then for some unknown reason it froze the terminal session to my Raspberry Pi while generating the report. – k0pernikus Mar 8 '13 at 12:19

27

The rule syntax above may work on some distributions, but did not work on mine (Raspbian). Since I never found a single document that explains all the ins and outs, I wrote my own, to be found here. This is what it boils down to.
1. find out what's on ttyUSB:

```
dmesg | grep ttyUSB
```

2. list all attributes of the device:

```
udevadm info --name=/dev/ttyUSBx --attribute-walk
```

(with your device number(s) instead of x, of course). Pick out a unique identifier set, eg idVendor + idProduct. You may also need SerialNumber if you have more than one device with the same idVendor and idProduct. SerialNumbers ought to be unique for each device.
3. Create a file `/etc/udev/rules.d/99-usb-serial.rules` with something like this line in it:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="1234", ATTRS{idProduct}=="5678",
SYMLINK+="your_device_name"
```

(assuming you don't need a serial number there, and of course with the numbers for idVendor and idProduct that you found in step 2.
4. Load the new rule:

```
sudo udevadm trigger
```

5. Verify what happened:

```
ls -l /dev/your_device_name
```

will show what ttyUSB number the symlink went to. If it's `/dev/ttyUSB1`, then verify who owns that and to which group it belongs:

```
ls -l /dev/ttyUSB1
```

Then just for the fun of it:

```
udevadm test -a -p  $(udevadm info -q path -n /dev/your_device_name)
```

edited Jul 24 '15 at 7:44

answered Feb 7 '15 at 14:20

RolfBly
442 ● 1 ● 6 ● 12

So is there any solution if the `idVendor` and `idProduct` are exactly the same? (two sensors attached on identical model USB to UART modules) – Steven Lu Jul 22 '15 at 2:00

@StevenLu Yes, see step 2, do `udevadm info --name=/dev/ttyUSB1 --attribute-walk` for both devices and look for serial numbers, they should be unique for each device. If your sensors have no serial number, can you specify what they are? – RolfBly Jul 22 '15 at 9:31

that's awesome, i will report back when i try this out – Steven Lu Jul 22 '15 at 19:43

My $2 USB to UART dongles have serial number 0001. Can't say I'm surprised. Looks like I have to identify the sensors based on their output protocol. – Steven Lu Aug 8 '15 at 20:18

@StevenLu Bad luck. FTDI USB-UART converters do have a unique serial number, AFAIK. A few extra bucks, but less time to develop. – RolfBly Aug 10 '15 at 20:41

▲
7
▼

The multiple-identical-USB-device problem

I have a Rasperry Pi with four cameras. I take pix with `fswebcam` which identifies the cameras as `/dev/video0` .. `video3`. Sometimes the camera is `video0`, `vide02`, `video4` and `video6` but we can forget about that for now.

I need a persistent ID to identify a camera number so that, e.g. `video0` is always the same camera because I caption the pictures. Unfortunately this doesn't happen reliably - on boot, the cameras get enumerated as `video0` .. `video3` but not always the same way.

The cameras all have the same ID and serial number.

The solution to this problem involves udev rules, but there's a lot of fishhooks there as well.

If you issue the command

```
udevadm info –attribute-walk –path=/dev/video0
```

you get a screed of output but the salient bits are

```
KERNEL=”video0”, SUBSYSTEM=”video4linux” and KERNELS=”1:1.2.4:1.0”.
```

The KERNELS bit is a USB hub port. With four cameras there are four of these - they do not change on reboot , but the `video{x}` associated with a port *may* change.

So we need a udev rule to tie a video number to a USB hub port - something like:

```
KERNEL==”video0”,SUBSYSTEM=”video4linux”,KERNELS==”1:1.2.4:1.0”,SYMLINK+=”camera0”
```

Looks simple – access the camera with

```
fswebcam –d  $realpath /dev/camera0
```

Except it doesn’t work – if you put this in a udev rule and the system has allocated video0 (on boot) to a different port, the udev rule is ignored. The symlink to `/dev/camera0` basically says `no such device`. Square one.

What we want is to bind a symlink to a USB hub address, not a `video{x}` number. It took a Python program.

First step was to run

```
fswebcam –d /dev/video${x}  tst.jpg
```

for `x` between 1 and 8. The existence of `tst.jpg` after each call identifies whether there is a camera on this video number. From this make a list of active video numbers. My experience has been that it is either `0,1,2,3` or `0,2,4,6` for cameras I have used.

Others may of course build this list using a different process.

Then for each video number in the list run

```
udevadm info –attribute-walk –path=/dev/videox > dd
```

and extract the `KERNELS= line` from `dd`. From this process you end up with a list of the USB port addresses for the cameras. Sort this list so that at the next step, you always process it in the same order. Call this the "address list".

Run the `udevadm … > dd` thing again and make a list that looks like

```
KERNEL==”video0”, SUBSYSTEM=”video4linux”,KERNELS==”1:1.2.4:1.0
”,SYMLINK+=”camerax”. Call this the “video list”.
```

Now step through the address list - for each entry find the corresponding entry from the video list. Create a new list that looks like a collection of lines like

```
KERNEL==”video0”, SUBSYSTEM=”video4linux”,KERNELS==”1:1.2.4:1.0
”,SYMLINK+=”camera2”
```

The x (symlink number) is replaced by the sequence number in the address list.

Now you have a udev rule that works. A symlink that is tied to a USB hub address no matter what video number is allocated to that port at boot.

Write the final list into a file `/etc/udev/rules.d/cam.rules`. Run `udevadm trigger` to activate it and the job is done. `/dev/camera2` will be the same camera (USB port) regardless of its video number.

edited Nov 29 '16 at 9:46

k0pernikus
4,512 ● 15 ● 45 ● 67

answered Nov 29 '16 at 1:07

Ian Boag
71 ● 1 ● 1

Welcome on unix stackexchange. Please format your answer using markdown. I just did it for you. Also keep in mind that we want answers to be to the point. This reads more like a blog entry (which is not entirely bad) yet it's not that helpful to first read about approaches that didn't work. You may scrap that part. – k0pernikus Nov 29 '16 at 9:48

Sorry. I'm new here. I have researched this problem for months. I did find others struggling with the same problem and I did not find an answer that worked for me. Just so I know, where would you advise that I post something like this? I did restrain myself and not include the Python source :-) – Ian Boag Nov 29 '16 at 22:44

1

I was also able to find a unique device in `/dev/serial/by-id`. I haven't tried a reboot yet, but the files in that directory were just links to the appropriate device file (`ttyACM[0-9]`).`

I am running arch linux on Raspberry Pi, but I stumbled across them just by doing a `find` for filenames containing "Arduino". My python programs run fine using those files as devices to read/write data to/from my Arduinos (so far, two on a single Pi).

edited Feb 15 '16 at 2:03
sam
14.7k ● 3 ● 16 ● 27

answered Feb 14 '16 at 23:41
huey_driver
11 ● 2

0

Just to say that the above worked for me and also automounted the device for me after I had placed an entry in /etc/fstab (and it also calls umount after removal of the stick)

i.e.

/etc/fstab

```
# See /etc/udev/rules.d/5-usb-disk.rules
/dev/backup     /vol/backup     ext4    defaults,errors=remount-ro 0       1
```

cat /etc/udev/rules.d/5-usb-stick.rules

```
#
# the next line creates a symlink to this disk drive called /dev/backup
# i.e.
#   root:# ls -la /dev/backup
#   lrwxrwxrwx 1 root root 3 Jul 22 19:33 /dev/backup -> sg0

# Backup usb stick - create /dev/backup
# ATTRS{model}=="Cruzer Blade    "
ACTION=="add", ATTRS{model}=="Cruzer Blade    ", SYMLINK+="backup"

# Clean up after removal
ACTION=="remove", ATTRS{model}=="Cruzer Blade    ", RUN+="/bin/umount /vol/backup"
```

So after inserting my usb stick I get:

```
root:# mount | grep sd
/dev/sda1 on /vol/backup type ext4 (rw,relatime,errors=remount-ro,data=ordered)
```

answered Jul 22 '17 at 18:46

Will

21 ● 3