Learning how to disable specific USB devices by their ports in Linux

migueleonardortiz.com.ar/linux/learning-how-to-disable-specific-usb-devices-by-their-ports-in-linux/1645

Miguel

3 de septiembre de 2018

I was required to disable an specific port for an specific device. If an «X» device is connected in any port of the computer I should be able to enable/disable it at will.

In our scenario the idea is to enable or disable a Genius GK100010 Numeric Pin Pad and other several devices when they are connected to any USB port of our computer. (I have installed Lubuntu 12.04).

There are other alternatives as the use of xinput or using udev rules but in this article I will cover the procedure with the Manual driver binding and unbinding feature included in kernel 2.6.13-rc3.

At the end of this reading you'll learn:

- How identify usb devices
- How to manually bind/unbind usb ports
- How to script an ON/OFF usb port for certain device

For this solution we'll need:

- Bash 4 (or higher) to use readarray | #you can replace this with your own «for/loop» to read each device
- Kernel 2.6.13-rc3 (or higher) | #mandatory

List our device

First, we must connect the device and list it with lsusb:

Shell

```
root@mortiz:~# lsusb
#                   |           |
#                   v           v
Bus 002 Device 021: ID 040b:2000 Weltrend Semiconductor
Bus 002 Device 002: ID 17ef:6099 Lenovo
Bus 002 Device 003: ID 05e3:0610 Genesys Logic, Inc. 4-port hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
```

In my case, the mentioned device above its recognized as «Weltrend Semiconductor». Using those kind of names isn't a good practice. Instead we'll focus on it's **ID NUMBER, which is «040b:2000»** when scripting, I'll talk about this in «scripting a solution» below.

Identify which port is using

Now, we must investigate which port is being used for that device, the file **uevent** will store all the information about the device, there's a correlation between that file and the **Bus** and **Device** information provided by **lsusb**, so we take both previous values and stick them together with a slash:

Shell

```
002/021
```

Now, using this values we'll search for it's respective **uevent** file this way:

grep -l # list files with coincidences and suppress matched lines

Shell

```
#                   |
#                   v
root@mortiz:~# grep -l 002/021 /sys/bus/usb/devices/*/uevent
/sys/bus/usb/devices/2-2/uevent
```

Now, the port being used by our USB device is the «2-2», easy to see in **/sys/bus/usb/devices/2-2/uevent**

Turn the port on and off

We could directly turn on and off the port in this way:

```
echo '2-4' > /sys/bus/usb/drivers/usb/bind #ON
echo '2-4' > /sys/bus/usb/drivers/usb/unbind #OFF
```

Be careful, the Bus and Device change if you reconnect the device or restart your computer, to learn how to make this changes in a foolproof way keep reading through scripting a solution below.

Scripting a solution

As I've mentioned before the Bus and Device may change when reconnecting or restarting the computer. We must find a reliable way to always find the device. Remember the ID NUMBER I've mentioned?

Shell

```
root@p0008bc4:~# lsusb
#                   |
#                   v
Bus 002 Device 021: ID 040b:2000 Weltrend Semiconductor
Bus 002 Device 002: ID 17ef:6099 Lenovo
```

The value: **040b:2000** is the safest identifier to enable or disable this device. Let's take a look on our uevent file:

Shell

```
root@p0008bc4:~# cat /sys/bus/usb/devices/2-
2/uevent
MAJOR=189
MINOR=148
DEVNAME=bus/usb/002/021
DEVTYPE=usb_device
DRIVER=usb
PRODUCT=40b/2000/205 # <----
TYPE=0/0/0
BUSNUM=002
DEVNUM=021
```

As you can see our previous coincidence was the «DEVNAME» when we used grep with the value «002/021». We know that values may change but the ID NUMBER won't and we'll find it in the variable «PRODUCT».

There's a small difference between the ID NUMBER provided by lsusb: **040b:2000** and the PRODUCT in our uevent file: **PRODUCT=40b/2000.** The format and a missing '0' make them different, anyway this is the best way to find our device.

So we must change **040b:2000** to **40b/2000** and search that device ID NUMBER in our ports:

```
root@p0008bc4:~# grep -l '40b/2000' /sys/bus/usb/devices/*/uevent
/sys/bus/usb/devices/2-4:1.0/uevent
/sys/bus/usb/devices/2-4:1.1/uevent
/sys/bus/usb/devices/2-4/uevent
```

As you can see, three files appear but the port **2-2** it's the same in each of them. To filter the output just add tail -1 like this:

```
root@p0008bc4:~# grep -l '40b/2000' /sys/bus/usb/devices/*/uevent | tail -1
/sys/bus/usb/devices/2-4/uevent
```

And add it to a variable:

Objective-C

```
PORTID=$(grep -l '40b/2000' /sys/bus/usb/devices/*/uevent | tail -1)
echo $PORTID
/sys/bus/usb/devices/2-4/uevent
```

With the help of tr and awk we split the string (/sys/bus/usb/devices/2-2/uevent) and take the column where our ports in use are (2-2):

```
1  PORTID=$(echo $PORTID | tr "/" " "|awk '{print
2  $5}')
3  echo $PORTID
4  #after tr
5  sys bus usb devices 2-4 uevent
6  # after awk
   2-4
```

And that's how we should search for the port of this device, then we are able to turn it on and off at will:

```
1  echo $PORTID > /sys/bus/usb/drivers/usb/bind #ON
2  echo $PORTID > /sys/bus/usb/drivers/usb/unbind #OFF
```

I've written a full functional script to do this tasks with an specific device **listed in a file called «devices.list»** and must be placed in /usr/local/bin/devices.list, the content of the file is the identifier of lsusb:

```
1  root@florida:/usr/local/bin# cat devices.list
2  040b:2000
```

Possible errors

When you perform the operation of «turn on and off» as I've explained the system enables or disables a port by a **symlink**:

```
# Available ports
root@mortiz:~# ls -ltr /sys/bus/usb/drivers/usb/
total 0
lrwxrwxrwx 1 root root    0 sep  3 13:39 usb3 ->
../../../../devices/pci0000:00/0000:00:10.0/usb3
lrwxrwxrwx 1 root root    0 sep  3 13:39 usb2 ->
../../../../devices/pci0000:00/0000:00:10.0/usb2
lrwxrwxrwx 1 root root    0 sep  3 13:39 usb1 ->
../../../../devices/pci0000:00/0000:00:02.2/0000:02:00.4/usb1
--w------- 1 root root 4096 sep  3 13:39 uevent
<strong>lrwxrwxrwx 1 root root    0 sep  3 13:39 2-4 ->
../../../../devices/pci0000:00/0000:00:10.0/usb2/2-4</strong>
--w------- 1 root root 4096 sep  3 16:28 bind
--w------- 1 root root 4096 sep  3 17:01 unbind
# Then we execute an "OFF" like this:
echo '2-4' > /sys/bus/usb/drivers/usb/unbind #OFF
# And our symlink is gone:
root@p0008bc4:~# ls -ltr /sys/bus/usb/drivers/usb/
total 0
lrwxrwxrwx 1 root root    0 sep  3 13:39 usb3 ->
../../../../devices/pci0000:00/0000:00:10.0/usb3
lrwxrwxrwx 1 root root    0 sep  3 13:39 usb2 ->
../../../../devices/pci0000:00/0000:00:10.0/usb2
lrwxrwxrwx 1 root root    0 sep  3 13:39 usb1 ->
../../../../devices/pci0000:00/0000:00:02.2/0000:02:00.4/usb1
--w------- 1 root root 4096 sep  3 13:39 uevent
--w------- 1 root root 4096 sep  3 16:28 bind
--w------- 1 root root 4096 sep  3 17:01 unbind
```

Yes, it would be a cool trick to delete or create the symlink but I prefer doing it by the «echo» procedure because I don’t know if there are other changes.

The thing with this is, check that the symlink exists or not before trying to perform an ON / OFF operation while scripting and have in mind that my mentioned script doesn’t cover corner cases when you turn it off, disconnect and change the port of the device.