rapid technology

sabtu, 26 april 2008

USB_ModeSwitch - Activating Switchable USB Devices on Linux

USB_ModeSwitch is (surprise!) a mode switching tool for controlling "flip flop" (multiple device) USB gear.

Several new USB devices (especially high-speed wireless WAN stuff, they're expensive anyway) have their MS Windows drivers onboard; when plugged in for the first time they act like a flash storage and start installing the driver from there. After that (and on every consecutive plugging) this driver switches the mode internally, the storage device vanishes (in most cases), and a new device (like an USB modem) shows up. The WWAN gear maker Option calls that feature "ZeroCD (TM)".

As you may have guessed, nothing of this is documented in any form and so far there is no official Linux driver available. On the good side, most of the known devices work out of the box with the available Linux drivers like "usb-storage" or "usbserial". That leaves the problem of the mode switching from storage to whatever the thing is supposed to do.

Fortunately there are things like human reason, USB sniffing programs and "libusb". It is possible to eavesdrop the communication of the MS Windows driver, to isolate the command or action that does the switching, and to reproduce the same thing with Linux.

USB_ModeSwitch makes the last step considerably easier by taking the important parameters from a configuration file and doing all the initialization and communication stuff.
It does NOT check for success afterwards as of now. The right approach would be to consult /proc/bus/usb/devices (or the output of "lsusb") before and after execution to note any changes.

Download

The latest release version is 0.9.3. The archive contains the source and a 386 binary compiled without any optimizations. I used libusb-0.1.12.
There are changes and updates to the config file more often than new releases; most of the valuable knowledge about devices is contained there. So you better use the latest version linked here.

- Download usb_modeswitch-0.9.3.tar.bz2

mengenai saya

LIFE INSPIRATION

Lihat profil lengkapku

- Load the latest usb_modeswitch.conf (release version); the default place is "/etc". Last updated 2008-03-09
- Don't forget libusb if you don't have it.

There is a beta version, usb_modeswitch-0.9.4beta2.tar.bz2 , that supports multiple devices and probably fixes the udev problems. Please test!

How to install

If you want to compile it for yourself, just run "compile.sh" or type on the shell:

```
$ gcc -l usb -o usb_modeswitch usb_modeswitch.c
```

That's as easy as it gets ... And it should be HIGHLY portable (OS X anyone?).

Take the fresh executable "usb_modeswitch" (or the one provided with the archive) and put it into your path (preferably "/sbin" or "/usr/sbin").
Put "usb_modeswitch.conf" into "/etc" and edit it according to your hardware. It's heavily commented and should tell you what to do.

Alternatively you can use the brand new command line interface to tell USB_ModeSwitch the things it needs to know; try "usb_modeswitch -h" to list the parameters. This way you can handle multiple configurations. If any command line parameters are used, the default config file is NOT read.

Important: USB_ModeSwitch - like all programs with libusb routines - has to be run as root. Otherwise strange error messages come up ...

Troubleshooting: if you're next to certain that you have the right values for your device, and run after run USB_ModeSwitch seems to do something but nothing changes, there are most likely timing issues involved. Try to run the program manually after a defined delay immediately following the plugging-in of the device, starting from 1 second and working up to, say, 6. If any run was successful, note the time for later steps of automatization (see below).
This is caused by the hot-plugging mechanics of some modern distribution and depends on when they grab and use the device. Maybe someone can provide more insight there.

Known working hardware

Personally, I could only test my Option Icon; the list here - as well as all the necessary data - relies on reports from third parties (people, that is). So don't be surprised if you hit sudden obstacles even with your device listed here. You have been warned.

There are three known methods for initiating the switching process so far:
1. sending a SCSI command (which is rarely or never used for flash devices) to the storage device
2. actively removing (rather detaching) the storage driver from the device
3. sending a special control message to the device

It is recommended to use the latest firmware available as there have been issues with at least one device (Icon 7.2) in that respect.

- **Option GlobeSurfer Icon** (aka "**Vodafone EasyBox**")
  The thing that started it all, because I wanted it to work on my Linux router.
  All known Option devices use the USB storage command REZERO UNIT for switching.

- **Option GlobeSurfer Icon 7.2**
  If you get hardware lockups of this thing when plugging in (flashing LEDs), update the firmware.
- **Option GlobeSurfer Icon 7.2 with HSO driver interface**
  A next generation firmware with vendor/device ID unchanging. Your "7.2 ready" device might change its behaviour after re-flashing with this firmware; newer Option devices most likely come loaded with it. Use the new "TargetClass" parameter to recognize already switched devices.
  Note: for HSO driver questions and howtos turn to the fine Pharscape site!
- **Option Icon 225 HSDPA** (aka "**T-Mobile web'n'walk Stick**")
  New Firmware, HSO interface
- **Option GlobeTrotter HSUPA Modem** (aka "**T-Mobile wnw Card Compact III**")
  New Firmware, HSO interface
- **Option GlobeTrotter GT MAX 3.6** (aka "**T-Mobile wnw Card Compact II**")
- **Option GlobeTrotter EXPRESS 7.2** (aka "**T-Mobile wnw Card Express II**")
- **Option GlobeTrotter GT MAX "7.2 Ready"**
  Timing is an issue, see delayed script below. Lucas Benedičič did extended tests and found a minimum delay of three to four seconds working reliably, depending on the setup and probably the machine.
- **Huawei E220** (aka "**Vodafone EasyBox II**", aka "**T-Mobile wnw Box Micro**")
  We now have two options (no pun intended!) for this: 1. removal of "usb-storage" 2. the special control message found by Miroslav Bobovsky. The latter is independent of "usb-storage" and even leaves the storage portion of the device functional.
- **Huawei E270**
  Same setup as the E220.
- **Huawei E630** (Experimental!)
  Beware: there are modem-only variants around (without the storage part); for these no switching is required.
- **ZTE MF620** (aka "**Onda MH600HS**")
  Uses the USB storage command TEST UNIT READY for switching.
- **ZTE MF622**
  Detachment of storage driver is sufficient; a minimum delay is recommended (3-4 seconds). See delayed script below
- **Novatel Wireless Ovation MC950D HSUPA** and **XU950D**
  Uses the USB storage command START/STOP (Eject) for switching.
- **Novatel U727 USB modem**
  Similar setup as the MC950D

How to automate

Mind that you have to run USB_ModeSwitch every time you plug your device or cold boot with it. If you have "udev" in your distribution (almost all do) it's really not hard to automate this and just forget about it.

If you get a success result when running the tool manually, but the device still shows up as storage, try the latest beta version. If this doesn't work either, timing might be an issue with many of the more your setup; in that case the delayed execution of USB_ModeSwitch might help. See the special script below.

You should have a folder named "/etc/udev" or similar. Somewhere in there (I have a folder "rules.d") you find some files with the extension ".rules". Create a new one (or edit an existing one, but by convention **not** the default "50-something.rules"). Mind that the parsing order depends on the file name; my instinct tells me to put the switching stuff rather at the end of the line so that the whole storage business is taken care of beforehand.
In the chosen/new file enter/add the line

```
SUBSYSTEM=="usb", SYSFS{idProduct}=="", SYSFS{idVendor}=="", RUN+=""
```

That's basically it.

From here, there are two ways to continue. If your GSM device is recognized by a recent version of the "option" driver you shouldn't have to do anything but to load the module (most certainly handled by udev automatically). Instead or if your (serial) device is not supported by that module you can always use "usbserial", but it needs to be told the device IDs (plus a performance-related option):

```
SUBSYSTEM=="usb", SYSFS{idProduct}=="", SYSFS{idVendor}=="",
 RUN+="/sbin/modprobe usbserial vendor= product= maxSize=4096"
```

As for the difference between "usbserial" and "option", here is a quote from **option.c**:

```
  This driver exists because the "normal" serial driver doesn't work too well
 with GSM modems. Issues:
 - data loss -- one single Receive URB is not nearly enough
 - nonstandard flow (Option devices) control
 - controlling the baud rate doesn't make sense
```

Following that, I'd recommend trying the "option" driver first. In recent kernels it recognizes several Option, Huawei and Novatel devices (among others) out of the box. And following kernel developers mail traffic it looks like this driver is becoming the standard for GSM devices as more models are added. In the latest kernels the module entry reads "USB driver for GSM and CDMA modems" (Device Drivers / USB support / USB Serial Converter support).

Devices supported by the "option" driver that don't change their IDs after switching might run into problems because of the driver trying to attach before the switching happened. In this case it might help to blacklist it and to load it manually via the helper script after execution of usb_modeswitch. Again, developers are working on the "option" driver to probe for the device class before binding, so this problem might be handled in kernel 2.6.24.

For plain serial devices that keep their ID after switching a script "/sbin /mydevice_switch.sh" can be created:

```
#!/bin/sh
/sbin/usb_modeswitch
sleep   # probably not necessary, try out
/sbin/modprobe usbserial vendor= product= maxSize=4096
```

And then add this rule:

```
SUBSYSTEM=="usb", SYSFS{idProduct}=="", SYSFS{idVendor}=="", RUN+="/sbin/mydeviceswitch.sh"
```

If **timing** is an issue with your device or setup it might help to delay the execution of USB_ModeSwitch, to allow other drivers like "usb-storage" to finish their activation. Again, use the helper script - called "/sbin

/mydevice_switch.sh" here - and fill it like this:

```
#!/bin/sh
sh -c "sleep 4; /usr/bin/usb_modeswitch" &
exit 0
```

Luigi Iotti reported problems on some systems (RHEL 5, CentOS 5) of udev always waiting for background scripts to finish. Here is his solution for a changed "/sbin/mydevice_switch.sh":

```
#!/bin/sh
# close these FDs to detach from udev
exec 1<&- 2<&- 5<&- 7<&-
sh -c "sleep 4; /usr/bin/usb_modeswitch" &
exit 0
```

Contribute

USB_ModeSwitch comes quite handy for experimenting with your own hardware if not supported yet. You could try this approach:

Note the device's Vendor and Product ID from /proc/bus/usb/devices (or from the output of "lsusb"); the assigned driver is usually "usb-storage".
Then try spying out the USB communication to the device with the same ID inside M$ Windoze.
I recommend this tool: "SniffUSB" (http://benoit.papillault.free.fr/usbsnoop/index.php.en).

This is the extremely short version. I will post a more detailed HOWTO in the future.

Please post any improvements, new device information and/or bug reports to the ModeSwitchForum !
Or send me an old-fashioned e-mail (see below).

Whodunit

Copyright (C) 2007 Josua Dietze (digidietze at this domain)

Command line parsing, decent usage/config output and handling, bugfixes added by:

- Joakim Wennergren (jokedst) (gmail.com)

TargetClass parameter implementation to support new Option devices/firmware:

- Paul Hardwick (http://www.pharscape.org)

Created with initial help from:

- "usbsnoop2libusb.pl" by Timo Lindfors (http://iki.fi/lindi/usb/usbsnoop2libusb.pl)

Config file parsing stuff borrowed from:

- Guillaume Dargaud (http://www.gdargaud.net/Hack/SourceCode.html)

Hexstr2bin function borrowed from:

- Jouni Malinen (http://hostap.epitest.fi/wpa_supplicant, from "common.c")

Code, ideas and other input from:

- Aki Makkonen
- Denis Sutter
- Lucas Benedičič
- Roman Laube
- Luigi Iotti

More contributors (device specific) are listed in the config file.

History

Version 0.9.4beta2, 2008/03/19 Possible udev release fix Version 0.9.4beta, 2008/03/16
Multiple device support
Version 0.9.3, 2008/03/09
More devices, no other changes from 0.9.3beta
Version 0.9.3beta, 2007/12/30
New TargetClass parameter for recent Option firmware (Paul Hardwick), more devices
Version 0.9.2, 2007/11/02
New Huawei mode (code from Miroslav Bobovsky, added by Denis Sutter), more devices
Version 0.9.1beta, 2007/09/11
Added command line parsing (jokedst), cleaned up config stuff (jokedst), bug fixes, doc updates
Version 0.9beta, 2007/08/12
Name change from "icon_switch", parameter file and generalizing
Version 0.2, 2006/09/25
Code cleaning, more messages
Version 0.1, 2006/09/24
Just very basic functionality ...

Legal

diposkan oleh life inspiration di 15.01

tidak ada komentar:

Poskan Komentar

link ke posting ini

Buat sebuah Link

Posting Lebih Baru Beranda Posting Lama

Langganan: Poskan Komentar (Atom)