Handspring−Visor mini−HOWTO

Table of Contents

Handspring−Visor mini−HOWTO

Ryan VanderBijl

v0.4, Aug 7, 2000

Using the Visor with Linux and your USB port

1.Introduction

1.1 About this Document

This document was first created by Ryan VanderBijl, April 6, 2:00am. It has been edited by Miles Lott, and I have recieved hints from others.

I recieved my Handvisor today, and have struggled getting it to work, and found no good HOWTO yet, so thought i'd make one. By some random mutation, you might be able to find a new version on my web page, but that is up to chance. Test your luck at:

http://www.calvin.edu/~rvbijl39/

or

http://usbvisor.sourceforge.net

I used the vi editor. Anything else would have been evil. No spell check has yet been run on this document. So, if you have a problem with my spellling, tuff luck.

1.2 Disclaimer

Hmm... what did I claim? Oh well. I am really a newbie at this usb/visor stuff, so i really couldn't answer to many of your questions. There are people who could help you much better, who maintain the sites I refer to at the end of the document. However, feel free to send me questions, or updates to this document. My email address is: rvbijl39@calvin.edu

1.3 History

- v0.1 − initial release
- v0.2 − HTML−ized version
- v0.3 − SGML−ized, Added modifications by Miles Lott
- v0.4 − clarifications, updates, suggested by Matt Shook, Steven Coffman, Miles Lott, and info from Greg KH.

1.4 What is missing from this document?

I actually "lost" my machine with USB capabilities, so I am no longer able to test the USB stuff out. It would be very nice if people could send in documentation on how to get PPP connections working. (Or any other related cool thing working).

1.5 Copyright

2.Installing Stuff

2.1 Requirements

Some obvious things are required, such as a computer, USB ports, hand visor (and USB cradle), Linux, etc. You also should know how to compile and install stuff, like the kernel. If you do not, perhaps the kernel−HOWTO will be helpful: http://www.linuxdoc.org/HOWTO/Kernel−HOWTO.html.

2.2 Kernel Stuff

First, you will need to get a kernel which supports USB. You can either get the "latest" development kernel, which is version 2.4.0−test5 (as of this writing, Aug 7, 2000). You may get the latest version from ftp://ftp.kernel.org/pub/linux/kernel/v2.4/. Otherwise you need to get the latest stable kernel version, and the latest backport of the USB code. The latest (as of Aug 7, 2000) stable kernel is 2.2.16, and can be gotten from ftp://ftp.kernel.org/pub/linux/kernel/v2.2/. The latest backport can be gotten from the linux−usb project, and is found here: http://www.suse.cz/development/usb−backport/ The current version (Aug 7, 2000) is: usb−2.4.0−test2−pre2−for−2.2.16−v3. There may be patches to the backport available. You should be able to find them at http://sourceforge.net/patch/?group_id=1404. Download and install the source, including the usb back port if you are using version 2.2. Below shows an example of installing the the 2.2 source code, and the backport.

```
cd /usr/src
tar xzvf linux−2.2.14.tar.gz
cd linux
gzip −dc ../usb−2.3.50−1−for−2.2.14.diff.gz | patch −p1
make distclean
```

Next, a critical step, is determining which type of USB host controller you have. The USB host controller is the hardware in your computer which handles USB input/output. Intel chipset motherboards, i.e. the BX chipset, are typically UHCI controllers. Most addon cards are OHCI controllers. You can determine the type of USB hardware available by using: 'lspci −v'. (See the Links/References if you do not have lspci). If you see something like:

```
    USB Controller: ......
    Flags: .....
    I/O ports at ....
```

Then you have a UHCI based controller. If you see something like:

```
     USB Controller: .....
     Flags: ....
     Memory at .....
```

Then you have a OHCI based controller. (Documentation to figure out type type of controller was leached from the kernel documentation, Documentation/usb/scanner.txt). You could refer to http://www.linux−usb.org for further details. The kernel documentation in the directory

(Documentation/usb/usb.txt) may also be helpful in determining which type of controller you have.

If you dont have a USB controller you need either buy the serial cradle for the Visor (what are you doing reading this?), or buy a USB−capable device (ie. add−on card or motherboard).

Now we need to configure, and make your kernel. During configuration, make sure you enable:

- USB support (CONFIG_USB),
- the appropriate controller − UHCI, alternate UHCI, or OHCI (CONFIG_USB_UHCI, CONFIG_USB_UHCI_ALT, or CONFIG_USB_OHCI)
- USB Serial Converter support (CONFIG_USB_SERIAL)
- USB Handspring Visor Driver (CONFIG_USB_SERIAL_VISOR) (serial converter's sub−option)

(I'm not sure what the difference between UHCI drivers are, nor which one is recommended for use. Anyone?) I have compiled these into the kernel, and as modules. Both work. The developer(s) prefer the module method, but that is up to you.

After you did that, finish installing your kernel, and reboot.

Not done yet! To make the Visor driver work, we need to finish making the USB Serial driver to work. For that, you can refer to the Linux Kernel Documentation: Documentation/usb/usb−serial.txt. Basically, what it tells you to do is to make the devices. You can do this by excuting the commands, as root:

```
mknod /dev/ttyUSB0 c 188 0
mknod /dev/ttyUSB1 c 188 1
etc...
chmod 666 /dev/ttyUSB*
```

Right now the code is limited to 16 connections/ports/devices, so only 0 to 15 need to be done. The chmod is to allow users to be able to access the Visor device. It is the opinion of the author of this document that this should be save for a personal computer. Multiuser computers may want to look into this.

When a Visor connects, there are two "ports" opened. (For most people, this will be /dev/ttyUSB0, and /dev/ttyUSB1 or similar). The first port (zero), is a generic connection. The second port is the hotsync port. This feautre allows for future developments, like, for example, to export a filesystem from the Visor. A usefull thing to do is to create a link to the hotsync port so that synchronization software will use the approriate device be default. You can do this by:

```
cd /dev
ln −s /dev/ttyUSB1 pilot
```

Just for fun, you might also want to create a link from /dev/visor to /dev/ttyUSB1, just, well, because we have a visor, not a pilot (or Palm(tm)). The actual device number may change, depending on how many (active?) USB−serial devices you have on your system. A message containg the device actually used is displayed when the craddle hotsync button is pressed.

For people who compiled the USB code in as modules, you will also need to insert the modules into the running kernel. As root, run:

```
/sbin/modprobe usb-uhci
/sbin/modprobe usb-ohci
/sbin/modprobe usb-serial
```

2.3 Installing ColdSync

ColdSync is a software package which contains programs to talk with you're Visor (or your Palm(tm), if you have one of those). Another relatively popular package to do this is pilot−link. However, there seems to be some bugs in how pilot−link works, especially with the USB drivers. Thus ColdSync is the recommended package. UHCI−based systems seem to be working completely fine with coldsync, while OHCI−based systems still seem to have some problems. (Both are worse using pilot−link). For details, look at the visorusb devel list: http://www.geocrawler.com/lists/3/SourceForge/1169/.

As of this writing (aug 7, 2000), the latest stable version of ColdSync is 1.2.5. The latest version (as far as I know) of pilot−link, is 0.9.3. The one advantage of pilot−link is that there are more graphical interfaces which use the pilot−link libraries than coldsync.

The programs can be found here:

- http://www.ooblick.com/software/coldsync/
- ftp://ryeham.ee.ryerson.ca/pub/PalmOS/pilot−link.0.9.3.tar.gz or perhaps: http://pilot−link.sourceforge.net/

Download and install the program you wish to use. (coldsync has some INSTALL notes, which you should read.)

```
cd /usr/src
gzip −dc coldsync−1.2.5.tar.gz | tar xvf −
cd coldsync−1.2.5
./configure
make
make install
```

If you choose to be lazy, there are probably redhat and debian packages which you may download and install. I dont support those.

2.4 Almost Done! (aka testing)

We are pretty much done. Now we need to test it. Of course, the visor needs to be in the cradle, and the cradle needs to be plugged into your USB port.

To test, we can use the coldsync. For the computer to realize that the visor really is there, you need to hit the hotsync button on the cradle. (When a USB device connects, I've noticed that a bunch of info is dumped to the screen. Update: this is when debugging is enabled). Eventually, the visor driver will make an entry into the /proc/drivers/visor, which will list the minor number and purpose. More on that later. Then you can run coldsync:

```
coldsync −p /dev/visor
```

(Remember, /dev/visor is a symbolic link to /dev/ttyUSB1). From here, use other documentation. Manpages are useful.

3.Links/References

For this document, I used:

- ColdSync: http://www.ooblick.com/software/coldsync/
- PalmOS HOWTO: http://www.orbits.com/Palm/
- Pilot−link: ftp://ryeham.ee.ryerson.ca/pub/PalmOS
- Pilot−link: http://pilot−link.sourceforget.net
- Linux Kernel: ftp.kernel.org/pub/linux/kernel/
- Kernel USB backport: http://www.linux−usb.org
- USB Visor page: http://usbvisor.sourceforge.net
- lspci tool: http://atrey.karlin.mff.cuni.cz/~mj/pciutils.html