ASK KLAUS!

Klaus Knopper is the creator of Knoppix and co-founder of the LinuxTag expo. He currently works as a teacher, programmer, and consultant. If you have a configuration problem, or if you just want to learn more about how Linux works, send your questions to:

klaus@linux-magazine.com.

Fedora Core 5

I recently installed the Fedora Core 5. I installed the productivity version on an AMD Athlon 1300 using a Lite On DVD drive. Although the installation went fine, I cannot mount the drive for any other use after the install.

An error message would be helpful here, but I am guessing that:

1. There is a mountpoint (empty directory) that is supposed to be the location where you can see what's on a mounted DVD. Usually this is */media/dvd* or some similar path.
2. There is a device file that matches this drive. A DVD writer attached as master on the second IDE/ATAPI slot is usually */dev/hdc*, whereas a SATA drive would be */dev/scd0* or */dev/sr0*.
3. Look into the text file */etc/fstab* and check for the presence of a line approximately like the following:

```
/dev/hdc /media/dvd ➥
iso9660 ro,users,noauto 0 0
```

The meaning of this line is something like, "The content of the DVD in device */dev/hdc* can be seen at */media/dvd* after any user has mounted it read-only (ro), but it will not be mounted automatically on boot (noauto), and it will not be backed up or checked for invalid data."

If such a line is present in */etc/fstab*, mounting the DVD from an unprivileged account should work fine:

```
mount /media/dvd
```

Some hardware detection tools automatically add this line as soon as a medium is inserted. Some desktops even create icons for easier mounting with a few clicks.

If you check your desktop *Properties*, you can see if the correct device and mountpoint were used to mount the drive, and you can change them if they are set wrong.

Probing for Drivers

Can you please explain how Knoppix finds out which drivers to use for a given PC hardware configuration. I had visions of a great database table in cyberspace that listed the driver(s) needed for each piece of hardware. Only in my dreams! Probing components can extract the manufacturer's name and any other data that the manufacturer cares to include, but how do you determine what's needed to make the component work? Do you think the day will ever dawn when any PC will work easily with printers and scanners in any Linux distribution?

The file you are searching for is */usr/share/misc/pci.ids*. (See also *http://pciids.sf.net/*.)

Most hardware components have a PCI vendor and product id written as two hexadecimal numbers. Type

```
lspci -n
```

or, for more verbose output:

```
lspci -v
```

This is also true for devices that are not really PCI.

With older kernels, the kernel had its own kind of *pci.ids*, with the product names included in its code. Since there are a LOT of hardware components out now, this approach turned out to be a waste of space, so a new scheme was implemented that kept the human readable table outside of the kernel, and the vendor and product ids are read from the system bus instead.

For hardware detection, we still need a mapping between those PCI ids and the name of a kernel module to load with modprobe. Knoppix/hwsetup uses a modified *pci.ids* text table to lookup module names from the PCI ids and simply loads the modules listed in there. But this is only done for special cases that either don't have a mapping or are detected wrongly by other tools.

The Udev daemon, which is the modern "plug & play" of Linux hardware detection, uses a feature that has been built in into the kernel, or rather, into each individual module, since some versions ago.

Do the following:

```
modinfo ipw2100
```

(If you don't have this module, chose something different). Apart from some maybe useful information about module parameters, license, and authors, this command also shows a list of pci aliases for this module (i.e., which cards are supposed to be handled by this module).

Whenever Udev scans the system and detects hardware that has not been activated before, it will scan the modules list for a match in PCI vendor and product ID and load the module containing the "best match" for this card.

Sounds good? Unfortunately, the perfect driver doesn't always win the Udev contest for deciding which module to load for a hardware component. Sometimes, Udev loads a module that claims to know about the device but really doesn't. Sometimes there are concurrent modules for the same device. We are currently investigating how to handle these cases for wifi cards in Knoppix, since they are also managed by Udev.

Sometimes, when there are only a few choices (such as USB controllers), the best way seems to be just to load all available modules one by one and let those modules that don't match fail silently. For Knoppix, you can check the */linuxrc* script for this plain and simple trial-and-error based hardware detection if no fancy hardware scanning or module loading system is available.

Audacity Problems

Hello, I just started reading Linux Magazine a couple of months ago, and I find your articles very interesting. I have a problem with Audacity.

I can get Audacity to work upon a fresh install of the system, but after a day or so, I get a message that says, "There was an error initializing the audio I/O layer. You will not be able to play or record audio." I have tried to uninstall then reinstall, but I get the same message. Also, when I hit OK and the program starts, I go into *Edit* > *Preferences* to look for the I/O devices, but there are none. How can I get Audacity working? Or the question might be, how do I get the I/O devices back into Audacity? Why do the I/O devices disappear after a couple of days?

I noticed a similar problem, which has only occurred since Udev appeared in virtually all Linux distros.

Traditionally, "everything is a file" in Unix/Linux. This philosophy means that you'll find a device file for almost every hardware component (except network devices, which are a different subsystem) in the */dev* directory. Try:

```
ls -l /dev/dsp /dev/mixer
```

This command will give you a list of the device files used by most programs for audio playback and recording (except for the newer ALSA sound devices, which are in */dev/snd/**) .

Each of these device files must have a matching driver (or rather, a kernel module) activated.

If the device files have been removed or replaced by a malfunctioning program or script, you try to recreate them as root, for example, attempt recovery with:

```
mknod /dev/dsp c 14 3
mknod /dev/mixer c 14 0
chmod 666 /dev/dsp /dev/mixer
```

Try the following:

```
dd if=/dev/dsp of=/dev/null ➥
bs=4k count=1
```

(this records 4 kilobytes of audio data from the sound card) If you do not get an error message, there is probably a working driver behind the device file. If you get an error message like "error opening device */dev/dsp*," then you need to load the right module for your sound card. Loading the right module is the tricky part; you will need to find out which module to load, run *modprobe modulename* to load the driver, and add *modulename* to */etc/modules* to have it autoloaded at boot time.

It is easiest to run the preferred hardware detection tool for the distro of your choice to find out which module matches the sound card.

Now, as I mentioned, this is the traditional way. With the new Udev hardware management system, the *udevd* daemon will (or at least *should*) load the right module for your sound card automatically, and it also *should* create a device file in */dev*, so you *should* never need to type the *mknod* commands.

But if *udevd* fails to recognize the card, no module will be loaded, and the device file will also be missing. The files will appear as soon as you load the sound module for your card. This could also be an explanation of why devices are disappearing. If you have upgraded your distro and are now using Udev, old automatic scripts may still be trying to unload "unused" kernel modules. For Udev, unloading a driver also means removing the device file, and when there is no device file for sound, the kernel also won't autoload the sound module if someone accesses the file.

I would try the following steps to find out why sound support quits working after a few days:

1. Ensure that the sound module for your card is loaded.
2. Ensure that */dev/dsp* and */dev/mixer* are present. If they are not but */dev/snd/** device files are there, you probably need to additionally load the ALSA -> OSS bridge modules:

```
modprobe snd-pcm-oss
modprobe snd-mixer-oss
```

   If udevd is running, it should now create */dev/dsp* and */dev/mixer* for you.
3. Check the settings of Audacity (or other multimedia tools) for the presence of a setting that contains an existing sound device. An empty field does not always mean "use the default." When in doubt, try */dev/dsp* as the sound device for recording and/or playback. If your program is ALSA-compatible, you can also use the ALSA devices in */dev/snd/*.
4. Check that the sound devices are not locked up by another program or a sound daemon, like artsd or esd.

Open Source Hard Disk Tools

First of all, you are doing a great job with Knoppix. It is my absolute favorite Linux live cd.

Although our office is running completely on an Open Source thin client network, a problem we are facing is the gradually increasing pile of unusable hard disks.

I would be interested in knowing if there any Open Source tools in Linux that can recover data and fix broken hard disks?

Fixing hard disk errors with a pure software solution is difficult to accomplish with an easy-to-use, all-in-one GUI, because the tool always requires some knowledge of the hardware and the way data is stored on the disk. There are some mostly console-based tools you can use for different parts of the data rescue.

Start by making a 1:1 copy of a hard disk partition to work on. This should be one of the first steps, because with "partly broken" hard disks, each access to defective sectors, even if read-only, can increase damage. In general, you should always work with a copy of the data you wish to rescue.

```
dd-rescue /dev/hda1 hda1.img
```

hda1.img is now a bitwise copy of the first partition of your master IDE hard disk attached to the first IDE controller. File system repair tools can now work with this copy, instead of with the broken hard disk. This is vitally important because defective data structures sometimes cannot be repaired on the broken drive because of non-writable sectors. Using the fully-writable copy instead will enable hard disk repair tools to actually change the file system into a mountable stage (at least, this is the goal of most repair tasks).

If the partition table of the disk is broken and cannot be repaired by *testdisk*, which is an excellent tool for hard disk geometry analysis and partition repairing, you will have to copy the entire disk with *dd-rescue* instead of only the partition you want to save.

Depending on the file system type that has once been hosted on the rescued partition, you could attempt to run the filesystem-dependent repair tools, such as dosfsck for DOS/FAT12/FAT16/FAT32 filesystems, e2fsck for ext2 filesystems, or reiserfsck for Reiserfs. You may have

to use *--rebuild-tree* and *--rebuild-sb* with the aforementioned copy of the partition as an argument. Caution: if the partition is not a block device, some kind of "force" option is sometimes required to let the program work on a file instead.

If the file system structure is successfully repaired, it should now be possible to loopback-mount the file system copy:

```
mkdir -p /media/rescue
mount -o loop,ro -t ➥
filesystemtype hda1.img ➥
/media/rescue
```

This allows you to access the file system tree directly and make a tar archive:

```
cd /media/rescue
tar -zcpPvf ➥
/tmp/rescued-hda1.tar.gz ./
```

In some cases, you won't get the file system structure back. If a btree+ filesystem like reiserfs or xfs breaks badly, you will have data that was once inside files and directories now scattered all over the place, and you will have to search manually for the beginning and end of files. This is the point where data rescue can get time-consuming and expensive, because what you now have to do is search for the beginning and end of logical data structures and put them back into files manually, and file data is not always present in a continuous block. If you are lucky, there are helper tools for the requested type of data, such as photorec, which searches for JPEG-type content and movies on a raw data blob.

When searching for arbitrary types of data, you will sometimes have to use an on-disk hex editor such as lde or hexedit, search for file type signatures, and use your best guess of how long the file once was when saving data back into files. This process can be very difficult, because with journaling file systems, there are often several copies of a single file on a partition in various states of completion. You will have to find and save them all, then see which one looks OK. If you know Perl, it may be helpful to automate this process with a script. ■

Send your Linux questions to klaus@linux-magazine.com.