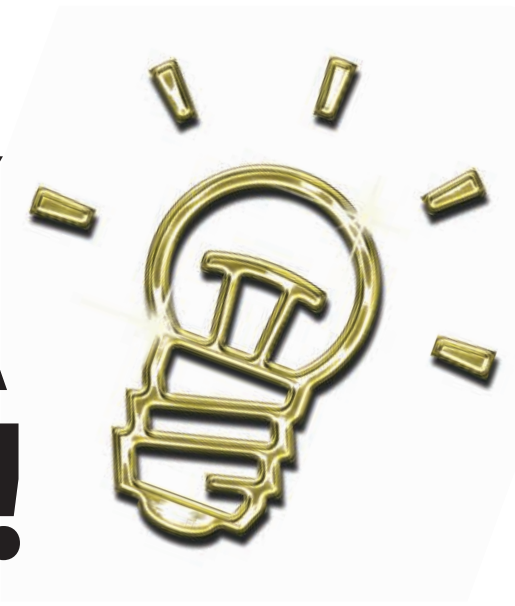
ASK KLAUS!



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

WiFi

? Thank you for the reply to my email asking for help with my laptop WiFi. I am using Ubuntu 8.04.1. */etc/wpa_supplicant* contains no *wpa_supplicant.conf*, but */etc/network/interfaces* contains lines that look correct. And *sudo wpa_cli status* also looks correct. I have no idea how to do the real serious bit, which is check if the DHCP request is coming too early or coming at all. I use *System* | *Administration* | *Network* to enter the correct type and password, but something appears to be resetting them to a default.

With modern computers, things sometimes get changed without me knowingly doing anything. The WPA type and password is the only one that causes me serious trouble, except that I reinstalled Ubuntu 8.04.1 this week and I can only connect to the internet by cable, not WiFi. I live in hope that I am learning things nearly as fast as they change!

Indeed, lots of things designed to make configuration easier for users tend to make the system more complex and introduce more possibilities for errors.

I'm not sure what exactly your desktop's network configuration GUI at *System* | *Administration* | *Network* does exactly, but usually with Debian-like systems, it should just manage entries in */etc/network/interfaces*, rather than maintain its own configuration files. Your WiFi setup might be correct, even without a *wpa_supplicant.conf* file; *wpa_cli* might change parameters for WiFi/WPA dynamically.

Recent Debian and Ubuntu versions support an extended syntax in */etc/network/interfaces* that looks like this:

```
auto wifi0
iface wifi0 inet dhcp
 wpa-driver wext
 wpa-ssid <your_essid>
 wpa-ap-scan 1
 wpa-key-mgmt WPA-PSK
 wpa-psk <your_wpa_key>
```

The command

```
ifup wifi0
```

will call *wpa_cli* with the parameters from */etc/network/interfaces* to change the running *wpa_supplicant* configuration to match your WPA setup. Note that *wpa_supplicant* might be running. A line like

```
pre-up pidof wpa_supplicant ⮐
>/dev/null 2>&1 || ⮐
wpa_supplicant -u -B
```

directly below *iface wifi0...* in */etc/network/interfaces* will start *wpa_supplicant* if it's not running already.

Now about the things that could go wrong: *wpa-driver wext* is used for all WiFi adapters that support the Linux wireless extensions (the vast majority of chipsets supported by the kernel do). However, you might have to use a different driver in rare cases; consult the *wpa_supplicant* man page and help files. Not all "old" drivers are compiled into current *wpa_supplicant* versions, though. Also, *wpa_supplicant* might not be able to get the card up and scanning for a signal before the DHCP client starts. You can check with the command

```
iwconfig wifi0
```

(use your card's name instead of *wifi0*) to see if the WiFi adapter tries to associate with an access point. Seeing the MAC address of an access point in the output and a signal strength of 70% or more is a good indication.

The DHCP client (*dhclient*) could fail to get an IP address from the DHCP server running on the access points. Some WiFi adapters have to be brought to an "up" state before they accept anything from the local network. The command for this is

```
ifconfig wifi0 up
```

without an IP address. This can be put into */etc/network/interfaces* as a "pre-up" directive, as demonstrated previously for *wpa_supplicant*.

Now the most annoying part is getting an IP address via DHCP. Although everything else seems to be set up correctly, another service might be hogging the WiFi adapter – probably NetworkManager. Because of its easy auto-configuration and handling with a desktop applet (nm-applet), it has become very popular in many distributions. NetworkManager runs its own DHCP client, which is

```
ps auxwww | grep dhclient
```

for current or

```
ps auxwww | grep dhcdbd
```

for older (0.6.4) versions. Because NetworkManager's own DHCP client keeps the interface in a "scanning" state, everything you set manually (static IP addresses) or dynamically via *dhclient* will just "disappear" again after a few seconds.

To manually configure WiFi (rather, "semi-automatic" configuration with ifup), make sure that NetworkManager is not running

```
/etc/init.d/network-manager stop
/etc/init.d/dhcdbd stop
```

and that no other DHCP client is keeping the WiFi adapter busy.

If *iwconfig wifi0* shows no signal, it could be your WiFi adapter has a switch that is turned off. Some WiFi modules have an *rfkill = 0* option to ignore the setting of the switch, but for many adapters, you really have to run a current kernel that can enable the device when pressing the "WiFi" button.

UMTS Modem

I recently converted a friend of mine to Linux (Ubuntu), and he really likes it, but he needs to connect to the web via a BandRich C100 UMTS modem. I have tried every suggestion on the forums and nothing works – it will not connect. What am I doing wrong? Because of this little glitch, two other guys who were going to switch to Linux are hesitating.

UMTS modems just work like regular modems. Keep in mind that Linux does not need "drivers" for each and every device or brand name. Rather, kernel modules just need to support the associated serial device chipset that is used in the modem.

Recent kernels have support for this modem. When you plug it in, a file called *ttyUSB0* should appear inside the */dev* directory, which allows dialup programs to access the modem. If it doesn't, you need a newer Linux kernel.

Now you need to supply access information to the modem, as you do when installing under Windows. Because you are changing a system configuration that is global for your system, you need administrator access; use *sudo* or *su* to switch to the root account. Note that this is a one-time configuration task. Making the connection is very simple afterwards.

For a UMTS dialup connection, two configuration files need to be created independent of a dialing program. I use the manual shell method here to create the connection because it works everywhere, but you can transfer the same information to a graphical dialup program like kppp if you prefer a graphical tool over the command line.

As prerequisites, pppd and chat need to be installed. The file */etc/chatscripts/umts* will talk to the modem and should contain the lines shown in Listing 1. Be very careful about the spelling. Once it is called by pppd/chat, it

- sends the PIN number for unlocking the modem. Change *1234* to your specific PIN.
- sends the provider-specific setting *AT + CGDCONT = 1,"IP","internet"*. Depending on the mobile service provider you use, this can vary, so you need to ask your provider about the right setting if it is not stated in your connection information that was provided when you bought the modem.
- dials **99 1#*. This phone number is not real, but it starts Internet dialup for UMTS. The shorter number **99#* might work as well.

The second file you should place in */etc/ppp/peers/umts* is for configuring Internet protocol parameters (Listing 2). Among all the other parameters described in detail with *man pppd*, the line

```
user ""
```

indicates that this connection works without login/password authentication, which is true for all UMTS providers, as far as I know (authentication is done by your modem's phone number). Again, */dev/ttyUSB0* indicates the device file matching the modem. The rest just makes sure that the modem sets everything up for Internet access (default route, nameservers, etc.).

After installing these two files, the command

```
pppd nodetach call umts
```

Listing 1: /etc/chatscripts/umts

```
TIMEOUT 120
ABORT BUSY
ABORT "NO CARRIER"
"" ATE1
OK AT+CPIN=1234
TIMEOUT 4
OK-AT+CPIN?-OK AT+CGDCONT=1,"IP","internet"
TIMEOUT 120
ABORT ERROR
REPORT CONNECT
TIMEOUT 10
OK ATD*99***1#
CONNECT \c
```

Listing 2: Configure Internet Protocol Parameters

```
noauth
connect "/usr/sbin/chat -v -f /etc/chatscripts/umts"
/dev/ttyUSB0
115200
modem
persist
crtscts
debug
passive
noipdefault
ipcp-accept-local
ipcp-accept-remote
usepeerdns
user ""
```

will initiate the connection. Because of the *debug* in the second configuration file, you will be able to follow the progress of establishing a connection. Once you see lines with addresses, you can access services on the Internet, and you can end the connection with Ctrl + C in the window in which you started pppd.

Partitions

In spite of having a Linux box for a couple of years, I am essentially a "newbie." I have several books on the subject but have not found answers to my questions about formatting and partitioning hard drives.

I have purchased a USB external hard disk drive with a stated capacity of 1TB. I want to format it for efficient use of its great capacity.

Many years ago (I was running Windows 3.1 back then), I recall that the most efficient partition size was 128MB. In such a partition, a simple text file that took about 2KB would take something like 64KB on a 1GB disk with only one partition. As I recall, the issue had to do with the number of "slots" in the VTOC.

I suspect there is no reason to follow the same rules today, but what are the hardware and software considerations involved in formatting and partitioning a large disk drive for efficient use of the resources? Also, what are the values (e.g., 128MB) that guide or limit formatting and partitioning? I have not been able to find such information.

Also, this particular drive has some backup utilities that only work with Windows machines. They are in a single partition formatted for NTFS, which is not useful in my openSUSE 10.2 environment. What should I be doing to (1) get rid of the Windoze junk and (2) make this drive as useful as it can be in my Linux environment?

You should be able to find information about filesystems [(V) FAT, ext2, ext3, JFS, ReiserFS, XFS, and more] in the man pages of the corresponding *mk < filesystemname >* formatting tool and at developer sites. Also, the Wikipedia articles about specific filesystems are excellent.

For huge partitions, filesystems with a "static" file and directory index table (file allocation table) are probably not good choices. FAT32 and ext2 use such an index of files, and it can be exhausted quickly when the partition is being filled with millions of small files. Depending on the block and chunk size, files can physically have a larger space allocated than necessary. With static filesystems, it is possible to have 99% of space free, yet be unable to create a new file because the filesystem's file index is full. If using ext2, for example, you would have to plan carefully whether you will use the disk for many small files or for fewer huge files, then choose the *mke2fs -N* and *-I* parameters accordingly.

With newer filesystems, you no longer have a fixed index of file entries; the (former) file allocation table is instead spread across the entire partition as a linked list. This list is extended as needed and optimized for finding files quickly with balanced trees. Also, the maximum file size is no longer limited to a few gigabytes as it was in FAT32.

If you are using only Linux to access the disk, I would recommend ReiserFS, JFS, or XFS for most efficient use. Only consider ext3 if you don't plan to create more than about 32,000 subdirectories inside one directory because it has (almost) the same inode limitations as ext2.

NTFS, in theory, uses similar approaches, but you will have to install ntfs-3g to mount the NTFS partitions as read/write under Linux. However, many Unix-specific file types, such as symlinks, device files, and sockets, are not supported by NTFS. If you plan to use the disk for data exchange with Windows systems, for video processing, or for other multimedia files that don't require special file attributes or permissions, NTFS might not be such a bad choice.

As a data disk, having just one big partition will probably give you the most flexibility. You can get rid of junk on the disk by creating a new filesystem with *mkreiserfs*, *mke3fs*, *ntfsformat*, or the formatting tool that matches your filesystem of choice. To REALLY delete all data safely, you can first overwrite the entire disk with zeros or random numbers with

```
dd if=/dev/zero ⮐
of=/dev/diskname bs=1024k
```

or

```
dd if=/dev/urandom ⮐
of=/dev/diskname bs=1024k
```

or the *wipe* tool, which is also supposed to shred partition content in a nonrecoverable way.

Afterwards, because you have just invalidated the boot record and partition table, you can use *fdisk* or *cfdisk* to create new partitions on the disk before formatting it with the filesystem of your choice.

Printer Drivers

I like your Knoppix 5.1.1 best. It works on most of my machines. Knoppix 5.3.1 is more difficult and does not work on as many machines as Knoppix 5.1.1.

My question: Knoppix 5.1.1 does not have all the printer drivers installed I need. Is it difficult to add printer drivers to Knoppix 5.1.1 (e.g., for the HP Deskjet F2180)?

You need a current *hplip* package to fully support this printer (also to use the integrated scanner with XSane; it's an all-in-one printer/scanner device, isn't it?).

The commands to install *hplip* are

```
sudo apt-get update
sudo apt-get install hplip
```

but if you notice that this pulls in too many packages as dependencies – for example if your cups-based printing system is too old – you might try an older version

```
sudo apt-get ⮐
install -t stable hplip
```

to get your printer up and running. ■

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