# KNOPPIX HACKS™

Includes Knoppix on CD-ROM

## 100 Industrial-Strength Tips & Tools

O'REILLY®

Kyle Rankin

CHAPTER FIVE

# Put Knoppix in Your Toolbox

## Hacks 37–51

A system administrator's toolbox and desk tell you that his job requires that he wear many hats and use many tools. First, there are the hardware tools: screwdrivers, torx wrenches, CAT5 crimpers, duct and electrical tape, and of course, a hammer. Then there are the reference tools: books, manuals, CD-ROM documentation, and the secret phone number to upper-tier-vendor tech support. After these tools are the software tools: DOS boot floppies, driver CD-ROMs, BIOS flashing utilities, and a number of other special-purpose rescue disks. Lastly, there is the most important tool for a system administrator: caffeine.

Oftentimes, a system administrator doesn't know which of these tools he will be required to use in any given circumstance. "My computer is broken" or "the Internet is down" mean any number of things, so when an administrator goes to solve a problem, he often brings a number of different tools and is prepared for any problem.

Over time, a clever (or lazy, depending on how you want to look at it) administrator figures out how to consolidate all his most useful tools, just so there is less to carry. If you are this kind of administrator, Knoppix is for you. While a Knoppix disc is a great Linux demonstration tool for a new user and a great desktop environment for an average Linux user, in the hands of an experienced system administrator, it is the ultimate software toolbox for any number of auditing, troubleshooting, or emergency uses.

In this chapter, the hacks are roughly organized into two sections. The first section deals with tools that can help you administer a network. Knoppix not only comes with a lot of excellent tools to audit your servers over the network, but it can actually replace some of them in a pinch. The second section deals with tools that can help you administer local hardware. Here, you will find tools that help you clone, wipe, and scan hard drives, and probe hardware. The fact that all of these tools run directly from the CD means that you can take Knoppix to any computer on your network and turn it into a troubleshooting tool.

Knoppix might be worth considering as an option in a recovery plan. You could even create custom disks **[Hack #94]** that already contain any files and settings unique to your network, and any extra services or modules that Knoppix does not contain. Then, emergency failover from a crashed server would be as simple and as fast as booting a CD.

## HACK #37 Run Remote Desktops

Use a computer running Knoppix as a base for remotely controlling other computers on the network via rdesktop or xvncviewer.

System administrators often need to be in two places at once. You might be on the phone walking a person through a technical problem when you realize that it would be much simpler if you could perform the problem-solving steps yourself. You might need to perform the same task on multiple computers, such as a manual virus or spyware scan or software update, that requires some initial setup and then a lot of waiting. If you could access all of the computers at the same time, you could start on the second computer once the first got going. In any of these cases, you might want to remotely control the computer, and with Knoppix, you can connect to both Remote Desktop Protocol (RDP) and any Virtual Network Computing (VNC) server using software on the CD. This hack explains the steps and software required to turn any machine on the network running Knoppix into a mobile command center for remotely controlling all of the computers on your network.

### VNC

VNC is an open source remote access project created by AT&T Labs at Cambridge, U.K. VNC's original purpose was to enable remote access to computers running X from thin clients that could be disconnected and reconnected later from the same or another thin client. The fact that the software is open source and runs on a variety of different platforms (Windows, Linux, Solaris, and OS X among others) has made it rather popular to both system administrators, who want a single program to remotely control multiple platforms, and to programmers, who have improved the VNC protocol by adding encryption and compressions and incorporated it into their open source projects. The current open source version of VNC is maintained by the company RealVNC and can be downloaded for free from its site at *http://www.realvnc.com*.

Knoppix includes the Linux RealVNC client *xvncviewer* to connect to remote VNC servers. For system administrators who are familiar with that program, open a terminal, and type this command to connect to a remote VNC server:

```
knoppix@ttyp0[knoppix]$ xvncviewer server :display#
```

Now type the password for that server at the prompt. The *xvncviewer* program also has a large number of options to enable full-screen mode and control settings such as color depth. Use *xvncviewer* from the command line if you are already experienced with the program or your connection requires special options. If you're not comfortable with the command line or don't have one open, you can click on K Menu → Internet → More Programs → xvncviewer to launch a GUI that makes connecting to remote machines quick and easy.

If you are completely new to VNC or you plan on managing multiple connections at once, you might find the included KDE application *krdc* (for KDE Remote Desktop Connection) a better choice. *Krdc* allows you to manage multiple VNC and RDP sessions from a single easy-to-use program. To launch *krdc*, click K Menu → Internet → Remote Desktop Connection. When first run, you are presented with a simple window that prompts you to enter the address of the computer to which you wish to connect. For a VNC connection, this is as simple as typing the hostname or IP address for the remote computer, followed by a colon and the display name. Usually, the remote machine is running a single VNC session, so to connect to the machine at the IP address 192.168.0.1, type the following command and click Connect:

```
192.168.0.1:0
```

*Krdc* then prompts you for your connection type so it can choose the settings that best suit your connection (such as a lower color depth for low-speed connections). After configuring your connection, *krdc* next prompts you for the remote server's VNC password and, once it is provided, connects you. *Krdc* superimposes a small taskbar at the top of your VNC window that tells you which server this window belongs to and allows you to toggle full-screen mode and close or minimize the window. This taskbar is particularly useful if you are in full-screen mode and can't remember the key combination to escape it (Ctrl-Alt-Enter). If the taskbar bothers you, you can easily set it to hide by clicking the pushpin icon.

One nice feature of *krdc* is that it keeps track of servers to which you have already connected, and the next time you run the program you can quickly select your server from the drop-down menu. *Krdc* also saves session information, and you only have to enter settings, such as the resolution for the remote connection and the connection rate, once.

## RDP

Knoppix also comes with tools to connect to servers accepting RDP connections. RDP is a protocol used by Microsoft for its Terminal Services software to allow mouse, keyboard, and even sound channels to be accessed

remotely. The functionality to make at least a single RDP connection to a machine exists out of the box in Windows XP Professional, Windows Server 2000 and 2003, and NT Server 4.0 Terminal Server Edition. For instance, to enable RDP connections on a Windows XP Professional machine, click on System under the Control Panel and check "Allow users to connect remotely to this computer" under the Remote tab.

The primary client for RDP connections under Linux is the command-line program *rdesktop*. Like *xvncviewer*, *rdesktop* has a number of command-line arguments to tweak settings, such as color depth and desktop geometry, and even forward sound to your local machine. To reference all of these settings, run **`man rdesktop`** or visit the official site at *http://www.rdesktop.org*, but for most usage, simply type this command in a terminal:

```
knoppix@ttyp0[knoppix]$ rdesktop servername
```

If the remote computer accepts RDP connections, you are presented with a standard Windows login page. Once connected, you can toggle full-screen mode by pressing Ctrl-Alt-Enter in the *rdesktop* window or by passing the `-f` argument to *rdesktop* when you start it.

Similar to VNC connections, RDP connections are also managed within the *krdc* program in much the same way. The primary difference is the syntax used for the hostname. For VNC connections, the syntax is *hostname:display* or *vnc:/hostname:display*; with RDP connections, the syntax is *rdp:/hostname*. To connect to a machine running at 192.168.0.1 at the prompt, type this command and click Connect:

```
rdp:/192.168.0.1
```

You are prompted for the resolution to use for the desktop and are then presented with the login screen. This presents a similar result as *rdesktop* only with the *krdc* taskbar appearing along the top of the screen, allowing you to toggle full-screen mode and a few other settings. These sessions are saved with any VNC sessions, making *krdc* an excellent choice for system administrators who are on a mixed network of VNC and RDP servers.

## NX Server

Knoppix 3.4 has also introduced a suite of tools to connect to NoMachine's NX server. You can use the NX server to create encrypted and compressed remote connections to X, VNC, and RDP servers, which are responsive even over a dial-up connection. NoMachine's NX client and other software included on Knoppix are licensed under the GPL, but it is worth noting that the NX server does require that you purchase a license from NoMachine. Further information about the NX server can be found at *http://www.nomachine.com*.

If you have an NX server to which you wish to connect, start the NX Connection Wizard by clicking K Menu → Internet → NX Connection Wizard. The wizard asks you a series of questions about the server's IP address, your connection type, and the protocol the remote connection is using to share the desktop. Fill out the information in the wizard to see the NX Client login window, and the session for the server you have just configured is selected in the drop-down Session menu. Type in your NX server login and password, and click Login to connect to the remote NX server and start your remote desktop connection. For further help with using the included NX software, Knoppix has a direct link to NoMachine's support page that is accessible by clicking K Menu → Internet → NX Help on the Web.

## Share the Local Desktop

Knoppix also supports sharing its own desktop with remote users by using the VNC protocol. This is useful when you find yourself talking someone through repairing a system that is unable to boot. The machine is unbootable so you can't take advantage of any remote control utilities the computer may already have. You know that with Knoppix, you can use some of the advanced system-recovery tools to fix the system, but it might be difficult to talk the user through all of the commands (not to mention that there is always a potential for typos that could cause further damage). If the user has a Knoppix disc (plan ahead and hide a copy under every user's machine), then she can boot and get network access. You can then walk her through the simple steps of sharing her desktop, and remotely connect and finish the system recovery.

Sharing the local Knoppix desktop is pretty simple. The user's first step is to run the Desktop Sharing applet by clicking K Menu → System → Desktop Sharing. Have the user click "Create Personal Invitation…" in the main window to create a personal invitation to share her desktop, which then displays a new window containing the address and the temporary password to use for the connection. This information can be entered into any VNC-compatible client on the remote end, causing a prompt to appear on the local user's screen and requesting the user to accept the remote desktop connection.

The user can also click "Invite via Email…" to send an email containing connection information to the system administrator. This email provides a direct link to click on if the recipient is running a KDE desktop with *krdc* installed. The sender must have an email account set up on her Knoppix machine for this to work.

The randomly generated password expires after an hour, so any new connections after that point require creating a new invitation. To remove an invitation before it expires, click "Manage invitations" on the main Desktop Sharing Wizard screen to see all current invitations, along with options to delete them and create new invitations. The Desktop Sharing Wizard makes sharing your current KDE desktop pretty easy even for people new to VNC or Linux, and it is simple to explain to users over the phone or through email.

With all of the different remote desktop protocols Knoppix supports, along with the fact that it includes a simple method to share its own desktop, you might find it worthwhile to hand out an emergency Knoppix CD to friends or clients for those times when you need to do some quick technical support but are unable to physically be there. If a client has a network of machines that needs support, you can use the desktop-sharing feature of Knoppix to connect remotely to a machine on the network booted off of Knoppix, and then use that machine as a remote command center to connect to the rest of the machines within the network. This allows you to support all of the machines from a single remote connection.

## HACK #38 Run X Remotely with FreeNX

Use FreeNX to connect to a remote desktop that's responsive even over a slow dial-up connection.

Before exploring the technical details of NX, you should run the test drive first to see the performance NX offers; NX Client 1.3.2 is already included with Knoppix 3.4. To start NX Client, click K Menu → Internet → NX Client for Linux.

NX technology is new. It's actually so new that the newest developments are (at the writing of this book) not yet included in Knoppix. However, this hack gives you an overview of what to expect with the Knoppix Version 3.6.

The NX Connection Wizard starts and allows you to create a new session. It asks for a name for the session, *nxserver* host, and, optionally, a port, which in most cases is just the SSH port (22). You can also select the speed of your connection. Even if you have a very fast connection, it's worthwhile to try modem speed first.

If you don't have an NX Server to which to connect, visit NoMachine's web site at *http://www.nomachine.com/testdrive.php* and sign up for its test drive. You'll receive an email with details on how to connect to its test server.

You can select the type of the connection (Windows, Unix, VNC) and the preferred Desktop in the next step. NX can connect to other servers at the backend, so it's also useful as a secure and fast gateway to Windows or VNC machines.

You can use a full screen (in which case, you can click on the top-right pixel to minimize the session) or a specified size for the session window. You can also select "SSL encryption" to tunnel all traffic over SSH.

The advantages are clear:

- You don't have to open any port other than SSH, which, in most cases, is open for remote shell access anyway.
- Users don't have to fiddle with complicated SSH client and forwarding setups. Just installing NXClient is enough. NXClient is of course available for all major operating systems, such as Windows, Linux, and MacOS X.

As the last step, you can choose to create an icon on your desktop for that session automatically (which is recommended) and to configure advanced options.

Don't worry; you can always select the advanced options dialog from *nxclient* later.

As soon as you've finished, you should see the Login dialog of NX Client. Open up a web browser to *http://www.nomachine.com/testdrive.php*, and then enter your name and email address. Some minutes later, you'll receive an email with your test-account data to enter in a test drive's client.

Insert the account data and press Login. The NX Client then creates a connection, authenticates the user, and establishes the X-Server connection. Then a window appears and a normal KDE session is started—in Italy.

If the user authenticates, but it then times out, try to activate SSL encryption by checking "Enable SSL encryption of all traffic" on the Desktop screen of the NX Connection Wizard, and then reconnect.

The session should be very fast, and you should be able to browse the Web, write email, and do your office work on it. Indeed, I do this regularly. Wherever I am, I can connect to my PC at home and graphically read my email—even if it's just a modem uplink.

## The NX Technology

How can NX achieve this speedup of X?

There are five major reasons:

*Very efficient X Protocol compression*
: The X Protocol is highly compressible. Each X-Request or Confirmation has a fixed part and a variable part. With Differential X Protocol Compression (DXPC), you can transmit what has changed only on the display, instead of the complete desktop.

*Caching of the protocol*
: The X Protocol compression makes it possible to cache the data to improve responsiveness. For example, in VNC it takes equally long to open the same menu multiple times while NX sessions get faster with time. For example, the first time the menu opens in NX it takes some time, but the second time, the menu just pops up as if it were opened locally. Also, due to a disk cache, this effect is also preserved if you start a new session.

*Round-trip suppression*
: A *round trip* in the X11 protocol is a request plus the wait for confirmation. While you can increase bandwidth without problems, it's not possible to reduce latency (as the speed of light, and in this case electricity, has a maximum speed).

  One round trip is tolerable, but imagine that you have to make 1000 round trips, and you have to wait each time for the answer over a link with high latency, which is very slow. This effect is especially bad with modern tool kits, such as QT or GTK, because they are typically programmed to run on the same machine—not over the network.

  NX solves round-trip problems locally by usage of an *nxagent* that groups requests and then sends them chunked to the client.

*Compression of X-Images*
: NX uses state-of-the-art compression techniques like PNG and JPEG to compress huge bitmaps. VNC uses this technique too, but VNC always compresses the entire screen along with fonts, because it cannot distinguish between the different elements on the desktop. With NX, just the X-Images are compressed, and the fonts and most other elements on the desktop are crystal clear.

*Chunking of image data*
: Image data is the biggest part of a desktop to be transferred over the network. Even if it's possible to compress it, you still want to use the desktop while a huge image transfers. NX never uses all of the bandwidth and always has a small control channel so that it can stop the transfer of

the chunked images to react to a mouse click or similar events. As a result, the desktop is more responsive.

## Set Up NX Server

NoMachine sells a commercial server with support but has also put all core components under the GPL, which allows anyone to write a free server, which I did.

The following instructive details were not programmed at the writing of this book, so it is possible that the actual program differs in some ways from what is described here.

To set up the NX server, click K Menu → KNOPPIX → Services → Manage NX Server. This informs you that you are starting a service that allows other users to access this computer. The server then creates a user called *nxfree* and starts the SSH service.

Then it starts an interface, and you can manage your server:

*Add user*
: Before users can use your NX terminal server, add them to your server.

*Remove user*
: If you no longer want a certain user to use your NX server, remove her.

*Stop server*
: Stop the NX server.

*Quit*
: Quit the management program but leave the server running.

As it is not currently clear how the NX server interface will handle these functions, the following shows you how to perform them from the command line.

## User Management

To add a new user *joe* to Knoppix, open a console and type:

```
knoppix@ttyp0[knoppix]$ sudo adduser joe
```

You are then asked a number of questions about this user, including his full name. Fill in the fields, and then choose a password for the account. Joe can now log in to this server with *ssh*. However, if he wants to use NX, you must activate his account for the NX server. First, I add *joe* to the NX user database, then I give him a password:

```
knoppix@ttyp0[knoppix]$ sudo nxserver --useradd joe
knoppix@ttyp0[knoppix]$ sudo nxserver --passwd joe
```

Joe can now use the NX Client on his laptop to connect to this machine.

## Server Management

The NX server has a number of command-line options:

`--help`
: Shows a small help page.

`--useradd`
: Adds a user.

`--userdel`
: Deletes a user.

`--userlist`
: Lists all configured users.

`--passwd`
: Sets a password for a user.

`--start`
: Starts NX server.

`--stop`
: Stops NX server. This option does not stop the SSH daemon.

`--restart`
: Restarts NX server.

`--status`
: Shows whether the server is currently running.

`--list`
: Each session that starts on the server receives a unique session ID. This option lists all running sessions.

`--terminate`
: Terminates all sessions for a user. Alternatively, you can terminate users based on the display number they use to connect.

`--suspend`
: Suspends a session to be reconnected later.

`--send`
: Sends a message to the specified user.

`--broadcast`
: Enables you to send a message to all connected users.

`--lock`
: Locks the display of a user.

`--unlock`
: Unlocks the display of a user.

NX can help you as a tool for remote administration. Knoppix offers the NX server so setting it up is very easy.

—*Fabian Franz*

## HACK #39 Browse Windows Shares

Tweak a few settings for the lisa daemon so you can graphically browse Windows shares on your network.

Microsoft has made it easy to browse for and connect to network fileshares. If you are used to browsing network shares under Windows, learning how to browse network shares under KDE might seem like a black art. Under Knoppix, the issue is aggravated by the fact that Knoppix does not automatically configure *LISa* (KDE's LAN information server) to work beyond the local machine. Luckily, it takes only a few steps to get network browsing up and running.

First, reconfigure LISa so it scans for all fileshares on your network. To do this, you must move *lisarc*, LISa's read-only configuration file, out of the way, and then start *kcontrol* as root so that you can create a new configuration file:

```
knoppix@ttyp0[knoppix]$ sudo mv /etc/lisarc /etc/lisarc.bak
knoppix@ttyp0[knoppix]$ sudo kcontrol
```

When the KDE Control Center opens, click Internet & Network → Local Network Browsing, select the LISa Daemon tab, and click "Guided LISa Setup." If your network is already correctly configured, either by Knoppix automatically or by you manually, the guided setup should provide you with all of the correct defaults. Just click Next through all of the options, and then click Apply at the bottom of the screen to save the changes.

With the new settings in place, start the LISa daemon:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/lisa start
Starting LAN Information Server: lisa.
```

Now click K Menu → Home, type **`lan://localhost`** in the location bar, and press Enter. The machines on the network that LISa detects should appear, named according to their IP address, as in Figure 5-1.

Click on a machine to see the different filesharing services it offers (Figure 5-2). Click on any of the folders to access files that the services offering. To access Windows shares, click on the SMB folder.

There are a number of filesharing services LISa supports, including FISH (filesharing over SSH), FTP, NFS, and SMB.

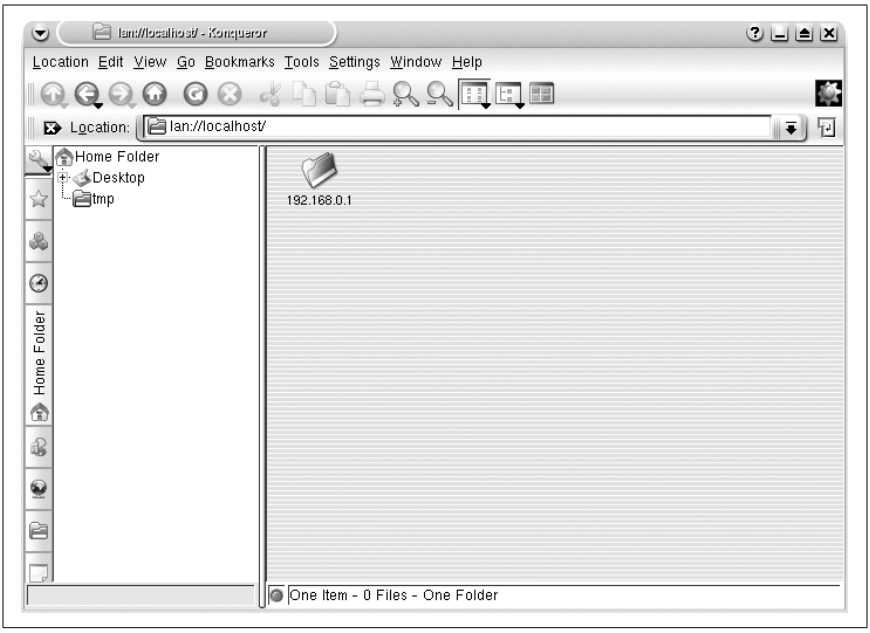


*Figure 5-1. Networks detected by LISa*

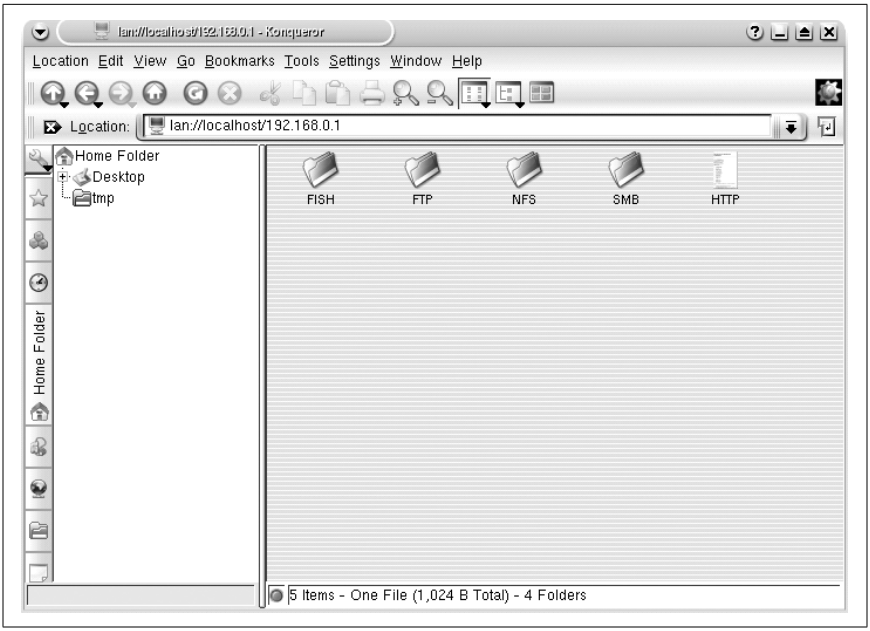


*Figure 5-2. View filesharing services*

# HACK #40 Create an Emergency Router

Turn Knoppix into a router or firewall.

Avoid thinking that Knoppix can be used only for demonstration purposes or is fit only for light desktop use. Knoppix is a full-fledged portable installation of Linux, which means it can do most anything an installed version of Linux can do. For instance, Knoppix comes ready to use as a fully functional router or firewall with all of the normal utilities, such as *route* and *iptables*, that you use on any other Linux distribution. These tools make Knoppix particularly handy if you need an emergency Network Address Translation (NAT) router or a bridge. When the router goes down, you can take your Knoppix "demonstration" CD, boot it on a spare machine with two NICs, and demonstrate how to save the day. With just a few commands, you can route across any of the network connections Knoppix supports from DSL to dial-up to wireless. This hack walks you through turning a machine into a bridge and then a NAT router.

## Configure the Network

The machine you are using as the emergency router must have two different network connections that already work independently of each other. This can be satisfied with two network cards, a network card and a modem, a network card and a wireless card, or any two network connections that Knoppix supports. Configuring network connections under Knoppix is covered in "Connect to the Internet" [Hack #17].

After both networks are working, you can link the two either with a bridge or with NAT. Generally, you want to use a bridge to connect two local networks so that machines on either network can communicate directly with any machine on the other network. Use NAT when you need to share a single Internet or network connection across a local network with the NAT machine acting as a sort of firewall. Machines on the other side of the NAT are not able to communicate directly with local machines unless you set up firewall rules on the NAT machine to forward ports.

To create either of these routers, you must enable IP forwarding in the Linux kernel. Most firewall and routing HOWTOs instruct you to do this by running the following command as root:

```
root@ttyp0[root]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

However, under Knoppix, you must change that command so that it works under the *sudo* environment by typing:

```
knoppix@ttyp0[knoppix]$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Now that IP forwarding is enabled, you can configure your bridge or NAT router.

If you are dropping this Knoppix machine in the place of a broken router, save a lot of trouble by giving Knoppix the same IPs as the previous router. In the case of a bridge, once you provide Knoppix with the same IPs and enable IP forwarding, the bridge is ready to go.

For the purposes of these examples, assume that the Knoppix computer is connected to two networks—`192.168.0.*` on `eth0` and `192.168.1.*` on `eth1`. Run *ifconfig*, and you should get the following output:

```
knoppix@ttyp1[knoppix]$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:DE:AD:BE:EF:00
          inet addr:192.168.0.5  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6918 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4678 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:675976 (660.1 KiB)  TX bytes:447963 (437.4 KiB)
          Interrupt:9 Base address:0xb800

eth1      Link encap:Ethernet  HWaddr 00:C0:FF:EE:00:00
          inet addr:192.168.1.5  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4933 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4988 errors:1 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:496574 (484.9 KiB)  TX bytes:749568 (732.0 KiB)
          Interrupt:3 Base address:0x100

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:33 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3016 (2.9 KiB)  TX bytes:3016 (2.9 KiB)
```

These networks already have a default route set up for each of these interfaces, which you can see by running the route command:

```
knoppix@ttyp1[knoppix]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
default         192.168.0.1     0.0.0.0         UG    0      0        0 eth1
```

## Build a Bridge

Creating a bridge with *route* is pretty straightforward once you see the commands involved. In fact, if both networks are already configured to use this machine as the gateway, and you have already enabled IP forwarding, then congratulations—you are finished! Otherwise, read the following instructions to learn how to configure the routing for your bridge.

So far I haven't had to change anything in the networking. In my example, I set up static IPs ("Connect to the Internet" **[Hack #17]**), but if you had DHCP running on either or both sides of the network with different default gateways, the bridge would have worked fine too. At this point, the Knoppix machine should be able to ping machines on both the `192.168.0.*` and the `192.168.1.*` networks, but machines on `192.168.0.*` shouldn't be able to ping `192.168.1.*` and vice versa.

I want to make the Knoppix machine the link between my two networks. For this to happen, the machines on either network must use the Knoppix machine as the bridge to the other network. If one of the two networks is already configured to use this Knoppix machine as its default gateway, then all packets going outside of the subnet route through it by default, and you don't have to bother with any extra routing for that network. If both networks are already set to use this machine as the default gateway, then you are finished. Either of these scenarios might be the case if you drop in Knoppix to replace a bridge and assign it the same IP addresses as the previous bridge.

If a network does not use the Knoppix machine as its gateway, you must add a route to the actual gateway on that subnet. This route tells the gateway to route any traffic going to the other subnet, through the Knoppix bridge. To route through the Knoppix bridge requires root access to the network's default gateway, to add the new route. In our example, the default gateways are `192.168.0.1` and `192.168.1.1`, respectively, so on `192.168.0.1`, run the following command as root:

```
root@ttyp0[root]# route add -net 192.168.1.0 netmask 255.255.255.0
gw 192.168.0.5
```

On `192.168.1.1`, run:

```
root@ttyp0[root]# route add -net 192.168.0.0 netmask 255.255.255.0
gw 192.168.1.5
```

Once you set up these new routes, machines on either side of the bridge can ping each other, and your bridge is complete.

## Network with NAT

Performing IP masquerading or NAT with Knoppix is as simple as configuring it as a bridge, if not simpler. NAT is commonly used to share a single public IP address (like you might get from a DSL or cable provider) with a local network behind the NAT router.

For NAT to work, all of the machines on the local network must be configured to use the Knoppix machine as the default gateway. In our example, the `192.168.1.*` network is behind this NAT "firewall" to access the `192.168.0.*` network, so each of the machines on `192.168.1.*` is using `192.168.1.5` (the IP address we assigned the NIC connected to the local network) as their default gateway.

The NAT works by taking all of the packets coming from `192.168.1.*` (the local network) and going to `192.168.0.*` (the external network) and making them appear as though they are from `192.168.0.5`—the IP address we assigned the NIC connected to the external network. When a machine on the external network responds, it responds directly to `192.168.0.5`. Then the Knoppix machine translates the address to refer to the `192.168.1.*` machine that originally sent the packet. Then Knoppix forwards it to the local network. For all intents and purposes, the `192.168.0.*` network doesn't know that the `192.168.1.*` network exists.

To set up Knoppix as a NAT router, you really only need to type in a single *iptables* command. To create a NAT for our example network, type:

```
knoppix@ttyp0[knoppix]$ sudo iptables -t nat -A POSTROUTING -s
192.168.1.0/255.255.255.0 -o eth0 -j SNAT --to-source 192.168.0.5
```

This *iptables* command creates a rule to take all packets coming from the `192.168.1.*` network and going from `eth0` and makes them appear as though they are from `192.168.0.5`. If you want to use IP masquerading instead of NAT (useful for forwarding over a dial-up connection that might drop while the computer is booted, which results in a different IP), type the following command instead:

```
knoppix@ttyp0[knoppix]$ sudo iptables -t nat -A POSTROUTING -o eth0 -j
MASQUERADE
```

Substitute `ppp0` for `eth0` if you are forwarding over a dial-up connection. At this point, you should be able access the outside `192.168.0.*` network from any of the machines on the `192.168.1.*` network.

The *iptables* command creates a NAT rule, but doesn't actually create a proper firewall. NAT does prevent people from easily accessing any local IPs behind the NAT router. However, if you are interested in setting up Knoppix with firewall rules suitable for your network, you can reference one of

the many great HOWTOs and tutorials on using stateful packet filtering under Linux with *iptables*.

### See Also

- The official netfilter page at *http://www.netfilter.org/documentation* (in particular, the packet-filtering HOWTO).
- The Advanced Routing HOWTO at *http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Adv-Routing-HOWTO.html*.

## HACK #41 Create an Emergency File Server

When files need to be transferred quickly over the network, Knoppix can serve as a quick makeshift file server.

There are many different occasions that might call for creating an emergency file server. For instance, you might have a file server with multiple drives that has suffered hard-disk failure on the root drive. All of the hard disks that have shared data are fine, but you need to still serve files while waiting on the replacement drive. This is the perfect environment for anyone wanting to learn how to configure Samba or start up a quick temporary file server. For experienced systems administrators, this makes it easy to share all the drives on an infected system for a quick virus scan by a centralized corporate virus scanner while the infected OS is shut down. You can even connect a spare machine to a printer, boot Knoppix, and quickly configure a makeshift network printer server.

Knoppix includes a nice GUI configuration utility to configure Samba (the Linux SMB file server). You aren't required to use the Knoppix Samba utility to set up a file server. If you already know which kind of configuration you want, simply edit */etc/samba/smb.conf* directly, and then run this command to start the Samba process:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/samba start
```

To use the Knoppix-provided Samba configuration utility, click K Menu → KNOPPIX → Services → Start Samba Server. Each time you run this script, it wipes out the previous Samba configuration, so don't run this script if you have made changes to *smb.conf* that you want to keep. This program prompts you to choose a password to assign to the *knoppix* user if there isn't already one, because Knoppix's default behavior requires a username and password to access any shares it creates.

After it confirms your password, Knoppix asks you whether you want to export all your hard drives so that remote machines can access them. If you answer "no" to this question, the script creates an *smb.conf* file that shares

your home directory and any printers that you have previously configured on the system. If you answer "yes" to this question, the script adds shares for all hard drives and CD-ROM drives that Knoppix has detected, and automatically mounts the devices for you as they are accessed. Once you click "yes" or "no," the script creates the *smb.conf* file and starts Samba.

If you only want to share all of the drives on your system as read-only over the network, then you are finished. By default, the share shows up in the `WORKGROUP` workgroup as a machine named `KNOPPIX`, and if you have shared all of the drives on your system, they appear with the same names they were previously assigned on your desktop, such as `hda1`. You can find the configuration for each drive at the bottom of the */etc/samba/smb.conf* :

```
[hda1]
    comment = /mnt/hda1
    browseable = yes
    path = /mnt/hda1
    writeable = yes
    preexec = /bin/mount /mnt/hda1
    postexec = /bin/umount /mnt/hda1
```

These shares all require that you log in with the username *knoppix* and the password you created for the *knoppix* account. Then Knoppix automatically mounts the drive when you access it and unmounts it when you are finished. Notice that by default, Samba is configured to allow writing to the drives, but Knoppix automatically mounts all drives read-only, so it overrides this, and you are still able to read from the drives. To allow write access to a drive, modify the `preexec` line in *smb.conf* to read:

```
preexec = sudo /bin/mount -o rw /mnt/hda1
```

and replace `/mnt/hda1` with the drive you are configuring. To allow anyone to access the share without requiring a password, add the following line to the share's configuration:

```
guest ok = yes
```

In general, you do not need to restart Samba for share level changes to take effect; however, changes you make to the global configuration (under the `[global]` header) require you to restart Samba.

To share something entirely different from the hard drives, add new shares to the default configuration. It doesn't matter which script option you chose earlier, just add the new configuration to the end of the file. The following example creates a new directory called *share* under the home directory *knoppix* and configures it as a guest share under Samba. First, create the *share* directory:

```
knoppix@ttyp0[knoppix]$ mkdir /home/knoppix/share
```

Then edit */etc/samba/smb.conf* as root, and add the following lines:

```
[share]
    comment = Knoppix share
    browseable = yes
    path = /home/knoppix/share
    writeable = yes
    guest ok = yes
```

Once you save your changes, the new share immediately appears, and anyone is able to view, add, and delete files in that directory. Keep in mind that this share is running off of your home directory that is in a ramdisk by default. The size of files stored here are limited by RAM, so for storing large files, you want to configure a share on the system's hard drives.

Remember that each time you run the Knoppix Samba configuration script, it creates a new *smb.conf*, and any manual changes are lost.

## HACK #42 Create an Emergency Web Server

When the web server goes down, use Knoppix to pick up the slack.

On first glance, Knoppix may not seem like a distribution to use for web serving because of its colorful GUI, the desktop applications, and the games. But included in this huge bundle of software is the complete Apache 1.3 server and a large set of modules that give you many of the common tools you need to turn Knoppix into a replacement web server.

Before you set up Knoppix as a web server, make sure Knoppix has all the tools you need for your web site. Look in */usr/lib/apache/1.3/* to see if the modules you need are there. Knoppix includes quite a few modules, including support for CGI, server-side includes, PHP4, *mod_rewrite*, and SSL, and also comes with MySQL so you can run a database-driven site. If you need to use any Apache modules, check */etc/apache/modules.conf* and make sure they are listed in there. Not all modules Knoppix includes are automatically listed in that file, so, for instance, to add server-side include support, add this statement to your copy of */etc/apache/modules.conf*:

```
LoadModule includes_module /usr/lib/apache/1.3/mod_include.so
```

If Apache doesn't have the modules you need, you must remaster Knoppix with your custom Apache setup **[Hack #94]**. Otherwise, the next step is to actually copy over the pages you want to serve, and configure Apache to use them.

If you choose, you can use your Apache configuration for your current server. This may mean restoring the configuration from tape backup if you cannot directly access it from its current hard drive. Simply copy your

complete Apache configuration to a suitable spot, like */home/knoppix/apache*, and use it by creating a symlink to it:

```
knoppix@ttyp0[knoppix]$ sudo mv /etc/apache /etc/apache.bak
knoppix@ttyp0[knoppix]$ sudo ln -s /home/knoppix/apache /etc/apache
```

Otherwise, you must edit Knoppix's Apache configuration. All of the Apache configuration in Knoppix can be found in a series of symlinks in */etc/apache/* that point to files on the CD-ROM, which are, of course, read-only. To make changes to these configuration files, you must first make them writable. The following step seems a little odd, but it breaks the symlink with the CD-ROM and gives you a writable *httpd.conf* on the ramdisk. You can repeat the process with other configuration files you need to modify.

```
knoppix@ttyp0[knoppix]$ sudo mv /etc/apache/httpd.conf /etc/apache/
httpd.conf.bak
knoppix@ttyp0[knoppix]$ sudo cp /etc/apache/httpd.conf.bak /etc/apache/
httpd.conf
```

With the configuration files now writable, you are able to modify */etc/apache/httpd.conf* and add any special changes you need to make for your site, such as adding multiple virtual hosts or changing the location of the *DocumentRoot* directory. Remember that when copying over the configuration and the accompanying web content, you must change any paths in *httpd.conf* to point to the new content directories that you have created. Also, if you are adding virtual hosts, remember to change the IP addresses to match this machine if necessary.

Once you have your files and configuration copied over, running Apache is as simple as:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/apache start
```

You shouldn't notice much of a performance hit for running off of the CD, because most of the site is running from ramdisk, and Apache itself runs completely from memory once it is loaded. However, there is less RAM to use overall because of the ramdisk Knoppix creates.

With this functionality, you can easily turn some desktop machines into mirrors of your web site or possibly even emergency replacements for the site while you change out hardware or perform software upgrades. The nice thing about using Knoppix for your emergency web server is that you can run it on top of any PC regardless of OS installation. When you are done, you can just log out and reboot the machine, and no one will know the difference.

# HACK #43 Run Other Emergency Services

It is easy to set up Knoppix as a DHCP, DNS, or MySQL server. This ability could prove useful in an emergency.

There are a number of other services that Knoppix includes that require only a couple of steps to get running. Most of the principles behind starting these services are the same—copy over a configuration and start the service. However, some of these services require a few more steps to get fully functional. Most of the services log to */var/log/syslog*, which Knoppix disables by default. To start the *syslog* service, click K Menu → KNOPPIX → Services → Start SYSLOG, which launches a terminal that displays live output of */var/log/syslog*.

## SSH

If you want to use Knoppix as a server for anything, you certainly want to be able to administer it remotely. Every administrator's favorite remote shell is openssh, and Knoppix includes it. It is incredibly simple to start the SSH service on Knoppix. Just click K Menu → KNOPPIX → Services → Start SSH Server. If you have not yet created a password for the *knoppix* user, the script prompts you to enter a new password so you can log in remotely. Alternatively, run:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/ssh start
```

## DHCP

DHCP allows you to automatically assign IP addresses to other computers on the network along with other basic network information. It's quicker than manually entering the network information into each computer. The DHCP configuration file in Knoppix is */etc/dhcp3/dhcpd.conf* and, by default, it is not configured to run on any network interface. First, back up the following file:

```
knoppix@ttyp0[knoppix]$ sudo mv /etc/dhcp3/dhcpd.conf
/etc/dhcp3/dhcpd.conf.bak
```

If you use this machine to replace another DHCP server on the network, simply copy the other machine's *dhcpd.conf* file to */etc/dhcp3/*. If you do not have a preconfigured *dhcpd.conf* to use, here is a simple template you can use to get started. Create this file with your favorite text editor, then copy it to */etc/dhcp3/dhcpd.conf* as root. Change the IP addresses to match your local network.

```
# how long the DHCP lease lasts in seconds
default-lease-time 600;
# maximum length of lease in seconds
max-lease-time 7200;
```

```
# name servers for clients on all subnets to use
option domain-name-servers 192.168.0.1, 192.168.0.2

##### here we put specific per-subnet options #####
subnet 192.168.0.1 netmask 255.255.255.0 {
   # IPs will be assigned between these two ranges
   range 192.168.0.50 192.168.0.99;
   option subnet-mask 255.255.255.0;
   option broadcast-address 192.168.0.255;
   # the gateway for the network
   option routers 192.168.0.1;
}
```

Once you have configured *dhcpd.conf*, start *dhcpd*:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/dhcp3-server start
```

If *dhcpd* is unable to start, view the syslog for details, including possible errors you might have made in *dhcpd.conf*.

## DNS

The name-resolution services provided by a DNS server are essential to any modern-day network. Knoppix comes with complete support for running your DNS server with the included BIND9 package. The simplest way to get your DNS server up and running is similar to the method used in "Create an Emergency Web Server" [Hack #42]. First, move */etc/bind/* out of the way with this command:

```
knoppix@ttyp0[knoppix]$ sudo mv /etc/bind/ /etc/bind.bak
```

Second, copy your complete BIND configuration (some distributions put it in */etc/bind/* while others put it in */etc/named/*) to your home directory, and symlink it so the system uses it instead:

```
knoppix@ttyp0[knoppix]$ sudo ln -s /home/knoppix/bind /etc/bind
```

Now start BIND by typing:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/bind9 start
```

Now your DNS server is up and running. If BIND does not start after this command, check the syslog for any errors it might have reported.

## MySQL

Databases are vitally important to most businesses, and a rising star in the database world is the open source MySQL database. This database has proven to be especially popular as a backend to dynamic web sites because of its low cost and amazing speed. If you have a MySQL database server that is down and need to run something in its place, you may be able to use

Knoppix, which contains the MySQL database program. To configure MySQL under Knoppix, first start the MySQL server:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/mysql start
```

There are different methods to import and export a database, and this section highlights methods to import to and export from a database using *mysqldump*. Of course, if you are creating an emergency Knoppix server because your database server is down, your importing methods are tied to whatever backup method you have decided to use.

If you want to move a single database to Knoppix, first log in to your original database server, and export it with:

```
root@ttyp0[root]# mysqldump database > database.txt
```

Then copy over the resulting database file using *scp*, FTP, or whichever file transfer protocol you prefer. Once the database is copied, run *mysql* and create a corresponding database on Knoppix:

```
mysql > CREATE DATABASE database;
```

You can then import your data with:

```
knoppix@ttyp0[knoppix]$ sudo mysql < database.txt
```

To copy all of the databases from one server to Knoppix, the procedure is similar but requires an extra step. First, back up your */usr/lib/mysql* directory, and create an empty one:

```
knoppix@ttyp0[knoppix]$ sudo mv /usr/lib/mysql /usr/lib/mysql.bak
knoppix@ttyp0[knoppix]$ sudo mkdir /usr/lib/mysql
```

Then export your complete database from the remote machine:

```
knoppix@ttyp0[knoppix]$ mysqldump --all-databases > all_databases.txt
```

Finally, copy *all_databases.txt* to Knoppix, and import it:

```
knoppix@ttyp0[knoppix]$ sudo mysql < all_databases.txt
```

## Inetd

Knoppix includes *inetd*, the Unix daemon that listens for incoming requests; when a request comes in, *inetd* starts the appropriate server daemon; *inetd* is disabled by default. Before you start *inetd*, check */etc/inetd.conf* and make sure that you don't mind if all the uncommented services are started. Even if you aren't sure, by default Knoppix allows only local connections to any of these services, so you are safe leaving them uncommented. This example shows you how to get FTP running with *inetd*.

Start *inetd* by typing the following command:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/inetd start
```

Now *inetd* listens on all of the ports configured in *inetd.conf* for connections. Once a connection is made, *inetd* starts the corresponding service.

At this point, if you attempt to connect to FTP on this server from another machine on the network, the connection is refused. One reason your attempt fails is because Knoppix disables anonymous FTP by default. A second reason might be because you haven't yet created a password for your *knoppix* user (with **`passwd knoppix`** in a terminal window). Most importantly, however, is that Knoppix uses *etc/hosts.deny* to disallow any remote connection to *inetd* services. You must edit */etc/hosts.allow* to allow remote connections.

Like most configuration files in */etc* under Knoppix, */etc/hosts.allow* is a symlink to a read-only file on the CD, so to edit it, you must move it to a backup file and then copy it back. In your */etc/hosts.allow* file, you see something like the following:

```
ALL : 127.0.0.1 LOCAL : ALLOW
ALL : ALL@ALL : DENY
```

The first field designates which service the rule is going to apply to. In both of these cases, the rule applies to all services. The second field is the list of hosts this rule applies to, in either IP address or hostname form. The third field determines whether this rule allows access or denies access. For example, if you want to allow your local subnet access to your FTP server, add a line reading:

```
in.ftpd : 192.168.0.* : ALLOW
```

Notice the use of the wildcard *. This tells *hosts.allow* to apply this rule to any host with an IP between `192.168.0.1` and `192.168.0.255`. Any changes to this file affect any new connections, so you don't need to restart *inetd*.

## NFS

Samba isn't the only filesharing method Knoppix supports. NFS (Network File System), the most commonly used Unix filesharing protocol, is also available. To configure NFS, you must first establish which directories you wish to share. If for instance, you wish to share a mounted filesystem, such as */mnt/hda1*, you must make sure that the filesystem is mounted *before* NFS is started. Also keep in mind that you are unable to unmount this filesystem as long as NFS is running. The */etc/exports* configuration file determines which directories are shared by NFS. Edit */etc/exports* as root, and add the directories you need to share. The syntax for this file is:

```
/share/path remote_host(options)
```

*remote_host* can be a particular hostname, IP, or an IP with wildcards, so if you want to share */mnt/hda1* with the entire `192.168.0.*` subnet, add the following line to */etc/exports*:

```
/mnt/hda1 192.168.0.*(rw)
```

To mount an NFS share remotely, you must also allow the remote connections to *portmap* and *mountd* in */etc/hosts.allow*. (As discussed previously, Knoppix also uses */etc/hosts.allow* to allow remote connections to *inetd* services.) If you haven't already done so, back up */etc/hosts.allow* and copy a version back, and add the following two lines to enable NFS access for your local subnet:

```
portmap: 192.168.0.* : ALLOW
mountd: 192.168.0.* : ALLOW
```

Now that all of the configuration files are prepared, make sure that any filesystems that must be mounted are mounted, and start the *portmap* and NFS services:

```
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/portmap start
knoppix@ttyp0[knoppix]$ sudo /etc/init.d/nfs-kernel-server start
```

If you want to monitor NFS-mount attempts, be sure to start the *syslog* daemon and read any error messages in case a connection request is refused.