Once upon a time on a Thinkpad A31p

wolfgang.loeffler@unibas.ch

This is the true story of a valiant hero, who sallied forth to do battle with *GNU/Debian Linux* on an *IBM Thinkpad A31p*. Please note that this is really nothing but an amusing story for your entertainment. Nothing else. Particularly, it is not to be mistaken for a manual or somesuch giving you instructions on how to install linux on your laptop. To put it in different words: Don't even think about blaming me for your very own actions. Think before you type.

Contents

1

Once upon a time on a Thinkpad A31p

This is the true story of a valiant hero, who sallied forth to do battle with *GNU/Debian Linux* on an *IBM Thinkpad A31p*. Please note that this is really nothing but an amusing story for your entertainment. Nothing else. Particularly, it is not to be mistaken for a manual or somesuch giving you instructions on how to install linux on your laptop. To put it in different words: Don't even think about blaming me for your very own actions. Think before you type.

Last Update: 1 January 2004 wolfgang.loeffler@unibas.ch

1.1. Hardware Information

Model Number:
2653R6G

Part Number:
TV2R6UK

BIOS Version:
1.01a (1NET07WW)

Processor:
Mobile Intel Pentium 4 with 2GHz

RAM:
2×256MB PC2100 DDR SDRAM with 400MHz (in 2 SODIMM Sockets)

Display:
15″ UXGA IPS TFT (FlexView) with 1600×1200 pixels

Video Chipset:
ATI Mobility FireGL 7800 with 64MB DDR RAM and 4xAGP bus — ATI Radeon Mobility M6 compatible

Audio Chipset:
Intel 82801CAM (ICH3-M) with AC'97 codec — i801 compatible

IDE Controller:
Intel 82801CAM (ICH3-M) — PIIXn compatible

Harddisk:
IBM Travelstar 60GH ATA-5 with 60.01GB capacity

CD-RW/DVD-ROM:
Matsushita UJDA 730 ATAPI DVD/CD-RW

USB Controller:
3 × Intel 82801CAM (ICH3-M) — UHCI compatible

IEEE1394 Firewire Controller:
unknown device 1180:0552 (Ricoh Co Ltd) (rev 0) — OHCI-1394 compatible

CardBus Controller:
Ricoh RL5c476 II (rev 168)

Modem:
Intel 82801CAM with AC'97 codec — Linmodem incompatible

Ethernet Controller:
82801CAM (ICH-3) (rev66) — EtherExpressPro/100 compatible

802.11b Wireless Controller:
Actiontec 802MIP — Intersil (Harris Semiconductor) PRISM2 compatible

Bluetooth Controller:
via Intel 82801CAM (ICH3-M) USB controllers

Modem:
3Com 3CCM156 Modem PC-Card

Ethernet:
3Com 3CCE589EC Ethernet PC-Card

WLAN:
US Robotics 22Mbps Wireless Cable/DSL Router USR-8022 (dead after 54 weeks)
US Robotics Wireless Turbo Access Point & Router USR-8054
US Robotics Wireless Access 11Mbps PC Card USR-2410

SCSI:
Adaptec APA1480A SCSI CardBus-Card; Nikon Filmscanner LS-2000

IEEE 1394 (FireWire):
LaCie d2 120GB Hard Disk

USB:
a pack of mice

1.2. Unsolved Problems

- When the machine blanks the screen using APM, any sound currently playing on the sound chip suddenly becomes choppy. All subsequently played sounds seem to play fine, though. Disabling Suspend while on AC power and fiddling with other BIOS setups may help, but we did not yet solve this one.
- After the screen has been blanked, we have to cycle trough all possible <Fn-F7> settings to switch it on again.

2

Preparation Work

2. 1. Removing the Stickers

Once upon a time computers were delivered unadorned. Nowadays you will find any number of stupid stickers glued to the machine in prominent places, informing you about what kind of operating system the machine was originally designed for and what brand of processor it sports. Until a few years ago, these stickers were only mildly adhesive and could be removed without too much hassle.

Not so with our new Thinkpad. After peeling off the stickers, we are left with two sticky patches of super-adhesive glue. To remove these, we put a few drops of paraffin oil (a.k.a. petroleum) on the glue, wait a little bit and then brush away the resulting jelly with a paper tissue. The remaing oil traces can be removed with a mildly alcoholic window cleaner, which in its turn can be removed with a damp paper tissue.

2. 2. Backup of the Thinkpad Service Partion

We want to install a pure Linux box without any Windows partition. So we would like to have some backup archive, from which we could later on re-recreate the original Windows setup — just in case. IBM in their infinite wisdom decided, however, not to provide any CD-ROMs with this Thinkpad but install an invisible "service partition", from which the Windows system can be easily restored — and Linux just as easily destroyed — just by pressing <F11> at boot time. There are, however, no provisions whatsoever for the unlikely case of a complete hard-disk failure. To remedy this situation, we would like to make an archive copy of this service partition.

To this end, we boot from the Debian installation CD-ROM, mount this invisible service partition and copy its contents into two suitably sized directories on the main partition, from where we can burn the contents on CD-R.

After having booted from the Debian installation CD-ROM and having selected our favourite language as well as keyboard layout, we launch a text console by pressing <ALT-F2> and then <Return>. First, we use cfdisk to save the original layout of the partitions. We probably need this information, if we would later like to recreate the service partition.

Original partition layout:

```
          ---Starting---      ----Ending----     Start Number of
 # Flags Head Sect Cyl   ID  Head Sect Cyl     Sector  Sectors
-- ----- ---- ---- ---- ---- ---- ---- ---- --------- ---------
 1  0x80    1    1    0 0x0C  239   63 1023        63 114473457
 2  0x00  239   63 1023 0x1C  239   63 1023 114473520   2736720
 3  0x00    0    0    0 0x00    0    0    0         0         0
 4  0x00    0    0    0 0x00    0    0    0         0         0
```

Now we create the two mount points /mnt/windows and /mnt/service and then mount /dev/hda1 (the windows partition) to /mnt/windows and /dev/hda2 (the service partition) to /mnt/service. After this, we create two archive directories on /mnt/windows and finally use tar to copy the contents of /mnt/service into these two directories.

Transfering the contents of the service partition:

```
#> mkdir /mnt/windows /mnt/service
#> mount /dev/hda1 -t msdos /mnt/windows
#> mount /dev/hda2 -t msdos /mnt/service
#> mkdir /mnt/windows/arch-1 /mnt/windows/arch-2
#> cd /mnt/service
#> tar cf - recovery | (cd /mnt/windows/arch-1; tar xvf -)
#> tar cf - pcdr | (cd /mnt/windows/arch-2; tar xvf -)
#> tar cf - ibmwork | (cd /mnt/windows/arch-2; tar xvf -)
#> tar cf - mfg | (cd /mnt/windows/arch-2; tar xvf -)
#> tar cf - *.exe *.bat *.bin *.prv *.txt *.com *.sys *.hlp *.cpi *.arf *.ini \
> *.def *.tag scri | (cd /mnt/windows/arch-2; tar xvf -)
#> cd /
#> umount /mnt/windows
#> umount /mnt/service
```

After rebooting Windows, we actually have to move about half the files from the arch-1/recovery directory to arch-2/recovery to make things fit onto two 700MB CD-Rs.

To recover the original Windows installation, we would have to clear the linux boot loader from the MBR, recreate the partitions from the saved partition table, copy the contents of the two CD-Rs into the recreated service partition, light two incense sticks, clap twice, bow deeply, pray, reboot, press <F11> — and wait.

2.3. Tools for the Creation of the Hibernation File

For hibernation to work under Linux we have to create a special hibernation partition. The necessary tools are distributed as an Incredibly Braindead Mess, i.e. a self-extracting archive named stndalhd.exe, which (a) does not run under Windows2000, (b) does only extract to a diskette and (c) is protected by a license, which does not allow the redistribution of its contents in a less idiotic way.

This horrid thing can be *downloaded* directly from IBM's FTP-Server — or via the *IBM website* by going to [Support & Download | Thinkpad | Family: Thinkpad A31p | Machine Type: 2653 | Model: R6G], then selecting [Downloadable Files | Downloadable Files by Category: Power Management] and finally choosing [TP General — Hibernation utility diskette II for standalone boot].

To extract this archive, we actually need access to a DOS or Windows95 machine with at least one 3.5″ floppy disk drive, e.g. A:, and one additional hard or floppy disk drive, e.g. X:. Running stndalhd.exe from drive X: installs three files on the floppy disk A:, namely phdisk.exe, phdos.sys and save2dsk.xga. These are the tools we actually need later. We put them on some machine, from which we can fetch them via scp later, preferably in a special directory named ibm.

We will have to execute phdisk.exe after we have installed Linux. Therefore we also need some “microsoftish” operating system, which can boot our Thinkpad from the CR-ROM drive. For this purpose, we *download* the bootable *Ripcord* ISO-image from *FreeDOS* and put it on a CD-R.

3

Installation of the Debian Base System

3.1. Partitioning the Hard Disk

One of the biggest questions while installing Linux is: How should we partion our hard disk? One one hand, the more separate (read-only) partitions we have, the better we can protect the system. On the other hand, more separate partitions mean that we have to estimate our needs beforehand, i.e. now. Making use of *Logical Volume Management* simplifies matters greatly, since LVM allows us to resize partitions later.

Based on years of experience, wild guesses, trial and error we decide on the following approximate sizes for the primary partitions.

Planned partition layout:

Partition:	Purpose:	Intended:	Actual:	Type:
/dev/hda1	/boot	8MB	15.49MB	83 Linux
/dev/hda2	swap	512MB	541.91MB	82 Linux Swap
/dev/hda3	hibernation	590MB	619.32MB	A0 IBM Thinkpad Hibernation
/dev/hda4	extended	rest		05 Extended
/dev/hda5	/	64MB	69.68MB	83 Linux
/dev/hda6	LVM	rest	58765.28MB	8e Linux LVM

Planned volume layout:

Volume:	Purpose:	Size:
/dev/disk/usr	/usr	3072MB
/dev/disk/tmp	/tmp	128MB
/dev/disk/var	/var	1024MB
/dev/disk/cdr	/cdr	1400MB
/dev/disk/home	/home	10240MB

Note: The hibernation partition must be a primary partition. Furthermore, making the root partition part of LVM can be dangerous. Also note: Although we ask fdisk for the *intended* sizes, the *actual* sizes are slightly larger

due to geometric constraints of the drive. Lastly note: It may be a good idea to reserve a large enough partition for the preparation of CD-R images.

To create the hibernation partition, we need to run phdisk.exe from a DOS partition. To this end, we boot into FreeDOS and use fdisk to create the two DOS partitions c: (/dev/hda1) and d: (/dev/hda2) and then format c: and format d:

First preliminary partition layout:

```
Partition:      Purpose:      Intended:      Actual:    Type:
/dev/hda1       DOS                8MB      15.49MB    06  FAT12
/dev/hda2       DOS              512MB     541.91MB    06  FAT16
```

Now we boot the Debian installation system with the *bf24* kernel by pressing <F3> at the boot prompt and then follow instuctions to add a preliminary linux partition and install a base system.

Second preliminary partition layout:

```
Partition:      Purpose:      Intended:      Actual:    Type:
/dev/hda1       DOS                8MB      15.49MB    06  FAT12
/dev/hda2       DOS              512MB     541.91MB    06  FAT16
/dev/hda3       preliminary      any           any      83  Linux
```

Then we boot into this preliminary system and go through the absolute minimum of configuration and make sure that the scp package is installed. Unfortunately, this leaves us without network configuration. We have to do this by hand.

First we have to edit the file /etc/network/interfaces to configure the ethernet device eth0. We have the static IP address www.xxx.yyy.zzz and our gateway is www.xxx.yyy.ggg.

/etc/network/interfaces:

```
auto lo eth0
iface lo inet loopback
iface eth0 inet static
  address    www.xxx.yyy.zzz
  netmask    255.255.255.0
  broadcast  www.xxx.yyy.255
  network    www.xxx.yyy.0
  gateway    www.xxx.yyy.ggg
```

Now we can start the network by loading the kernel module for the ethernet card and starting the interface.

Manual start of the network:

```
#> modprobe eepro100
#> ifup eth0
```

Then we mount the first DOS partition c: (/dev/hda1) and download the archive of the hibernation tools from the machine, where we have stored them.

Downloading the prepared hibernation tools:

```
#> mount /dev/hda1 /mnt
#> cd /mnt
#> scp "user@ip-nr-of-machine:ibm/*" .
```

Note: We actually have to use the IP number as we have not configured the resolver for machine names.

Now we reboot into FreeDOS and use fdisk to remove the preliminary linux partition /dev/hda3. Reboot into FreeDOS. At last we are ready to launch phdisk.exe and create the hibernation partition.

Having sucessfully created our hibernation partition, we reboot anew into the Debian installation system and continue partitioning the hard disk. The current partition layout is:

Third preliminary partition layout:

```
Partition:      Purpose:      Intended:      Actual:    Type:
/dev/hda1       DOS                8MB      15.49MB     06  FAT12
/dev/hda2       DOS              512MB     541.91MB     06  FAT16
```

```
/dev/hda4       hibernation       577MB     619.32MB     A0  IBM Thinkpad Hibernation
```

We use fdisk and repartition into:

Fourth preliminary partition layout:

```
Partition:      Purpose:      Intended:      Actual:    Type:
/dev/hda1       /boot              8MB      15.49MB     83  Linux
/dev/hda2       preliminary      512MB     541.91MB     83  Linux
/dev/hda3       hibernation      577MB     619.32MB     A0  IBM Thinkpad Hibernation
/dev/hda4       extended         rest                   05  Extended
/dev/hda5       /                 64MB      69.68MB     83  Linux
/dev/hda6       LVM              rest     58765.28MB     8e  Linux LVM
```

Note: To move the hibernation partition, we use the "extra functionality (experts only)" mode of fdisk and then "fix partition order" — preferably before we create any other partition.

We create the filesystems by hand, since we don't need no reserved blocks for /boot and the root filesystem /.

Manual creation of the filesystems:

```
#> mke2fs -m 0 /dev/hda1
#> mke2fs /dev/hda2
#> mke2fs -m 0 /dev/hda5
```

Now we have another problem. We should now define the LVM volume group and create the physical/logical volumes for /usr, /var, /tmp and /home. Unfortunately however, the Debian installation system does not support LVM. For this reason, we use the intended swap partition /dev/hda2 to install a preliminary Debian system without swap partition. As soon as possible, we will compile our custom kernel, setup LVM and then proceed rearranging things into their final layout.

3. 2. Installation of the Preliminary System

After having partitioned the hard disk and having formatted the partitions, we install a preliminary kernel as well as the base system and reboot, we make use of the good old tasksel and dselect install the following tasks and packages,

Debian tasks and packages for the preliminary system:

```
Task Development C
kernel-source-2.4.18
bzip2
linbz2-1.0
kernel-package
libncurses5-dev
lvm-common
lvm10
ext2resize
```

3. 3. Configuration of the Keyboard

We want put the <Control> in the Right Place™ where God has intended it to be, namely left to the <A> key. The old <Control> should take over as <Compose>. We don't need no <CapsLock>. To achieve this, we copy all the relevant kernel keymaps into /etc/console and modify them accordingly.

Copying the necessary files for the new keyboard layout:

```
#> cd /etc/console
#> cp /usr/share/keymaps/i386/qwerty/uk.kmap.gz .
#> cp /usr/share/keymaps/i386/include/qwerty-layout.inc.gz .
#> cp /usr/share/keymaps/i386/include/linux-keys-bare.inc.gz .
#> cp /usr/share/keymaps/i386/include/linux-with-alt-and-altgr.inc.gz .
```

```
#> cp /usr/share/keymaps/i386/include/euro.inc.gz .
#> mv uk.kmap.gz uk-mod.kmap.gz
#> gunzip uk-mod.kmap.gz
```

Then we modify uk-mod.kmap and set:

Modification of /etc/console/uk-mod.kmap:

```
keycode 29 = Compose
keycode 58 = Control
```

Finally we gzip things together again, update the symlink and load the new keymap:

Installing the new keyboard layout:

```
#> gzip uk-mod.kmap.gz
#> mv boottime.kmap.gz boottime.kmap.gz.ORI
#> ln -s uk-mod.kmap.gz boottime.kmap.gz
#> loadkeys boottime.kmap.gz
```

Note: We actually have to copy the included maps from /usr/share/keymaps/i386/include since the /usr partition is not yet mounted when the keymaps are loaded at boot time.

3.4. Configuration of LILO

We don't want those stupid symlinks /vmlinuz (and later /vmlinuz.old) cluttering our root directory. We remove them and install similar links in /boot. While updating /etc/lilo.conf we activate the compact option to speed up the loading of the kernel image.

Crating new symbolic links for the kernel images:

```
#> rm /vmlinuz*
#> cd /boot
#> ln -s vmlinuz-2.4.18-bf2.4 vmlinuz.old
```

Modifications in /etc/lilo.conf:

```
compact
image=/boot/vmlinuz
        label=linux
        read-only
        vga=normal
image=/boot/vmlinuz.old
        label=backup
        read-only
        vga=normal
        optional
```

Now we inform Debian about our changes and create a file describing the kernel image configuration in /etc/kernel-img.conf:

/etc/kernel-img.conf:

```
image_in_boot = yes
do_bootfloppy = no
```

3.5. Configuration of LVM

We then proceed to configure and build our own custom kernel using make menuconfig and make-kpkg. Almost any kernel configuration will do as long LVM is supported.

Kernel Config for LVM support:

```
—— Multi-Device Support (RAID and LVM) ——
         yes   Multiple devices driver support (RAID and LVM)
         yes   Logical volume manager (LVM) support
```

Compiling and installing the new kernel:

```
#> cd /usr/src
#> tar xjf kernel-source-2.4.18.tar.bz2
#> cd kernel-source-2.4.18
#> make menuconfig
#> make-kpkg clean
#> make-kpkg -rev Custom.1 kernel_image
#> cd ..
#> dpkg -i kernel-image_2.4.18
```

When the new kernel has been installed and lilo been updated sucessfully, we reboot.

Now we are ready to create the physical volume on /dev/hda6, add it to our volume group /dev/disk, create our logical volumes /dev/disk/usr, /dev/disk/tmp, /dev/disk/var and /dev/disk/home and the filesystems in them.

Creating the physical and logical volumes and filesystems:

```
#> pvcreate /dev/hda6
#> vgcreate disk /dev/hda6
#>
#> lvcreate -L 3072M -n usr disk
#> lvcreate -L 128M -n tmp disk
#> lvcreate -L 1024M -n var disk
#> lvcreate -L 1536M -n cdr disk
#> lvcreate -L 10G -n home disk
#>
#> mke2fs -j -m 0 /dev/disk/usr
#> mke2fs /dev/disk/tmp
#> mke2fs -j /dev/disk/var
#> mke2fs -j -m 0 /dev/disk/cdr
#> mke2fs -j /dev/disk/home
```

Now we mount these new partitions consecutively to /mnt and transfer the contents of the respective directories currently residing on /dev/hda2.

Relocating the directory structure to the new filesystems:

```
#> mount /dev/disk/var /mnt
#> cd /var; tar cf - . | (cd /mnt; tar xf -)
#> umount /mnt
#>
#> mount /dev/disk/usr /mnt
#> cd /usr; tar cf - . | (cd /mnt; tar xf -)
#> umount /mnt
#>
#> mount /dev/hda5 /mnt
#> mkdir /mnt/dev /mnt/cdrom /mnt/cdr /mnt/usr /mnt/var /mnt/tmp /mnt/home /mnt/proc
#> mkdir -p /mnt/mnt/cdrom
#> chmod 1777 /mnt/tmp /mnt/cdr
#>
#> cd /; tar cf - bin boot dev etc lib opt root sbin | (cd /mnt; tar xf -)
```

Note: Creating the new device directory /dev manually before any other directory might save the day, when some idiot goes for rm -rf * in the root directory. As rm descends the directory tree not alphabetically but in order of creation, /dev will be first to go. But if /dev/hda is gone, rm can't access the disk any more.

After this step, we update the filesystem layout in /mnt/etc/fstab.

/etc/fstab:

```
/dev/hda1      /boot       ext2     defaults,ro                          0 2
/dev/hda5      /           ext2     defaults,errors=remount-ro           0 1
```

```
/dev/disk/usr  /usr        ext3     defaults,ro                          0 2
/dev/disk/var  /var        ext3     defaults,errors=remount-ro           0 2
/dev/disk/tmp  /tmp        ext2     defaults,errors=remount-ro           0 2
/dev/disk/cdr  /cdr        ext3     defaults,errors=remount-ro           0 2
/dev/disk/home /home       ext3     defaults,errors=remount-ro           0 2
/dev/hdc       /mnt/cdrom  iso9660  user,ro,nosuid,nodev,exec,noauto     0 0
/dev/hdc       /cdrom      iso9660  defaults                             0 0
proc           /proc       proc     defaults                             0 0
none           /dev/pts    devpts   gid=5,mode=620                       0 0
```

Finally we need to update the root device in /mnt/etc/lilo.conf to /dev/hda5, set the new symlinks for the kernel images and rerun lilo.

Update of the root device in /etc/lilo.conf:

```
# /etc/lilo.conf
root=/dev/hda5
```

Installing the new LILO boot loader:

```
#> cd /mnt
#> vi etc/lilo.conf
#> cd boot
#> ln -sf vmlinuz-2.4.18 vmlinuz
#> ln -sf vmlinuz-2.4.18-bf2.4 vmlinuz.old
#> chroot /mnt lilo
```

After rebooting to switch to the new hard-disk layout we can finally use fdisk to remove our preliminary linux partition /dev/hda2 and replace it with a swap partition. And again we have to reboot to switch to the new hard-disk layout. At last we can activate the swap partition and add it to /etc/fstab.

/etc/fstab:

```
/dev/hda1      /boot       ext2     defaults,ro                          0 2
/dev/hda2      none        swap     sw                                   0 0
/dev/hda5      /           ext2     defaults,errors=remount-ro           0 1
/dev/disk/usr  /usr        ext3     defaults,ro                          0 2
/dev/disk/var  /var        ext3     defaults,errors=remount-ro           0 2
/dev/disk/tmp  /tmp        ext2     defaults,errors=remount-ro           0 2
/dev/disk/cdr  /cdr        ext3     defaults,errors=remount-ro           0 2
/dev/disk/home /home       ext3     defaults,errors=remount-ro           0 2
/dev/hdc       /mnt/cdrom  iso9660  user,ro,nosuid,nodev,exec,noauto     0 0
/dev/hdc       /cdrom      iso9660  defaults                             0 0
proc           /proc       proc     defaults                             0 0
none           /dev/pts    devpts   gid=5,mode=620                       0 0
```

Creating and activating the swap partition:

```
#> mkswap /dev/hda2
#> swapon -a
```

Note: We dont like to have our mount points in the root directory, we want them in /mnt. Unfortunately, Debian insists on /cdrom as the mount point for the installation. We will remove /cdrom and the respective entry in /etc/fstab when we reconfigure apt for network installs and updates.

4

Configuration of the Custom Kernel

4.1. Kernel Configuration Summary

Everytime we buy a new laptop, we are amazed about the lack of useful documentation about the actual hardware configuration and its capabilities. IBM's customer support website features an overwhelming amount of downloadable information — which turns out to be mostly irrelevant or dumbed down beyond any usefulness.

Our current kernel configuration is based on many guesstimates. It seems to work, though. But remember: This story is for your entertainment only. If you use this kernel configuration "as is" without reading and understanding the somewhat more detailed sections below, you will almost certainly damage your precious Thinkpad.

Kernel configuration:

```
── Code Maturity Level Options ──
        yes   Promt for development and/or incomplete code/drivers
── Loadable Module Support ──
        yes   Enable loadable module support
        yes   Set version information on all modules
        yes   Kernel modules loader
── Processor Type And Features ──
   Pentium-4   Processor family
        off   High Memory support
        yes   MTRR (Memory Type Range Regsiters) support
── General Setup ──
        yes   Networking support
        yes   PCI support
        any   PCI access mode
        yes   PCI device name database
        yes   Support for hot-pluggable devices
   ── PCMCIA/Cardbus support ──
            yes   PCMCIA/Cardbus support
```

```
              yes   CardBus support
         yes   System V IPC
         yes   Sysctl support
         ELF   Kernel core
         yes   Kernel support for ELF binaries
         mod   Kernel support for MISC binaries
         yes   Power Management support
         yes   Advanced Power Management BIOS support
         yes   Enable PM at boot time
         yes   Make CPU idle calls when idle
         yes   Enable console blanking using APM
         yes   RTC stores time in GMT
         yes   Allow interrupts during APM BIOS calls
-- Block Devices --
         mod   Loopback device support
         mod   Network block device support
-- Multi-Device Support (RAID and LVM) --
         yes   Multiple devices driver support (RAID and LVM)
         yes   Logical volume manager (LVM) support
-- Networking Options --
         yes   Packet socket
         yes   Packet socket: mmaped IO
         yes   Socket Filtering
         yes   Unix Domain Sockets
         yes   TCP/IP Networking
         yes   IP: TCP syncookie support (disabled by default)
-- ATA/IDE/MFM/RLL Support --
         yes   ATA/IDE/MFM/RLL support
   -- IDE, ATA and ATAPI Block devices --
              yes   Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
              yes   Include IDE/ATA-2 DISK support
              yes   Use multi-mode by default
              yes   SCSI emulation support
              yes   Generic PCI IDE chipset support
              yes   Sharing PCI IDE interrupts support
              yes   Generic PCI bus-master DMA support
              yes   Use PCI DMA by default when available
              yes   Intel PIIXn chipsets support
              yes   PIIXn tuning support
-- SCSI Support --
         yes   SCSI support
         mod   SCSI CD-ROM support
         yes   Enable vendori-specific extensions (for SCSI CDROM)
           2   Maximum number of CDROM devices that can be loaded as modules
         mod   SCSI generic support
         yes   Enable extra checks in new queuing code
   -- SCSI low-level drivers --
              mod   Adaptec AIC7xxx support
              253   Maximum number of TCQ commands per device
            15000   Initial bus reset in milli-seconds
-- Network Device Support --
         yes   Network device support
         mod   Dummy net driver support
   -- Ethernet (10 or 100Mbit) --
              yes   Ethernet (10 or 100Mbit)
              yes   EISA, VLB, PCI and on board controllers
              mod   EtherExpressPro/100 support
         mod   PPP (point-to-point protocol) support
         mod   PPP support for async serial ports
         mod   PPP Deflate compression
    -- PCMCIA network device support --
              yes   PCMCIA network device support
              mod   3Com 3c589 PCMCIA support
-- Input Core Support --
         yes   Input core support
         yes   Mouse support
        1600   Horizontal screen resolution
```

```
        1200   Vertical screen resolution
-- Character Devices --
        yes   Virtual terminal
        yes   Support for console on virtual terminal
        mod   Standard generic (8250/16550 and compatible UARTs) serial support
        yes   Unix98 PTY support
        256   Maximum number of Unix98 PTYs in use (0-2048)
   -- Mice --
             yes   Mouse support (not serial and bus mice)
             yes   PS/2 mouse (aka "auxiliary device" device) support

        yes   /dev/nvram support
        yes   Enhanced Realtime Clock support
        yes   /dev/agpgart (AGP Support)
        yes   Intel 440LX/BX/GX and i815/i830M/i840/i850 support
        yes   Direct Rendering Manager (XFree86 DRI support)
        mod   ATI Radeon
   -- PCMCIA character devices --
             mod   PCMCIA serial device support
-- File Systems --
        yes   Ext3 journalling file system support
        mod   DOS FAT fs support
        mod   MSDOS fs support
        mod   ISO 9660 CDROM file support
        yes   Microsoft Joliet CDROM extensions
        yes   Transparent decrompression extensions
        yes   /proc file system support
        yes   /dev/pts file system for Unix98 PYTs
        yes   Second extended fs support
        mod   UDF file system support (read only)
   -- Native Language Support --
        iso8859-1   Default NLS Option
             mod   Codepage 437 (United States, Canada)
             mod   Codepage 850 (Europe)
             mod   Japanese charsets (Shift-JIS, EUC-JP)
             mod   NLS ISO 8859-1 (Latin 1; Western European Languages)
             mod   NLS UTF-8
-- Console Drivers --
        yes   VGA text console
        yes   Video mode selection support
   -- Frame-buffer support --
             yes   Support for frame buffer devices
             yes   ATI Radeon display support

-- Sound --
        mod   Sound card support
        mod   Intel ICH (i8xx) audio support
-- USB Support --
        yes   Support for USB
        yes   UHCI Alternate Driver (JE) support
        yes   USB HIDBP Mouse (basic) support
-- Kernel Hacking --
        yes   Kernel debugging
        yes   Magic SysRq key
```

Note: As far as we understand the *Intel 82801 CAM I/O Controller Hub* has neither random generator nor watchdog functions.

4. 2. VESA Framebuffer

First of all, we want the VESA Framebuffer for our text consoles to make use of our nice screen resolution. In addition, we get a cuddly penguin boot logo at no extra cost. For this to work, we need to configure the kernel to support the VESA Framebuffer and then set the VGA mode in /etc/lilo.conf.

Kernel configuration for VESA framebuffer:

```
── Console Drivers ──
        yes   VGA text console
        yes   Video mode selection support
   ── Frame-buffer support ──
        yes   Support for frame buffer devices
        yes   VESA VGA graphics console
```

VESA Video Modes:

```
Mode:    LILO:      Graphics:      Text:
0x318    vga=792    1024x 768      128x48
0x31b    vga=795    1280x1024      160x64
0x374    vga=884    1600x1200      200x74
```

Video mode confguration in /etc/lilo.conf:

```
image=/boot/vmlinuz
        label=linux
        read-only
        vga=884
image=/boot/vmlinuz.old
        label=backup
        read-only
        vga=normal
        optional
```

4.3. Radeon Framebuffer

Our graphic chip is an *ATI Mobility FireGL 7800*, which seems to be quite similar to the *ATI Mobility M6* type of chips. The card reports itself in /proc/pci as device 1002:4c58 (ATI Technologies Inc) (rev0).

Unfortunately, the *ATI FireGL 7800* is not (yet) supported by the Linux Radeon Framebuffer. But with a little bit of hacking, we can make it so. In /usr/src/kernel-source-2.4.18/drivers/video/radeon.h we redefine the RADEON_LW type to our ID number and then reconfigure the kernel.

Kernel configuration for Radeon framebuffer support:

```
── Console Drivers ──
        yes   VGA text console
        yes   Video mode selection support
   ── Frame-buffer support ──
        yes   Support for frame buffer devices
        yes   ATI Radeon display support
```

Kernel hack in /usr/src/kernel-source-2.4.18/drivers/video/radeon.h:

```
#define PCI_DEVICE_ID_RADEON_LW         0x4c57     /* original */
#undef  PCI_DEVICE_ID_RADEON_LW
#define PCI_DEVICE_ID_RADEON_LW         0x4c58     /* dirty hack */
```

4.4. DRI (Direct Rendering Infrastructure)

Frankly we don't know whether the *fbdev* or the patched *radeon* XFree86 servers really make use of the Direct Rendering Infrastructure (DRI). Support for it does not hurt us, though.

Kernel Configuration:

```
── Character Devices ──
        yes   /dev/agpgart (AGP Support)
        yes   Intel 440LX/BX/GX and i815/i830M/i840/i850 support
        yes   Direct Rendering Manager (XFree86 DRI support)
```

```
        mod   ATI Radeon
```

At boot time, somebody or something tries to access DRI but does not how to do so. This produces the error message:

DRI module error message:

```
[date] [machine] modprobe: Can't locate module-char-major-226
```

To help out, we create the file /etc/modutils/dri and update /etc/modules.conf.

/etc/modules/dri:

```
alias  char-major-226  radeon  #  DRI
```

Updating /etc/modules.conf:

```
#> update-modules
```

4. 5. IEEE1394 Firewire (Disk)

According to /proc/pci, the the IEEE1394 Firewire Controller is a *1180:0552 (Ricoh Co Ltd) (rev 0)*, which is of the *OHCI-1394* type. To make use of our external LaCie Firewire Hard Disk, we have to configure for IEEE 1394, OHCI 1394, SBP-2 as well as SCSI disk support.

Kernel configuration for IEEE1394 FireWire:

```
── IEEE1394 (FireWire) Support ──
        mod   IEEE 1394 (FireWire) support
        mod   OHCI-1394 support
        mod   SBP-2 support (Harddisks etc.)
        mod   Raw IEEE 1394 I/O support
```

Kernel configuration for IEEE1394 FireWire disks:

```
── SCSI Support ──
         yes   SCSI support
         mod   SCSI disk support
           2   Maximum number of SCSI disks that can be loaded as modules
         yes   Enable extra checks in new queuing code
```

The actual configuration necessary to access the external hard disk is described in a different section.

4. 6. Hotplug (Modem, Ethernet, Scanner)

The CardBus (PCMCIA) controllers are *Ricoh RL5c476 II (rev 168)* devices, which are supported by the standard kernel. Since the standard kernel supports driver hotplugging, we don't have to install any special PCMCIA drivers any longer. All we have to do, is to install the packages providing the necessary infrastructure and then configure the kernel for our cards, i.e. the *3COM 3CCM156 Modem PC-Card*, *3COM 3CCE589EC Ethernet PC-Card*, and the *Adaptec APA1480A CardBus-Card*.

Debian packages for Hotplug (PCMCIA) support:

```
pcmcia-cs
hotplug
usbutils
```

Kernel configuration for hotplug support (Modem, Ethernet, Scanner):

```
── General Setup ──
        yes   Networking support
        yes   Support for hot-pluggable devices
   ── PCMCIA/Cardbus support ──
        yes   PCMCIA/Cardbus support
```

```
        yes   CardBus support
── Character Devices ──
        mod   Standard generic (8250/16550 and compatible UARTs) serial support
  ── PCMCIA character devices ──
        mod   PCMCIA serial device support
── Networking Options ──
        yes   TCP/IP Networking
── Network Device Support ──
        yes   Network device support
  ── PCMCIA network device support ──
        yes   PCMCIA network device support
        mod   3Com 3c589 PCMCIA support
── SCSI Support ──
        yes   SCSI support
        mod   SCSI generic support
        yes   Enable extra checks in new queuing code
  ── SCSI low-level drivers ──
        mod   Adaptec AIC7xxx support
        253   Maximum number of TCQ commands per device
      15000   Initial bus reset in milli-seconds
```

4.7. ALSA (Advanced Linux Sound Architecture)

The sound chip in this Thinkpad is an *Intel 82801CAM (ICH3-M)* using a *AC'97* front end. Although the standard kernel sound driver seems to work, we prefer to install the *ALSA* sound drivers (version 0.9). To do so, we need to install the relevant Debian packages and configure the kernel for general sound support.

Debian packages for ALSA (v0.9):

```
alsa-base             (v0.9)
alsa-source           (v0.9)
alsa-utils            (v0.9)
libassound2           (v0.9)
alsaconf
debhelper
debconf-util
html2text
```

Kernel configuration for ALSA:

```
── Sound
        mod   Sound card support
```

During the configuration of the alsa-source package, we select the intel8x0 and mpu401 drivers without ISAPnP from the menu. To compile the alsa-source package, we used again make-kpkg.

Compiling the ALSA kernel modules:

```
#> cd /usr/src
#> tar xzf alsa-driver.tar.gz
#> cd kernel-source-2.4.18
#> make-kpkg -rev Custom.1 modules_image
#> cd ..
#> dpkg -i alsa-modules-2.4.18_0.9+0beta12+3+p0+Custom.1_i386.deb
```

Finally, we must configure the alsa sound modules. To this end, we run alsaconf, select "0x31 Intel_i810/810E, i820,i840,MX440" as our card, choose some parameters and then update-modules.

Configuring ALSA:

```
#> alsaconf
#> update-modules
```

Unfortunately, there seems to be a bug in alsaconf, which makes it necessary to edit the resulting configuration file by hand, i.e. we have to replace the first occurence of "snd-card-intel8x0" with "snd-intel8x0".

/etc/alsa/modutils/0.9:

```
# ALSA
alias char-major-116      snd
alias snd-card-0          snd-intel8x0
# OSS/Free
alias char-major-14       soundcore
alias sound-slot-0        snd-card-0
# Soundcard
alias sound-service-0-0   snd-mixer-oss
alias sound-service-0-1   snd-seq-oss
alias sound-service-0-3   snd-pcm-oss
alias sound-service-0-8   snd-seq-oss
alias sound-service-0-12  snd-pcm-oss
options snd snd_major=116 snd_cards_limit=1 snd_device_mode=0666 snd_device_gid=29 snd_device_uid=0
options snd-card-intel8x0 snd_index=0 snd_id=CARD_0 snd_pbk_frame_size=128 snd_cap_frame_size=128 snd_m
```

Note: It is currently not advisable to install timidity together with the ALSA 0.9 version. timidity insists on installing the old version of libasound2. The following error message while trying to launch alsamixer is a strong hint that the old version of libasound2 got installed on top of the new one:

ALSA libasound2 error message:

```
#> alsamixer
ALSA lib control.c:601:(snd_ctl_open_noupdate) Invalid CTL default
alsamixer: function snd_ctl_open failed for default: No such file or directory
```

4.8. Winmodem

The internal modem is again based on the *Intel 82801CAM* chip in combination with the *AC'97*. This is one of these (literally) braindead "winmodems". According to the *Linmodem-HOWTO*, this specific spawn of engineering is not (yet) supported.

4.9. Ethernet

The ethernet adapter is an *Intel 82801CAM (ICH-3) (rev66)* controller, which is supported by the *EtherExpressPro/100* driver.

Kernel configuration for ethernet:

```
── General Setup ──
        yes   Networking support
── Networking Options ──
        yes   TCP/IP Networking
── Network Device Support ──
        yes   Network device support
   ── Ethernet (10 or 100Mbit) ──
        yes   Ethernet (10 or 100Mbit)
        yes   EISA, VLB, PCI and on board controllers
        mod   EtherExpressPro/100 support
```

For the semi-automagic configuration of the ethernet-device, we must have an active network connection when we boot the Debian installation system. If this is not the case during the installation of the etherconf package, dpkg-reconfigure etherconf is our friend.

Debian packages for ethernet configuration:

```
etherconf
libconfhelper-perl
liblogfile-rotate-perl
```

If we want that the modular ethernet driver is loaded automagically, we have to create the file /etc/modutils/eth0 and then update /etc/modules.conf:

/etc/modutils/eth0:

```
alias eth0 eepro100      # kernel ethernet driver for EtherExpressPro/100
```

Updating /etc/modules.conf:

```
#> update-modules
```

4.9.1. Intel PRO/100 Adapter Base Driver

Intel offers an *alternate driver* for the PRO/100 ethernet controllers. It was only a little bit hard to find in that corporate bloat of slowly, not completely or not at all loading graphics and scripts, since we did not know the filename of the tar-ball we were looking for. The subsequent compilation and installation of this driver is, however, completely painless. We only have to remember to register the new module afterwards.

Compiling and installing the Intel EtherExpress100/Pro driver:

```
#> cd /usr/src
#> tar -xzf e100-2.1.15.tar.gz
#> cd e100-2.1.15/src
#> make clean
#> make install
```

/etc/modutils/eth0:

```
# alias eth0 eepro100    # kernel ethernet driver for EtherExpressPro/100
alias eth0 e100          # intel ethernet driver for EtherExpressPro/100
```

Note: In newer kernel versions (2.4.20), this driver is now part of the kernel-source package and needs no longer be installed separately.

4.10. 802.11b Wireless Network

The 802.11b wireless network card shows up as an *1260:3873 (Harris Semiconductor) (rev1)*, but is in reality an *Actiontec 802MIP*, which is supported by the *linux-wlan Project*. The corresponding Debian packages are available in the "unstable" (a.k.a. "sid") distribution:

Debian packages for wlan-ng:

```
wireless-tools-25
linux-wlan-ng
linux-wlan-ng-modules-2.4.18
```

Due to a plethora of unmet dependencies it is not possible to just install these three binary packages on our "woody" based system. We have to compile the kernel modules as well as the support packages from source — as task which we defer to the next chapter.

However, since our wireless router wants us to connect via DHCP, we have to configure the kernel for Packet Socket and Socket Filtering:

Kernel configuration for DHCP:

```
── Networking Options ──
         yes   Packet socket
         yes   Packet socket: mmaped IO
         yes   Socket Filtering
```

4. 11. Bluetooth

Bluetooth is attached via the *Intel 82801CAM (ICH3-M)* USB controllers. This setup is supported by *BlueZ*, the *Official Linux Bluetooth Protocol Stack*.

As we don't have any bluetooth devices, we don't configure Bluetooth.

4. 12. IDE (Disk)

The IDE controller is an *Intel 82801CAM (ICH3-M)*. The hard disk is an *IBM Travelstar 60GH* ATA-5 with 60.01GB capacity. The controller seems to be similar to the *Intel PIIX4* chipset. Everything is nicely supported by the kernel:

Kernel configuration for IDE disks:

```
── ATA/IDE/MFM/RLL Support ──
         yes   ATA/IDE/MFM/RLL support
   ── IDE, ATA and ATAPI Block devices ──
         yes   Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
         yes   Include IDE/ATA-2 DISK support
         yes   Use multi-mode by default
         yes   Generic PCI IDE chipset support
         yes   Sharing PCI IDE interrupts support
         yes   Generic PCI bus-master DMA support
         yes   Use PCI DMA by default when available
         yes   Intel PIIXn chipsets support
         yes   PIIXn tuning support
```

4. 13. SCSI (DVD/CD-RW, Scanner)

The DVD/CD-RW combi drive is a *Matsushita UJDA 730* ATAPI DVD/CD-RW, which we want to use as an "SCSI-over-IDE" device. Additionally we have an *Adaptec APA-1480A* CardBus SCSI controller to connect our scanner.

Kernel configuration for SCSI (DVD/CD-RW, Scanner):

```
── ATA/IDE/MFM/RLL Support ──
         yes   ATA/IDE/MFM/RLL support
   ── IDE, ATA and ATAPI Block devices ──
         yes   Include IDE/ATA-2 DISK support
── SCSI Support ──
         yes   SCSI support
         mod   SCSI CD-ROM support
         yes   Enable vendori-specific extensions (for SCSI CDROM)
           2   Maximum number of CDROM devices that can be loaded as modules
         mod   SCSI generic support
         yes   Enable extra checks in new queuing code
   ── SCSI low-level drivers ──
         mod   Adaptec AIC7xxx support
         253   Maximum number of TCQ commands per device
       15000   Initial bus reset in milli-seconds
```

Note: If we configure for multiple LUN probing, we will end up with 7 CD-ROM devices as the DVD/CD-RW drive responds to all probed LUNs.

We have to inform lilo about our setup by editing /etc/lilo.conf:

Update of /etc/lilo.conf:

```
image=/boot/vmlinuz
        label=linux
        read-only
        vga=792
        append="hdc=ide-scsi"
```

The DVD/CD-RW drive shows up as Host: scsi0 Channel: 00 Id: 00 Lun: 00 and as the first SCSI CD-ROM, i.e. /dev/scd0. For convenience, we install symlinks for /dev/cdrom, /dev/dvd and /dev/cdrw.

Creating symlinks for the DVD/CD-ROM/CD-RW/:

```
#> rm /dev/cdrom /dev/dvd /dev/cdrw
#> ln -s /dev/scd0 /dev/cdrom
#> ln -s /dev/scd0 /dev/dvd
#> ln -s /dev/scd0 /dev/cdrw
```

Finally we also have to update /etc/fstab.

/etc/fstab:

```
/dev/hda1      /boot       ext2     defaults,ro                          0 2
/dev/hda2      none        swap     sw                                   0 0
/dev/hda5      /           ext2     defaults,errors=remount-ro           0 1
/dev/disk/usr  /usr        ext3     defaults,ro                          0 2
/dev/disk/var  /var        ext3     defaults,errors=remount-ro           0 2
/dev/disk/tmp  /tmp        ext2     defaults,errors=remount-ro           0 2
/dev/disk/cdr  /cdr        ext3     defaults,errors=remount-ro           0 2
/dev/disk/home /home       ext3     defaults,errors=remount-ro           0 2
/dev/scd0      /mnt/cdrom  iso9660  user,ro,nosuid,nodev,exec,noauto     0 0
/dev/scd0      /cdrom      iso9660  defaults,noauto                      0 0
proc           /proc       proc     defaults                             0 0
none           /dev/pts    devpts   gid=5,mode=620                       0 0
```

After inserting the *APA 1480A* card, the scanner shows up as Host: scsi1 Channel: 00 Id: 00 Lun: 00 and as the second generic SCSI device, i.e. /dev/sg1. Unfortunately, the AIC7xxx driver is apparently not hot-un-pluggable. In other words, we must shutdown the machine to remove the SCSI card. Otherwise the kernel will panic sooner or later — with unpredictable consequences for the rest of the SCSI subsystem.

4. 14. LVM (Logical Volume Management)

There is nothing new in our kernel configuration for Logical Volume Management (LVM). We can add, however, a few cosmetic configurations.

Kernel configuration for LVM:

```
—— Multi-Device Support (RAID and LVM) ——
         yes   Multiple devices driver support (RAID and LVM)
         yes   Logical volume manager (LVM) support
```

The LVM library scans for a bunch of devices while heading to find disks. If these are not supported by the kernel, the module loader tries to find and load the corresponding kernel modules. This produce somes error messages, which we can easily avoid.

LVM startup error messages:

```
[date] [machine] modprobe: Can't locate module block-major-8
[date] [machine] modprobe: Can't locate module block-major-33
[date] [machine] modprobe: Can't locate module block-major-34
```

block-major-8 are SCSI disk devices, block-major-33 and block-major-33 the third and forth IDE hard disk/CD-ROM interfaces. As we don't have any SCSI disks and don't access the CD-ROM as an IDE device, we can create /etc/modutils/lvm-exclude and then update /etc/modules.conf to make LVM shut up.

/etc/modutils/lvm:

```
alias  block-major-8   off        # SCSI disks
alias  block-major-33  off        # 3rd IDE controller
alias  block-major-34  off        # 4th IDE controller
```

Updating /etc/modules.conf:

```
#> update-modules
```

Note: We have to be careful about the block-major-8 entry and should probably remove it, if we want to access IEEE1394 FireWire hard disks.

4. 15. APM (Suspend & Hibernate)

Thanks to our acrobatics while partitioning the hard disk, everything works now out of the box. However, with such a large amount of RAM to write to or read from disk, shutting down the machine and rebooting is likely to be quite a bit faster than hibernating.

Kernel configuration for APM:

```
—— General Setup ——
         yes   Power Management support
         yes   Advanced Power Management BIOS support
         yes   Enable PM at boot time
         yes   Make CPU idle calls when idle
         yes   Enable console blanking using APM
         yes   RTC stores time in GMT
         yes   Allow interrupts during APM BIOS calls
```

Note: The last option “Allow interrupts during APM BIOS calls” is required for APM to work on out Thinkpad.

4. 16. Thinkpad Control Tools

There exists tpctl, a nifty package with a suite of small programs, which let us talk directly to some of the hardware in our Thinkpad.

Debian packages for Thinkpad control tools:

```
thinkpad-base
thinkpad-source
tpctl
```

No special configuration is necessary for the Thinpad kernel-modules. We only need to unpack the tar-ball, recompile the kernel, install it and then try to find out what all those nice options in (n)tpctl really are for.

Compiling and installing the Thinkpad kernel modules:

```
#> cd /usr/src
#> tar xzf thinkpad.tar.gz
#> cd kernel-source-2.4
#> make-kpkg -rev Custom.1 kernel_image
#> make-kpkp -rev Custom.1 modules_image
#> cd ..
#> rm -rf /lib/modules/2.4.18
#> dpkg -i kernel-image2.4.18_Custom.1_i386.deb
#> dpkg -i alsa-modules-2.4.18_0.9+beta12+3+p0+Custom.1_i386.deb
#> dpkg -i thinkpad-modules-2.4.18_3.5-1+Custom.1_i386.deb
```

4.17. Thinkpad Buttons

Another nice program is tpb (ThinkPad Buttons), which monitors volume, brightness and other settings by analysing the NVRAM. If we want to make use of this tool, we have to configure the kernel for NVRAM support.

Kernel Configuration for NVRAM support:

```
── Character Devices ──
        yes   /dev/nvram support
```

5

Debian and other Software Packages

5.1. Updating Debian

Keeping our Debian system up to date is very simple. We only need to configure apt for network updates. To this end, we install the apt-spy package and run apt-spy to find the fastest Debian mirrors. Then we edit /etc/apt/sources.list. Downloading and installing the latest Debian updates is then as simple as issuing the two commands apt-get update and apt-get upgrade.

Debian Packages for Finding Fast Debian Mirrors:

```
apt-spy
```

/etc/apt.sources:

```
# WOODY  main contrib non-free
# The following site was benchmarked at 14.49 kB/s
deb      ftp://ftp.skynet.be/debian/ woody main contrib non-free
# The following site was benchmarked at 14.29 kB/s
deb      ftp://ftp.stw-bonn.de/pub/mirror/debian/ woody main contrib non-free
# The following site was benchmarked at 14.29 kB/s
deb      ftp://toxo.com.uvigo.es/debian/ woody main contrib non-free
# Official Debian Mirror DE
deb      ftp://ftp.de.debian.org/debian woody main contrib non-free
# NON-US main contrib non-free
deb      http://non-us.debian.org/debian-non-US woody non-US/main non-US/contrib non-US/non-free
# SECURITY
deb      http://security.debian.org/ stable/updates main contrib non-free
```

Updating all Debian packages:

```
#> apt-get update
#> apt-get upgrade
```

5.2. Local Debian Tree

The Debian distribution is currently organised along three releses: “stable” (a.k.a. “woody”), “testing” (a.k.a. “sarge”) and “unstable” (a.k.a. “sid”). Under normal circumstances and for most purposes we want to install packages from “woody” only, which corresponds to the default setup.

As its label implies, “woody” is stable and does therewith not represent the bleeding edge of new development. In most cases, this does not matter. In a few cases, however, we need or want a more recent package with new features, which may be available in “sarge” or “sid” but not in “woody”. We put these few packages in our local Debian tree, from which we can install them without upgrading the whole distribution.

Another reason for a local tree is that we might not always agree with the compile-time configuration of a Debian package and want to recompile it from source. The resulting binary package will then be stowed in the local Debian tree and overrides the default package.

/etc/apt/sources.list:

```
# LOCAL
deb      file:/usr/local/debian woody main
deb      file:/usr/local/debian sarge main
# WOODY  main contrib non-free
# The following site was benchmarked at 14.49 kB/s
deb      ftp://ftp.skynet.be/debian/ woody main contrib non-free
# The following site was benchmarked at 14.29 kB/s
deb      ftp://ftp.stw-bonn.de/pub/mirror/debian/ woody main contrib non-free
# The following site was benchmarked at 14.29 kB/s
deb      ftp://toxo.com.uvigo.es/debian/ woody main contrib non-free
# Official Debian Mirror DE
deb      ftp://ftp.de.debian.org/debian woody main contrib non-free
# NON-US main contrib non-free
deb      http://non-us.debian.org/debian-non-US woody non-US/main non-US/contrib non-US/non-free
# SECURITY
deb      http://security.debian.org/ stable/updates main contrib non-free
```

Creating a local Debian tree:

```
#> cd /usr/local
#> mkdir -p debian/dists/woody/main/binary-i386/local
#> mkdir -p debian/dists/sarge/main/binary-i386/local
#> mkdir debian/dists/woody/source
#> mkdir debian/dists/woody/rpm
#> mkdir debian/dists/sarge/source
#> mkdir debian/dists/sarge/rpm
#> touch debian/override.local.woody
#> touch debian/override.local.sarge
#> touch debian/dists/woody/main/binary-i386/Packages
#> touch debian/dists/sarge/main/binary-i386/Packages
#> apt-get update
```

The binary-i386/local directories are, where we stow the binary packages. In the source directories we install, unpack and compile the source packages, in the rpm directories we install and convert RedHat RPM packages.

Note: We should not resdistribute binary packages we have compiled within this framework since they may have non-standard dependencies, which are hard or impossible to satisfy in other (more standard) setups.

5.3. GCC

Selecting a good version of the GNU C Compiler gcc is not unlike selecting a good wine. Most people don’t know and don’t care. But for those who know and care, there are big differences. The default version of gcc shipped with Debian, i.e. gcc 2.95.4, was a “vintage” version, whereas most later versions up and through gcc 3.0.x are more or less “fusel” versions. For this reason we want to replace gcc 3.0.4 from “woody” with gcc 3.2.1 from “sarge”.

The Debian *Packages* web-page is a good starting point to find all the packages we need (and their dependencies).

Debian Packages for gcc 3.2.1:

```
gcc-3.2           (1:3.2.1-0pre3)
gcc-3.2-base      (1:3.2.1-0pre3)
libgcc1           (1:3.2.1-0pre3)
cpp-3.2           (1:3.2.1-0pre3)
libc6             (2.2.5-14.3)
libc6-dev         (2.2.5-14.3)
locales           (2.2.5-14.3)
libdb1-compat     (2.1.3-7)
binutils          (2.13.90.0.10-1)
modutils          (2.4.19-3)
```

Downloading and Installing gcc 3.2.1 from "sarge":

```
#> cd /usr/local/debian/dists/sarge/main/binary-i386/local
#>
#> wget http://ftp.de.debian.org/debian/pool/main/g/gcc-3.2/gcc-3.2_3.2.1-0pre3_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/g/gcc-3.2/gcc-3.2-base_3.2.1-0pre3_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/g/gcc-3.2/libgcc1_3.2.1-0pre3_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/g/gcc-3.2/cpp-3.2_3.2.1-0pre3_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/g/glibc/libc6_2.2.5-14.3_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/g/glibc/libc6-dev_2.2.5-14.3_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/g/glibc/locales_2.2.5-14.3_all.deb
#> wget http://ftp.de.debian.org/debian/pool/main/d/db1-compat/libdb1-compat_2.1.3-7_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/b/binutils/binutils_2.13.90.0.10-1_i386.deb
#> wget http://ftp.de.debian.org/debian/pool/main/m/modutils/modutils_2.4.19-3_i386.deb
#>
#> cd /usr/local/debian
#> cat >> override.local.sarge
gcc-3.2         optional local
gcc-3.2-base    optional local
libgcc1         optional local
cpp-3.2         optional local
libc6           optional local
libc6-dev       optional local
locales         optional local
libdb1-compat   optional local
binutils        optional local
modutils        optional local
#> dpkg-scanpackages dists/sarge/main/binary-i386/local override.local.sarge > \
> dists/sarge/main/binary-i386/Packages
#>
#> apt-get update
#> apt-get upgrade
#> apt-get install gcc-3.2
```

5. 4. Debian Source Packages (Gendroolification ;-)

As most devoted zealots of the *Gentoo Linux Metadistribution* will fervently tell us, compiling our own linux distribution is not only etremely geeky but also speeds up things due to appropriate compiler optimisations.

Hmmmm . . . well . . . err . . . we have some doubts that we will ever notice the speed difference between the i386- and the i686-compiled flavours of ls. In our opinion, processor optimisation makes sense for extremely computation and graphics intensive tasks only. Furthermore, we will probably never be able to cash-in the time we spend for the compilation for extremely large software packages like KDE before the next version comes out and we have to start compiling again. Until Debian sees the light and provides a binary distribution precompiled for the i686-type processors, the installation of source packages is not really worth the hassle — it is possible, though.

First we have to configure the system for this kind of “gendroolification”. To do so, we install some additional apt packages and pentium-builder:

Debian Packages for Gendroolification:

```
apt-show-source
apt-show-versions
pentium-builder
```

We have written a small shell script to simplify the configuration of the gcc-wrapper builder-cc. We can evaluate this script from either tcsh- or bash-compatible shells:

Pentium builder activation script:

```
#!/bin/sh
# bin/build-i686
if [ "$SHELL = "/usr/bin/tcsh" ]; then
  echo "setenv DEBIAN_BUILDGCCVER 3.2;"
  echo "setenv DEBIAN_BUILDARCH i686;"
else
  echo "export DEBIAN_BUILDGCCVER=3.2"
  echo "export DEBIAN_BUILDARCH=i686"
fi
exit
```

Now we simply add for (almost) all deb entries in /etc/apt/sources.list a corresponding deb-src entry, so that we can download the source packages with apt.

/etc/apt/sources.list:

```
# LOCAL
deb      file:/usr/local/debian woody main
deb      file:/usr/local/debian sarge main
# WOODY  main contrib non-free
# The following site was benchmarked at 14.49 kB/s
deb      ftp://ftp.skynet.be/debian/ woody main contrib non-free
deb-src  ftp://ftp.skynet.be/debian/ woody main contrib non-free
# The following site was benchmarked at 14.29 kB/s
deb      ftp://ftp.stw-bonn.de/pub/mirror/debian/ woody main contrib non-free
deb-src  ftp://ftp.stw-bonn.de/pub/mirror/debian/ woody main contrib non-free
# The following site was benchmarked at 14.29 kB/s
deb      ftp://toxo.com.uvigo.es/debian/ woody main contrib non-free
deb-src  ftp://toxo.com.uvigo.es/debian/ woody main contrib non-free
# Official Debian Mirror DE
deb      ftp://ftp.de.debian.org/debian woody main contrib non-free
deb-src  ftp://ftp.de.debian.org/debian woody main contrib non-free
# NON-US main contrib non-free
deb      http://non-us.debian.org/debian-non-US woody non-US/main non-US/contrib non-US/non-free
deb-src  http://non-us.debian.org/debian-non-US woody non-US/main non-US/contrib non-US/non-free
# SECURITY
deb      http://security.debian.org/ stable/updates main contrib non-free
```

To download, compile and install a source package, e.g. gnupg (and the packages it depends on), we use apt-get, dpkg-buildpackage and dpkg.

Getting, compiling and installing Debian source packages:

```
#> cd /usr/local/debian/dists/woody/main/source
#>
#> eval `build-i686`
#>
#> apt-get source gnupg
#> apt-get build-dep gnupg
#>
#> apt-get source zlib1g
#> apt-get build-dep zlib1g
#> cd zlib-1.1.4
#> dpkg-buildpackage
#> cd ..
```

```
#> mv zlib1g*.deb ../binary-i386/local
#> echo "zlib1g optional local" >> /usr/local/debian/override.local
#> echo "zlib1g-dev optional local" >> /usr/local/debian/override.local
#> rm -rf zlib*
#>
#> apt-get source libgdbmg1
#> apt-get build-dep libgdbmg1
#> cd gdbm-1.7.3
#> dpkg-buildpackage
#> cd ..
#> mv libgdbmg1*.deb ../binary-i386/local
#> echo "libgdbmg1 optional local" >> /usr/local/debian/override.local
#> echo "libgdbmg1-dev optional local" >> /usr/local/debian/override.local
#> rm -rf gdbm*
#>
#> cd gnupg-1.0.6
#> dpkg-buildpackage
#> cd ..
#> mv gnupg*.deb ../binary-i386/local
#> echo "gnupg optional local" >> /usr/local/debian/override.local
#> rm -rf gnupg*
#>
#> cd /usr/local/debian
#> dpkg-scanpackages dists/woody/main/binary-i386/local override.local > \
> dists/woody/main/binary-i386/Packages
#>
#> apt-get update
```

Lather, wash, rinse, repeat.

5. 5. XFree86 4.0.0 Framebuffer Server

Thanks to our little framebuffer-hack, it is possible to run XFree86 via the framebuffer device — albeit without graphics acceleration. Debian manages to puzzle together a surprisingly well done configuration with every schnickschnack we need. The only thing we need to do is to enter the path for our truetype fonts (and apply some minor cosmetic corrections).

The small test program glxgears runs at roughly 290 fps (7 fps in fullscreen mode). The framebuffer does, however, not really draw every frame to the screen, so the wheels don't run smoothly but hang and jump.

Debian packages for the XFree86 4.0.0 framebuffer server:

```
xfree86-common
xserver-xfree86
```

5. 6. Xi Graphics Accelerated-X Server

Xi Graphics offers an *Accelerated-X Server* for the IBM Thinkpad A31p. The license for the LX version with 3D OpenGL X acceleration costs USD 149.— (other versions also available).

The XiG tar-ball contains two RedHat RPM archives, i.e. a kernel module package xsvc_3.0-39.i386.rpm and the driver package Summit_LX-Gold-2.2-6.i386.rpm. First we convert, extract, compile and install the kernel module.

Compilation and installation of the XiG kernel module:

```
#> alien xsvc_3.0-39.i386.rpm
#> dpkg -i xsvc_3.0-39_i386.deb
#> cd /usr/src/xig/xsvc/
#> make clean
#> make
```

```
#> make install
#> modprobe xsvc
```

modprobe barfs an informal message that loading this module will taint the kernel. This can be safely ignored. It just means that we should not bother the linux kernel developers, if and when things go wrong with the kernel after loading this module.

Modprobe warning about XiG kernel module:

```
Warning: loading /lib/modules/2.4.18/misc/xsvc.o will taint the kernel: no license
```

More important is the output of the module itself. If everything is okay, the following message should be written to the system log-files (default Debian location: /var/log/messages):

XiG kernel module initialisation messages:

```
[date] [some kernel]: xsvc: v3.0 (devrel@xig.com) [$XiGDate: 2002/10/24 20:31:44 $]
[date] [some kernel]: xsvc: Intel 845, 64MB at 0xe0000000 (1f000217/01)
```

Unfortunately, neither alien nor the Debian version of rpm can handle the driver package Summit_LX-Gold-2.2-6.i386.rpm. So we have to unpack and install it manually. Using the Debian package stow, we can even be semi-clever about it.

Debian package for stowing foreign software packages:

```
stow
```

Installing the XiG Accelerated-X server:

```
#> mkdir /tmp/TMP
#> mkdir -p /usr/local/packages/XiG/share/doc/XiG
#> rpm2cpio Summit_LX-Gold-2.2-6.i386.rpm | (cd /tmp/TMP; cpio --extract \
> --make-directories --no-absolute-filename --preserve-modification-time)
#> tar cf - /tmp/TMP/usr/. | (cd /usr/local/packages/XiG; tar xf -)
#> mv README.* *.pdf *.txt /usr/local/packages/XiG/share/doc/XiG
#> cd /usr/local/packages/XiG/X11R6/lib/X11/AcceleratedX/etc
#> ln -s Xlicense.demo Xlicense
#> cd ../../..
#> cp X11/AcceleratedX/OpenGL/usr/lib/libGL* .
#> cd /etc/X11
#> ln -sf /usr/X11R6/bin/Xaccel X
#>
#> dpkg-divert --rename --divert /usr/X11R6/lib/@@@libGL.so.1.2.distrib \
> /usr/X11R6/lib/libGL.so.1.2
#> dpkg-divert --rename --divert /usr/X11R6/lib/@@@libGLU.so.1.3.distrib \
> /usr/X11R6/lib/libGLU.so.1.3
#>
#> cd /usr/local/stow
#> ln -s ../packages/XiG .
#> stow -v --target=/usr XiG
#> ldconfig; ldconfig
#> rm -rf /tmp/TMP
```

Note: We actually have to run ldconfig twice. In the first run, only the links libGL.so.1.3 and libGLU.so.1.3 are created. In the second run libGL.so.1 and libGLU.so.1 are set.

Although the hardware specs claim a 4xAGP bus, the server locks hard if 4xAGP is configured in XSetup. Anyway, with this server, glxgears runs at impressive 1393 fps (66 fps in fullscreen mode). Contrary to what is written in the XiG documentation, we see no speed difference between Debian's and XiG's versions of the OpenGL libraries.

If we want to switch back to the XFree86 server, we just update the symlink in /etc/X11:

Switching back to the XFree86 server:

```
#> cd /etc/X11
#> ln -sf /usr/X11R6/bin/XFree86 X
```

Removing the XiG server is also simple and easy:

Removing the XiG Accelerated-X server:

```
#> cd /usr/local/stow
#> stow -D --target=/usr XiG
#> cd ../packages
#> rm -rf XiG
#> rm /etc/Xaccel.ini
#> dpkg-divert --remove /usr/X11R6/lib/libGL.so.1.2
#> dpkg-divert --remove /usr/X11R6/lib/libGLU.so.1.3
#> ldconfig
```

5. 7. XFree86 4.2.1 Radeon Server

Lo and behold! The XFree86 4.2.1 server in the Debian "sarge" distribution incorporates a patch, which adds support for the *ATI Mobility FireGL 7800* chip — albeit without AGP support. Under this server, glxgears runs here at 256 fps (13 fps in fullscreen mode) only. At least the wheels turn more or less smoothly. This server is, by the way, fast enough to watch DVDs.

Debian packages for the XFree86 4.2.1 Radeon server from "sarge":

```
xserver-xfree86     (4.2.1-3)
```

Downloading and installing XFree86 4.2.1:

```
#> cd /usr/local/debian/dists/sarge/main/binary-i386
#>
#> wget http://ftp.de.debian.org/debian/pool/main/x/xfree86/xserver-xfree86_4.2.1-3_i386.deb
#>
#> cd /usr/local/debian
#> echo "xserver-xfree  optional  local" >> override.local.sarge
#>
#> dpkg-scanpackages dists/sarge/main/binary-i386/local override.local.sarge > \
> dists/sarge/main/binary-i386/Packages
#> apt-get update
#> apt-get upgrade
```

Note: xserver-xfree86_4.2.1-3 depends on libc6_2.2.5-14.3, which we did already install while upgrading gcc.

/etc/X11/xdm/Xservers:

```
:0 local /usr/X11R6/bin/X :0 vt12 -layout Radeon -deferglyphs 16 -terminate
```

To configure the Radeon server, we just add one more server layout definition to /etc/X11/XF86Config-4:

Addition of a second server layout to /etc/X11/XF86Config-4:

```
Section "Monitor"
        Identifier      "TFT UXGA Monitor"
        DisplaySize      305 228
        Gamma            1.7
        Option           "DPMS"
EndSection
Section "Device"
        Identifier      "ATI Framebuffer"
        Driver           "fbdev"
        Option           "UseFBDev"              "true"
EndSection
Section "Device"
        Identifier      "ATI Radeon"
        Driver           "ati"
EndSection
Section "Screen"
        Identifier      "Framebuffer Screen"
        Device           "ATI Framebuffer"
```

```
        Monitor         "TFT UXGA Monitor"
        DefaultDepth    24
        SubSection "Display"
                Depth           24
                Modes           "1600x1200"
        EndSubSection
EndSection
Section "Screen"
        Identifier      "Radeon Screen"
        Device          "ATI Radeon"
        Monitor         "TFT UXGA Monitor"
        DefaultDepth    24
        SubSection "Display"
                Depth           24
                Modes           "1600x1200"
        EndSubSection
EndSection
Section "ServerLayout"
        Identifier      "Framebuffer"
        Screen          "Framebuffer Screen"
        InputDevice     "Internal Keyboard"
        InputDevice     "Trackpoint"
        InputDevice     "Mouse"
EndSection
Section "ServerLayout"
        Identifier      "Radeon"
        Screen          "Radeon Screen"
        InputDevice     "Internal Keyboard"
        InputDevice     "Trackpoint"
        InputDevice     "Mouse"
EndSection
```

Note: The Xsetup configuration tool from the XiG Accelerated Server suggests a gamma value of 2.432 (and a few other colour corrections) for our TFT monitor. Using some visual gamma correction tables and this nifty java-applet, we get on a somewhat lower gamma value of 2.2. Nevertheless, we settle for a value of 1.7 (or even less). Note also: The physical display size of 305×228 mm² results in a resolution of 133 dpi.

5.8. Laptop-Net

There exists a nifty Debian package called laptop-net, which allows the automagic detection of the IP address and subsequent reconfiguration of the system as a function of this IP address. Unfortunately, this package does not work as advertised. At least not here and now.

Update: Even the more recent versions of the laptop-net packagec continue to suck. Unfortunately, the package maintainer does not bother to answer emails, so we decided to roll our own variant and flush this crap here down the virtual toilet.

First of all, the package installation process overwrites important configuration files. Furthermore, Intel's ethernet driver e100 is not supported by one of the scripts in the package. So we better divert them, before we add our own modifications.

Diverting important laptop-net configuration files:

```
#> dpkg-divert --rename /etc/default/laptop-net
#> dpkg-divert --rename /etc/laptop-net/schemes
#> dpkg-divert --rename /usr/share/laptop-net/shared.sh
#> cp /etc/default/laptop-net.distrib /etc/default/laptop-net
#> cp /etc/laptop-net/schemes.distrib /etc/laptop-net/schemes
#> cp /usr/share/laptop-net/shared.sh.distrib /usr/share/laptop-net/shared.sh
```

Then we add support for the Intel e100 driver by editing /etc/default/laptop-net and /usr/share/laptop-net/shared.sh.

Addition of Intel e100 driver to /etc/default/laptop-net:

```
# MODULE_NAME="eepro100"
MODULE_NAME="e100"
```

Addition of Intel e100 driver to /usr/share/laptop-net/shared.sh:

```
        case "${MODULE_NAME}" in
#        3c59x | 8139too | au1000_eth | eepro100 | epic100 | fealnx | hamachi \
        3c59x | 8139too | au1000_eth | e100 | eepro100 | epic100 | fealnx | hamachi \
        case "${MODULE_NAME}" in
#        3c59x | eepro100 | epic100 | old_tulip | pcnet32 | rtl8139 | sis900 \
        3c59x | e100 | eepro100 | epic100 | old_tulip | pcnet32 | rtl8139 | sis900 \
```

Next, the automatic ARP detection of our IP address does not work in all cases. For this reason we have to puzzle together our own method of automagically determining our network.

We have hacked together a little script, which can do the automatic network detection by pinging the gateway for static interfaces and by analysing the responses of the DHCP client. Being based on a shell script, this method is rather slow. Having to wait for ping-timeouts, it is even slower. But it seems to work. It would still be a good idea to write a small C program for this job. To integrate it into the laptop-net package, we have to modify /usr/share/laptop-net/shared.sh.

Modification of /usr/share/laptop-net/shared.sh for ping-discovery:

```
# ARP_DISCOVERY=/usr/lib/laptop-net/arp-discovery
ARP_DISCOVERY=/usr/local/packages/laptop-net/ping-discovery
```

Shell script for ping-discovery:

```
#!/bin/sh
# /usr/local/packages/laptop-net/ping-discovery:
if [ -z "$1" ]; then
  echo "Usage: ping_discovery <interface > <ip-map>"
  exit 1
fi
INTERFACE="$1"
IPMAPFILE="$2"
SCHEME_DIR=/var/lib/laptop-net/schemes
export PATH=/bin:/usr/bin:/sbin:/usr/sbin
# Load available schemes from /etc/laptop-net/ip-map.
SCHEME=`grep -v -E "(^$|#)" ${IPMAPFILE} | awk '{print $1}'`
if [ -z "$SCHEME" ]; then
    echo "@NO-CHOICES@"
    exit 1
fi
# Allow for extremely long negotation times with stoopid switches
sleep 30
# Test interface configuration: if already configured, return scheme; if
# unknown configuration, bring the interface down; if not configured, continue
IFTEST=`ifconfig | grep -A 1 "${INTERFACE}" | tail -1 | sed -e "s/:/ /" | awk '{print $3}'`
if [ -n "${IFTEST}" ]; then
  for s in ${SCHEME}; do
    IFACE=${SCHEME_DIR}/$s
    ADDRESS=`grep "address" ${IFACE} | awk '{print $2}'`
    if [ "${IFTEST}" = "${ADDRESS}" ]; then
      echo $s
      exit 0
    fi
  done
  ifconfig ${INTERFACE} down
fi
#-------------------------------------------------------------------------
# Loop over all non-DHCP schemes, try to bring up the interface and ping the
# gateway
for s in ${SCHEME}; do
    IFACE=${SCHEME_DIR}/$s
    DHCP=`grep "dhcp" $IFACE`
```

```
    if [ -z "$DHCP" ]; then
      ADDRESS=`grep "address" ${IFACE} | awk '{print $2}'`
      NETMASK=`grep "netmask" ${IFACE} | awk '{print $2}'`
      BROADCAST=`grep "broadcast" ${IFACE} | awk '{print $2}'`
      GATEWAY=`grep "gateway" ${IFACE} | awk '{print $2}'`
      ifconfig ${INTERFACE} ${ADDRESS} netmask ${NETMASK} broadcast ${BROADCAST} up
      ping -r -c 1 ${GATEWAY} &> /dev/null
      SUCCESS=$?
      ifconfig "${INTERFACE}" down
      if [ "${SUCCESS}" = 0 ]; then
        echo $s
        exit
      fi
    fi
done
#------------------------------------------------------------------------------
# Try to bring up the interface using DHCP (somewhat ugly since the DHCLIENT
# does not exit on failures but brings up the interface anyway, blech . . .)
for s in ${SCHEME}; do
    IFACE=${SCHEME_DIR}/$s
    DHCP=`grep "dhcp" $IFACE`
    if [ -n "$DHCP" ]; then
      /sbin/dhclient-2.2.x -e ${INTERFACE} &gt; /tmp/DHCLIENT.log
      SUCCESS=`grep "DHCPACK" /tmp/DHCLIENT.log`
      kill `pidof dhclient-2.2.x`
      ifconfig "${INTERFACE}" down
      rm -f /tmp/DHCLIENT.log
      if [ -n "${SUCCESS}" ]; then
        echo $s
        exit
      fi
    fi
done
echo "@NO-RESPONSES@"
exit 0
```

Note: We redefine the layout of the IP-to-SCHEME map file /etc/laptop-net/ip-map. Each line in this file contains now two entries: first the name of the scheme, and second the expected IP number under this scheme. If the network interface is configured via DHCP, only the name of the scheme is collected. In this case, the second entry with the IP number is not necessary.

Note also: We only need one single scheme for DHCP even if we use this for more than one network. That's what the different profiles are for.

/etc/laptop-net/ip-map:

```
# schemes for different static IP addresses
scheme1  www.xxx.yyy.zzz    = our expected IP address under scheme1
scheme2  WWW.XXX.YYY.ZZZ    = our expected IP address under scheme2
# one single scheme for DHCP
scheme3                       no IP number necessary
```

We also want to transfer the "control" of all network related boot scripts from the boot process rcS to the LaptopNet daemon ifd — without confusing the Debian package management system, of course. To do so, we have to remove the symlinks from the /etc/rcN.d directories and install them in the /etc/laptop-net/profiles/XXXX/rc.d, where XXXX is one of home, work or local. Since the Debian boot-script manager update-rc.d will reinstall these links as long as it find the corresponding script in /etc/init.d, we have to divert these to a different location.

Diverting the boot script from init to ifd:

```
#!/bin/sh
mkdir /etc/laptop-net/init.d
PACKAGES="dns-clean fetchmail ippl lprng ntp ntpdate ppp scandetd ssh xinetd"
for p in $PACKAGES; do
  dpkg-divert --rename --divert /etc/laptop-net/init.d/$p /etc/init.d/$p
  update-rc.d $p remove
done
```

Now we go to the rc.d directories and create the new symlinks by hand.

Creating the new startup/shutdown symlinks (example):

```
#> cd /etc/laptop-net/profiles
#> mkdir -p work/rc.d home/rc.d xxx-default/rc.d
#> cd work/rc.d
#> ln -s ../../../init.d/ssh S20ssh
#> ln -s ../../../init.d/ssh K20ssh
```

Now we have a problem with ifd, the daemon which watches the status of our ethernet adapter. There seems to exist a race between the daemon itself and the termination of the link-change shell script, which is forked by this daemon. The offending lines are in the source file ifd.c, more precisely in the subroutine execute:

src/ifd.c:

```
static int
execute (const char * msg, const char * cmd)
{
  int ret;
  FILE * f;
  char line [256];
  const char * suffix = " 2>&1";
  unsigned int limit = ((sizeof (line)) - ((strlen (suffix)) + 1));

  syslog (LOG_INFO, "executing: '%s'", cmd);
  strncpy (line, cmd, limit);
  (line[limit]) = '\0';
  strcat (line, suffix);
  f = (popen (line, "r"));
  while (fgets (line, 255, f))
    {
      (line [(strlen (line)) - 1]) = '\0';
      syslog (LOG_INFO, "+ %s", line);
    }
  ret = (pclose (f));
  if (WIFEXITED (ret))
    {
      if (WEXITSTATUS (ret))
        syslog (LOG_INFO, "%s exited with status %d",
                msg, (WEXITSTATUS (ret)));
      return (WEXITSTATUS (ret));
    }
  else
    {
      syslog (LOG_INFO, "%s exited on signal %d",
              msg, (WTERMSIG (ret)));
      return (-1);
    }
}
```

The while loop ends as soon as fgets encounters an EOF condition. But since we are connected to stdout of the forked shell process, this means that the shell process is terminating at this point. By the time pclose is called it has terminated, so that pclose essentially waits on a no longer existing process, i.e. forever. The dead shell becomes a zombie process and ifd remains stalled until killed.

In order to fix this problem, we have hack the source code of ifd and replace the subroutine execute by:

src/ifd.c:

```
static int
execute (const char * msg, const char * cmd)
{
  syslog (LOG_INFO, "executing: '%s'", cmd);
  return system(cmd);
}
```

All in all, our installation of laptop-net is hideous and slow. But it works.

5.9. PCMCIA-CS

Although we don't need to compile any kernel modules from the pcmcia-cs packge, we want to compile the corresponding tools. The reason herefore is simple: The pcmcia tools distributed in the binary package are not "trusting", i.e. we can not change the pcmcia schemes as a normal user. To change this, we have to install the pcmcia-source package, unpack the tar-ball, reconfigure the kernel, configure the pcmcia package and finally compile and install the new package.

Debian package for PCMCIA sources:

```
pcmcia-sources
```

Building and installing the pcmcia-cs package:

```
#> cd /usr/src
#> tar xzf pcmcia-cs.tar.gz
#> cd kernel-source-2.4.19
#> make oldconfig; make dep
#> cd ../modules/pcmcia-cs
#> make config
#> cp config.* debian
#> debian/rules binary-cs
#> cd ..
#> mv pcmcia-cs_3.1.33-6_i386.deb /usr/local/debian/dists/woody/main/binary-i386/local
#> cd /usr/local/debian
#> echo "pcmcia-cs_3.1.33-6_i386.deb" >> override.local
#> dpkg-scanpackages dists/woody/main/binary-i386/local override.local > \
> dists/woody/main/binary-i386/Packages
#> apt-get update
#> apt-get install --reinstall pcmcia-cs
```

We also want to integrate our PCMCIA setup with the laptop-net infrastructure. We base our configuration on the presupposition that laptop-net is always up, running and watching the status of eth0. Furthermore, we assume that eth0 will always be associated with the internal ethernet adaptor, so that our ethernet PC-card will always show up as eth1. Lastly, we want that eth1 gets managed by laptop-net only in the case, when no network is attached to eth0.

To do so, we use a script /usr/local/packages/laptop-net/pcmcia, which we call from the PCMCIA network script /etc/pcmcia/network.opts.

/usr/local/packages/laptop-net/pcmcia:

```
#!/bin/sh
INTERFACE="eth0"
PROF_CHANGE=/usr/share/laptop-net/profile-change
IFTEST=`ifconfig | grep -A 1 "${INTERFACE}" | tail -1 | sed -e "s/:/ /" | awk '{print $3}'`
if [ -z "${IFTEST}" ]; then
  ${PROF_CHANGE} $*
fi
exit 0
```

Update of /etc/pcmcia/network.opts:

```
    # Extra stuff to do after setting up the interface
    start_fn() { /usr/local/packages/laptop-net/pcmcia $DEVICE; return; }
    # Extra stuff to do before shutting down the interface
    stop_fn() { /usr/local/packages/laptop-net/pcmcia $DEVICE down; return; }
```

5. 10. Sub-Pixel Font-Rendering

Sub-Pixel Font-Rendering is a Good Thing™. It's actually an old idea, whose time has finally come or will come really soon now on a LCD display near you. To triple the horizontal screen resolution for font rendering right now, we have to install XFree86 4.2.1 and then compile and install the new font configuration scheme fontconfig and update the X Render Extension together with libXft.

Debian Packages needed for compiling fontconfig:

```
libexpat1-dev
libfreetype6-dev
```

Compiling and installing fontconfig and Xft:

```
#> eval `build-i686`
#>
#> cd /usr/local/packages
#> mkdir -p fontconfig-2.0/local/src
#> cd fontconfig-2.0/local/src
#> wget http://fontconfig.org/release/fcpackage.2_0.tar.gz
#> tar xzf fcpackage.2_0.tar.gz
#>
#> cd fcpackage.2_0/fontconfig
#> ./configure --prefix=/usr/local/packages/fontconfig-2.0/X11R6
#> make
#> make install
#> cd /usr/local/stow
#> ln -s ../packages/fontconfig-2.0 .
#> stow -v --target=/usr fontconfig-2.0
#>
#> cd /usr/local/packages/fontconfig-2.0/local/src/fcpackage.2_0/Xrender
#> xmkmf -a
#> make
#> dpkg-divert --rename /usr/X11R6/lib/libXrender.a
#> dpkg-divert --rename /usr/include/X11/extensions/Xrender.h
#> dpkg-divert --rename /usr/include/X11/extensions/render.h
#> dpkg-divert --rename /usr/include/X11/extensions/renderproto.h
#> make install
#>
#> cd ../Xft1
#> xmkmf -a
#> make
#> make install
#>
#> cd ../Xft
#> ./configure --prefix=/usr/local/packages/fontconfig-2.0/X11R6
#> make
#> dpkg-divert --rename /usr/include/X11/Xft/Xft.h
#> make install
#> cd /usr/local/stow
#> stow -v -D --target=/usr fontconfig-2.0
#> cd /usr/local/packages/fontconfig-2.0
#> mkdir -p lib/pkgconfig
#> ln -s X11R6/lib/pkgconfig/* lib/pkgconfig
#> rm -rf local/src/fcpackage.2_0
#> cd /usr/local/stow
#> stow -v --target=/usr fontconfig-2.0
```

Xrender and Xft1 can not be stowed and must be installed directly into the X11R6 tree. This makes removing these files a bit cumbersome.

Removing fontconfig-2.0:

```
#> cd /usr/local/stow
#> stow -v -D --target=/usr fontconfig-2.0
#> rm fontconfig-2.0
#> rm -rf /usr/local/packages/fontconfig-2.0
```

```
#> dpkg-divert --remove /usr/include/X11/Xft/Xft.h
#>
#> rm /usr/X11R6/lib/libXrender.so
#> rm /usr/X11R6/lib/libXrender.so.1.1
#> rm /usr/X11R6/lib/libXrender.a
#> rm /usr/include/X11/extensions/Xrender.h
#> rm /usr/include/X11/extensions/extutil.h
#> rm /usr/include/X11/extensions/region.h
#> rm /usr/include/X11/extensions/render.h
#> rm /usr/include/X11/extensions/renderproto.h
#> dpkg-divert --remove /usr/X11R6/lib/libXrender.a
#> dpkg-divert --remove /usr/include/X11/extensions/Xrender.h
#> dpkg-divert --remove /usr/include/X11/extensions/render.h
#> dpkg-divert --remove /usr/include/X11/extensions/renderproto.h
#> ldconfig
#>
#> rm /usr/X11R6/lib/libXft.so
#> rm /usr/X11R6/lib/libXft.so.1.2
#> ldconfig
```

Finally, we have to marry the new XFree86 font configuration fontconfig with the Debian font manager defoma. To this end we modify /etc/fonts/fonts.conf.

Modification of /etc/fonts/font.conf:

```
<!-- Font directory list configured on Sat Nov 16 01:41:41 CET 2002 -->
        <dir>/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType</dir>
        <dir>/var/lib/defoma/x-ttcidfont-conf.d/dirs/CID</dir>
        <dir>/usr/X11R6/lib/X11/fonts</dir>
        <dir>~/.fonts</dir>
```

5.11. Xprint — the X11 print system

Setting up and configuring a printer under Linux can be adventuresome. Hooking up a postscript printer can become a nightmarish, especially if we want to print japanese postscript files on said printer. A few programs are clever enough to download japanese postscript fonts into the printer before they attempt to print. Most programs, however, assume that the printer has installed japanese postscript fonts — which is usually not the case outside of Japan.

A project which attempts to remedy this situation is XPrint. This is essantially a X11 server, which does send its output to the printer instead of the screen. We want to use the development trunk release 008, since it supports TrueType fonts.

First we download the source tar-ball, which we extract into a separate directory for compilation. Compiling and installing is well described by the installation manual and is quite simple.

Downloading and Compiling Xprint:

```
#> cd /usr/local/packages
#> wget http://puck.informatik.med.uni-giessen.de/download/xprint_mozdev_org_source-2002-12-02-trunk.ta
#> tar xzf xprint_mozdev_org_source-2002-12-02-trunk.tar.gz
#> cd xprint/src/xprint_main/xc/
#> make World 2>&1 | tee -a buildlog.log
#> cd packager/
#> make make_xprint_tarball
#> cd /usr/local/packages
#> rm -rf xprint
#> mkdir -p xprint/usr/local/src/xprint
#> mv /tmp/xprint_server_030118184734.tar.gz xprint/usr/local/src/xprint
#> mv xprint_mozdev_org_source-2002-12-02-trunk.tar.gz xprint/usr/local/src/xprint
#> cd xprint
#> tar xzf usr/local/packages/xxprint/xprint_server_030118184734.tar.gz
#> cd etc
#> mkdir -p laptop-net/init.d
```

```
#> mv init.d/xprint laptop-net/init.d
#> rmdir init.d
#> cd /usr/local/stow
#> ln -s ../packages/xprint .
#> stow -v --target=/ xprint
```

The number in the name of the tar-ball will depend of the exact building date. To use the Xprint server, we have to add two variables to the environment:

/etc/environment:

```
# xprint server
export XPSERVERLIST="`/etc/laptop-net/init.d/xprint get_xpserverlist`"
export XPRINTER="lp"
```

Finally, we set the necessary links in the laptop-net directories:

Creating the new startup/shutdown symlinks (example):

```
#> cd /etc/laptop-net/profiles
#> mkdir -p work/rc.d home/rc.d xxx-default/rc.d
#> cd work/rc.d
#> ln -s ../../../init.d/xprint S34xprint
#> ln -s ../../../init.d/xprint K04xprint
```

Integrating Xprint into laptop-net leaves us with the problem that we must ensure that the two environment variables are always and under all profiles well defined. To do so, we set up /dev/null as printing device for profiles without a real printer and start lprng and Xprint with this empty device. Furthermore, we wrap all Xprint aware programs (in our case Mozilla) in shell scripts, which set these two variables explicitly before starting the program.

Xprint constructs its font path not by consulting the XFree86 configuration file but by searching for fonts.dir files below the XPROJECTROOT directory. This is not quite what we want for two reasons: First, Xprint misses all fonts managed by defoma, second, it finds all fonts — even those, which we have explicitly exluded in our XFree86 configuration file. This behaviour can easily be corrected by editing the Xprint boot script.

Adapting /etc/laptop-net/init.d/xprint:

```
get_system_fontlist()
{
  case "$(uname -s)" in
  *SunOS*)
    ;;
  Linux)
    XF86CONFIG="/etc/X11/XF86Config-4"
    xfontpath=`grep "FontPath" $XF86CONFIG | egrep -v "^#" | sed 's/FontPath//' | tr "\"," " "`
    ( for xpath in $xfontpath; do
      echo $xpath | sed 's/:unscaled//'
      done ) | sort | uniq
    ;;
  *)
    ;;
esac
}
```

5. 12. Mozilla

One of the most important programs supporting sub-pixel font-rendering right now is The Beast, a.k.a. Mozilla. Unfortunately the precompiled binaries available at the Mozilla Homepage lack this feature, so we have to use the source and compile the thing ourselves. Fortunately, this has become quite easy.

First we use the Unix Build Configurator to configure The Beast. To compile for sub-pixel font-rendering, we have to enable Xft support. Depending on the actual configuration of Mozilla, we then have to install a few Debian packages.

Mozilla Configuration File ~/mozconfig:

```
# sh
# Build configuration script
#
# See http://www.mozilla.org/build/unix.html for build instructions.
#
# Options for 'configure' (same as command-line options).
ac_add_options --prefix=/usr/local/packages/mozilla-1.2b
ac_add_options --with-pthreads
ac_add_options --with-system-jpeg=/usr/lib
ac_add_options --with-system-zlib=/usr/lib
ac_add_options --with-system-png=/usr/lib
ac_add_options --with-system-mng=/usr/lib
ac_add_options --enable-default-toolkit=gtk
ac_add_options --enable-toolkit-gtk
ac_add_options --disable-mailnews
ac_add_options --enable-xft
ac_add_options --enable-xprint
ac_add_options --enable-crypto
ac_add_options --disable-accessibility
ac_add_options --disable-installer
ac_add_options --disable-debug
ac_add_options --disable-logging
ac_add_options --enable-strip
ac_add_options --enable-elf-dynstr-gc
```

Debian packages needed to compile Mozilla:

```
libjpeg62-dev
zlib1g-dev
libpng2-dev
libmng-dev
libgtk1.2-dev
libfreetype6-dev
pkg-config
libidl-dev
```

Furthermore, Mozilla wants a symlink (or alias) from libIDL-config-2 to libIDL-config. If all necessary packages are installed, compiling and installing Mozilla is quite simple.

Compiling Mozilla:

```
#> cd /usr/local/packages
#> wget ftp://sunsite.cnlab-switch.ch/mirror/mozilla/mozilla/releases/mozilla1.2b/src/mozilla-source-1.2
#> tar xzf mozilla-source-1.2b.tar.gz
#> cd mozilla
#> cp ~/mozconfig .mozconfig
#> make -f client.mk build
#> make install
#> cd ..
#> rm -rf mozilla
#> ln -s mozilla-1.2b mozilla
```

We use a small shell script to set the MOZILLA_HOME environment variable and launch the actual Mozilla startup script.

Mozilla Startup-Script /usr/local/bin/mozilla:

```
#!/bin/sh
export XPSERVERLIST="`/etc/laptop-net/init.d/xprint get_xpserverlist`"
export XPRINTER="lp"
MOZILLA_HOME="/usr/local/software/mozilla"
set -e
# Check for DISPLAY
if [ -z $DISPLAY ]
then
    echo "No DISPLAY is set."
    exit 1
```

```
fi
if [ -x $MOZILLA_HOME/bin/mozilla ]; then
  exec $MOZILLA_HOME/bin/mozilla "$@"
else
  echo "$MOZILLA_HOME/bin/mozilla not found."
  exit 1
fi
exit 1
```

5. 13. tpb (ThinkPad Buttons)

tpb is a clever program, which analyses the content of the NVRAM to monitor the volume and brightness settings, thinklight and many things more. It also lets us attach a shell command to the <ThinkPad> key. To write the information to blend the status informations into the screen tpb uses xosd.

An antiquated version of xosd exists as a Debian package. The latest version has, however, a few interesting features, which make it worthwhile to compile and install it manually. tpb does not exist as an official Debian package (yet), but we could easily build one from the tar-ball. Since we want to modify the source code, we prefer to stow this package manually.

The reason for our modification is that tpb segfaults, if we try to use a non-existent font for displaying. tpb sets -*-lucidatypewriter-medium-r-normal-*-*-250-*-*-*-*-*-* as the default fall-back font. Neither is this font guaranteed to exist nor does tpb check its availability. We should therefore change the default font in cfg.h to 10x20.

If we stow tpb, we must either set an alias or write a shell script to teach tpb the location of its default configuration file, which in our case is /usr/local/etc/tpbrc.

Compiling and Installing xosd:

```
#> cd /usr/local/packages
#> mkdir -p xosd-2.0.1/src
#> cd xosd-2.0.1/src
#> wget http://www.ignavus.net/xosd-2.0.1.tar.gz
#> tar xzf xosd-2.0.1.tar.gz
#> cd xosd-2.0.1
#> configure --prefix=/usr/local/packages/xosd-2.0.1
#> make
#> make install
#> cd ..
#> rm -rf xosd-2.0.1
#> cd /usr/local/stow
#> ln -s ../packages/xosd-2.0.1 .
#> stow -v xosd-2.0.1
```

Moficiation of cfg.h:

```
/** #define DEFAULT_OSDFONT "-*-lucidatypewriter-medium-r-normal-*-*-250-*-*-*-*-*-*" **/
#define DEFAULT_OSDFONT "10x20"
```

Compiling and Installing tpb:

```
#> mknod /dev/nvram c 10 144
#> cd /usr/local/packages/
#> mkdir -p tpb-0.4.1/src
#> cd tpb-0.4.1/src
#> wget http://www.savannah.nongnu.org/download/tpb/tpb-0.4.1.tar.gz
#> tar xzf tpb-0.4.1.tar.gz
#> cd tpb-0.4.1
#> ./configure --prefix=/usr/local/packages/tpb-0.4.1
#> make
#> make install
#> cd ..
#> rm -rf tpb-0.4.1
#> cd ../bin
```

```
#> mv tpb tpb.bin
#> vi tpb
#> cd /usr/local/stow
#> ln -s ../packages/tpb-0.4.1 .
#> stow -v tpb-0.4.1
```

/usr/local/packages/tpb-0.4.1/bin/tpb:

```
#!/bin/sh
/usr/local/bin/tpb.bin --config=/usr/local/etc/tpbrc $@ &
```

5.14. Intel Fortran Compiler

Since we use our machine for scientific number crunching, we need a decent Fortran compiler. The combo f2c plus fort77 is way too slow and g77 had a few numerical problems last time we checked.

For this reason we want to install the Intel Fortan Compiler ifc. And once more we are amazed how commercial products just fail to install without a pain in the ass. alien as well as rpm refuse to install the package, so we have to do it manually.

Compiling and Installing ifc:

```
#> cd /usr/local/packages
#>
#> mkdir -p ifc7-7.0/lib/ifc7/license
#> cp ~/l_for_38668925.lic ifc7-7.0/lib/ifc7/license
#> chmod +r ifc7-7.0/lib/ifc7/license
#> mkdir -p ifc7-7.0/doc/ifc7
#>
#> mkdir intel-install
#> cd intel-install
#> tar xzf ~/INTEL.TAR.GZ
#> alien -g intel-ifc7-7.0-64.i386.rpm
#> cd intel-ifc7-7.0.orig/opt/intel/compiler70/ia32
#> tar cf - . | (cd /usr/local/packages/ifc7-7.0; tar xf -)
#> cd ..
#> mv training docs
#> cd docs
#> tar cf - . | (cd /usr/local/packages/ifc7-7.0/doc/ifc7; tar xf -)
#> cd ..
#> tar cf - man | (cd /usr/local/packages/ifc7-7.0; tar xf -)
#> chown -R 0.0 *
#> cd doc/ifc
#> chmod -R -x *
#> find . -type d -exec chmod +x {} \;
#>
#> cd /usr/local/stow
#> ln -s ../ifc7-7.0 .
#> stow ifc7-7.0
#>
#> ldconfig
#>
#> rm -rf /usr/local/packages/intel-install
```

Now we have to edit the wrapper script, which is supposed to launch the compiler:

/usr/local/packages/ifc7-7.0/bin/ifc:

```
#!/bin/sh
INTEL_LICENSE_FILE=/usr/local/lib/ifc7/license;
export INTEL_LICENSE_FILE;
if [ -z LD_LIBRARY_PATH ]
then
 LD_LIBRARY_PATH=/usr/local/lib
else
```

```
 LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
fi
export LD_LIBRARY_PATH
export -n IA32ROOT; unset IA32ROOT;
if [ $# != 0 ]
then
 exec -a "/usr/local/bin/ifc" /usr/local/bin/ifcbin "$@";
else
 exec -a "/usr/local/bin/ifc" /usr/local/bin/ifcbin;
fi
exit
```

5. 15. Linux WLAN

Our router wants us to connect via DHCP. So we have to install the corresponding Debian package first:

Debian packages for DHCP:

```
dhcp-client
```

The linux-wlan Project is developing a complete, standards based, wireless LAN system for Linux. The corresponding Debian package is available in the “unstable” (a.k.a. “sid”) distribution, which means that we have to compile it from source.

Unfortunately, the kernel modules refuse to compile and install automagically under “woody”, so we have to lend a helpin’ hand and here and there. We have to edit the Debian rules file and then copy the resulting kernel modules.

Debian source packages for linux-wlan-ng:

```
wireless-tools_25-3
linux-wlan-ng_0.1.15-6
```

Installation of linux-wlan-ng:

```
#> dpkg-source -x wireless-tools_25-3.dsc
#> cd wireless-tools-25
#> dpkg-buildpackage
#> cd ..
#> mv wireless-tools_25-3_i386.deb /usr/local/debian/dists/woody/main/binary-i386
#> echo "wireless-tools optional local" >> /usr/local/debian/override.local.woody
#>
#> cd /usr/src/modules
#> dpkg-source -x linux-wlan-ng_0.1.15-6.dsc
#> cd linux-wlan-ng_0.1.15/
#> vi debian/rules                    # SEE BELOW
#> dpkg-buildpackage -d
#> cd ..
#> mv linux-wlan-ng_0.1.15-6_i386.deb /usr/local/debian/dists/woody/main/binary-i386
#> echo "linux-wlan-ng optional local" >> /usr/local/debian/override.local.woody
#>
#> cd ../linux
#> make-kpkg -rev Custom.1 modules_image
#> cd ..
#> mv linux-wlan-ng-modules-2.4.18_Custom.1+0.1.15-6_i386.deb /usr/local/debian/dists/woody/main/binary-
#> echo "linux-wlan-ng-modules-2.4.18 optional local" >> /usr/local/debian/override.local.woody
#>
#> cd /usr/local/debian
#> dpkg-scanpackages dists/woody/main/binary-i386/local/ override.local.woody > dists/woody/main/binary-
#>
#> apt-get update
#> apt-get upgrade
#>
#> cd /usr/src/modules/linux-wlan-ng-0.1.15/debian/tmp
#> tar cf - lib | (cd /; tar xf -)
```

```
#> depmod -a
```

Edit of /usr/src/modules/linux-wlan-ng-0.1.15/debian/rules:

```
# line 29:
#  export DH_OPTIONS=-p$(PACKAGE) --mainpackage=$(PACKAGE)
   export DH_OPTIONS=-p$(PACKAGE)
# line 105:
# kdist_config:
  kdist_configure:
```

Note: In later versions of the linux-wlan-ng package (e.g. version 0.2.0-9) the automagic installation of the kernel modules is completely f. . .ed up. We have to do it manually:

Manual installation of the linux-wlan-ng modules:

```
#> cd /lib/modules/2.4.18/
#> mkdir net pcmcia usb
#> cp /usr/src/modules/linux-wlan-ng-0.2.0/src/p80211/p80211.o net
#> cp /usr/src/modules/linux-wlan-ng-0.2.0/src/prism2/driver/prism2_pci.o net
#> cp /usr/src/modules/linux-wlan-ng-0.2.0/src/prism2/driver/prism2_plx.o net
#> cp /usr/src/modules/linux-wlan-ng-0.2.0/src/prism2/driver/prism2_cs.o pcmcia
#> cp /usr/src/modules/linux-wlan-ng-0.2.0/src/prism2/driver/prism2_usb.o usb
#>
#> depmod -a
```

In the next step, we need to tell the system about our new network interface and the kernel modules. To do so, we edit /etc/network/interfaces and /etc/modutils/linux-wlan-ng and then update the modules list.

/etc/network/interfaces:

```
auto lo
iface lo inet loopback
iface wlan0 inet dhcp
      wireless_essid <SSID here>
      wireless_mode managed
      wireless_nick <insert name here>
```

/etc/modutils/linux-wlan-ng:

```
alias wlan0 prism2_pci
```

Updating the modules list:

```
#> update-modules
```

If everything is okay, the wlan0 interface should now be brought up if we type ifup wlan0 and receive some IP address from the router in the 192.168.123.0 subnet of our WLAN.

This simple WLAN setup is then expanded and integrated into our nifty network-profile handling mechanism.

5.15.1. Special Case: The USR-2410 PC Card

To support this card by the linux-wlan-ng drivers, we have to edit some configuration files and roll our own scripts. First, we have to uncomment the appropriate device definitions in /etc/pcmcia/wlan-ng.conf, otherwise the standard kernel drivers will get loaded.

/etc/pcmcia/wlan-ng.conf:

```
card "Intersil PRISM2 Reference Design 11Mb/s 802.11b WLAN Card"
   version "INTERSIL", "HFA384x/IEEE"
   bind "prism2_cs"
card "Intersil PRISM2 Reference Design 11Mb/s WLAN Card"
   manfid 0x0156, 0x0002
   bind "prism2_cs"
```

In the next step, we disable the mass of half-integrated half-messed-up scripts to handle the PC Card automagically. This is done by diverting the original wlan-ng script and the recplacing it by an empty script:

Diverting /etc/pcmcia/wlan-ng:

```
#> dpkg-divert --rename /etc/pcmcia/wlan-ng
```

New /etc/pcmcia/wlan-ng:

```
#!/bin/sh
exit
```

The next problem is that the standard WLAN tools called by ifup don't handle the WEP encryption for the linux-wlang-ng drivers correctly. We have to circumvent this by resticting the interface definition in /etc/network/interfaces to the bare minimum and then use our won script to setup the card:

New entry in /etc/pcmcia/wlan-ng:

```
iface wlan0-pccard inet dhcp
```

/usr/local/bin/wlan0:

```
#!/bin/sh
prog=`basename $0`
test "X$SUPERCMD" = "X$prog" || exec /usr/bin/super $prog ${1+"$@"}
PATH="/bin:/usr/bin/:usr/local/bin:/sbin:/usr/sbin"
ADDRESS=""
SCHEMEFILE=/var/state/network/scheme.wlan0
DEVICE=wlan0
touch $SCHEMEFILE
. /etc/wlan/shared
case "$1" in
     load)
          modprobe wlan0
          ;;
     unload)
          modprobe -r wlan0
          ;;
     up|start)
          echo "wlan0: start"
          if is_true $WLAN_DOWNLOAD; then
            wlan_download $DEVICE
          fi
          wlan_enable $DEVICE
          wlan_source_config $DEVICE
          wlan_user_mibs $DEVICE
          wlan_wep $DEVICE
          if is_true $IS_ADHOC ; then
            wlan_adhoc $DEVICE
          else
            wlan_infra $DEVICE
          fi
          ifup wlan0 2> /dev/null
          ;;
     down|stop)
          echo "wlan0: down"
          wlan_disable $DEVICE
          ifdown wlan0 2> /dev/null
          rmmod -a; rmmod -a
          ;;
     restart)
          echo "wlan0: restart"
          ifdown wlan0 2> /dev/null
          sleep 1
          ifup wlan0 2> /dev/null
          ;;
     scheme)
          if [ -z "$2" ]; then
            cat $SCHEMEFILE | sed "s/-/ /" | awk '{print $2}'
          else
            echo "eth0-$" > $SCHEMEFILE
```

```
            fi
            ;;
        *)
         SCHEME=`cat $SCHEMEFILE | sed "s/-/ /" | awk '{print $2}'`
         SCHEMES=`grep iface /etc/network/interfaces | grep wlan0 | sed "s/-/ /" | awk '{
printf "%s ", $3}'`
         echo "Usage: wlan0 [up|start|down|stop|restart|scheme [xyz]]"
         echo "where xyz is one of:   $SCHEMES"
         echo "Current scheme is:     $SCHEME"
         ;;
esac
```

This script includes the standard linux-wlan-ng configuration files in /etc/wlan. We set the SSID of our access point in /etc/wlan/wlan.conf:

/etc/wlan/wlan.conf:

```
SSID_wlan0="myssid"
```

Then we copy the default configuration file /etc/wlan/wlancfg-DEFAULT to /etc/wlancfg-myssid and edit it to represent our detailed setup:

New /etc/wlan/wlancfg-myssid:

```
#=======USER MIB SETTINGS============================
# You can add the assignments for various MIB items
#  of your choosing to this variable, separated by
#  whitespace.  The wlan-ng script will then set each one.
# Just uncomment the variable and set the assignments
#  the way you want them.
#USER_MIBS="p2CnfRoamingMode=1 p2CnfShortPreamble=mixed"
#=======WEP==========================================
# [Dis/En]able WEP.  Settings only matter if PrivacyInvoked is true
lnxreq_hostWEPEncrypt=true      # true|false
lnxreq_hostWEPDecrypt=true      # true|false
dot11PrivacyInvoked=true        # true|false
dot11WEPDefaultKeyID=0          # 0|1|2|3
dot11ExcludeUnencrypted=true    # true|false, in AP this means WEP is required.
# If PRIV_GENSTR is not empty, use PRIV_GENTSTR to generate
#  keys (just a convenience)
PRIV_GENERATOR=/sbin/nwepgen    # nwepgen, Neesus compatible
PRIV_KEY128=false               # keylength to generate
PRIV_GENSTR=""
# or set them explicitly.  Set genstr or keys, not both.
dot11WEPDefaultKey0=01:02:03:04:05:06:07:08:09:0A:0B:0C:0D
dot11WEPDefaultKey1=11:12:13:14:15:16:17:18:19:1A:1B:1C:1D
dot11WEPDefaultKey2=21:22:23:24:25:26:27:28:29:2A:2B:2C:2D
dot11WEPDefaultKey3=01:32:33:34:35:36:37:38:39:3A:3B:3C:3D
#=======SELECT STATION MODE==================
IS_ADHOC=n                      # y|n, y - adhoc, n - infrastructure
#======= INFRASTRUCTURE STATION  ===================
# What kind of authentication?
AuthType="opensystem"           # opensystem | sharedkey (requires WEP)
#======= ADHOC STATION ============================
BCNINT=100                      # Beacon interval (in Kus)
CHANNEL=6                       # DS channel for BSS (1-14, depends
                                #   on regulatory domain)
BASICRATES="2 4"                # Rates for mgmt&ctl frames (in 500Kb/s)
OPRATES="2 4 11 22"             # Supported rates in BSS (in 500Kb/s)
```

After inserting the card nothing happens. We have to start and stop the interface manually by issuing the commands wlan0 up and wlan0 down, respectively.

5. 16. Cisco VPN Client

There exists a piece of software called vpnclient, which lets us establish a secure, end-to-end encrypted tunnel to any Cisco central site remote access VPN product. This piece of software does even exist for Linux. Of course it does not work out of the box (at least not our version 3.7.2).

After unpacking the tar-ball, we have a look at the file interceptor.c, more precisely at the function supported_device. For some unknown reason, only the devices eth0 to eth9 and ppp0 to ppp9 are assumed to be valid networking devices. wlan0 is not mentioned at all.

To remedy the situation, we replace the provided function with our own:

supported_device in interceptor.c:

```
static int inline supported_device(struct device* dev)
{
    if(!dev->name) return 0;
    if(!strncmp(dev->name,"eth",3) && (dev->name[3]>='0' && dev->name[3]<='9'))
       return 1;
    if(!strncmp(dev->name,"ppp",3) && (dev->name[3]>='0' && dev->name[3]<='9'))
       return 1;
    if(ippp_dev(dev)) {
       isdn_net_local *lp = (isdn_net_local *) dev->priv;
       if(lp->p_encap == ISDN_NET_ENCAP_SYNCPPP) return 1;
    }
    if(!strncmp(dev->name,"wlan",4) && (dev->name[4]>='0' && dev->name[4]<='9'))
       return 1;
    return 0;
}
```

In contrast to what the documentation claims, we don't need to load the provided kernel module at boot time. Our kernel is compiled to load its modules automagically at need. We just have to let it know, which module should be loaded:

/etc/modutils/vpn:

```
# Cisco VPN
alias cipsec0 cisco_ipsec
```

Now we update the modules with update-modules and are all set. We don't need the provided boot script /etc/init.d/vpnclient_init at all.

Later we integrate the VPN tunnel into our own network profile management.

5. 17. CD Recording as non-root User

Trying to burn CD-Rs as non-root user can be a tiring adventure, since the documentation of cdrecord does not really reflect the actual setup under Debian and is no longer entirely up to date. Furthermore, most of the tips and tricks divulged in the various mailing lists did not really impress by their insights.

To make a long story short, the probably best way to set things up is by using the super command. This makes sure that cdrecord is run with the real UID 0 and avoids all kind of hassles, among them the following error message:

cdrecord error message:

```
cdrecord.mmap: Operation not permitted. WARNING: Cannot set RR-scheduler
cdrecord.mmap: Permission denied. WARNING: Cannot set priority using setpriority().
cdrecord.mmap: WARNING: This causes a high risk for buffer underruns.
```

To do so, we write a small shell script named cdrecord, put it in /usr/local/bin and set execute permissions for all and everybody. Since the original cdrecord in /usr/bin does not have execute permissions for everybody, our ver-

sion in /usr/local/bin will automagically executed, if a non-root user not in the group cdrom types cdrecord at the command prompt.

/usr/local/bin/cdrecord:

```
#!/bin/sh
prog=`basename $0`
test "X$SUPERCMD" = "X$prog" || exec /usr/bin/super $prog ${1+"$@"}
/usr/bin/cdrecord $@
```

The corresponding entry in /etc/super.tab reads:

/etc/super.tab:

```
cdrecord   /usr/local/bin/cdrecord uid=root info="burn CD-R" \
           <registered user here>
```

5. 18. External Firewire Hard Disk

Although the 2.4 kernel series does not yet support hotplugging of SCSI and IEEE 1394 devices, we can easily configure for a simple manual setup. We make use of two scripts, namely rescan-scsi-bus.sh available from the IEEE 1394 For Linux project web pages and a homegrown helper script, to load the kernel necessary modules and scan for new SCSI devices:

/usr/local/bin/firewire:

```
#!/bin/sh
prog=`basename $0`
test "X$SUPERCMD" = "X$prog" || exec /usr/bin/super $prog ${1+"$@"}
PATH="/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin"
ADDRESS=""
case "$1" in
   up|start)
     modprobe ohci1394 && modprobe sbp2 && modprobe sd_mod
     sleep 1
     rescan-scsi-bus
     ;;
   down|stop)
     modprobe -r ohci1394 && modprobe -r sbp2 && modprobe -r sd_mod
     sleep 1
     rescan-scsi-bus -r
     ;;
   restart)
     rescan-scsi-bus -r
     sleep 1
     rescan-scsi-bus
     ;;
   *)
     echo "Usage: firewire [up|start|down|stop|restart]"
     ;;
esac
```

Note: We have renamed rescan-scsi-bus.sh into rescan-scsi-bus.

As before, we register our helper script with the super framework by adding the following lines to /etc/super.tab:

/etc/super.tab:

```
firewire     /usr/local/bin/firewire uid=root info="start/stop IEEE 1394" \
              <registered user here>
```

After physically plugging in the hard disk and launching the Firewire HD support by issuing the command firewire start, the disk shows up as /dev/sda and can be accessed as any other SCSI disk.

6

Manual System Configuration

6. 1. Environment

We find it amazing how broken the default setup is. Depending on how we login, we end up with a completely different environment. The PATH variable, for example, is unnecessarily set (or modified) by init, login and almost every shell profile. It's not too complicated to harmonise things a little bit.

/etc/environment:

```
# /etc/environment: default environment sourced by
#  - login sh/bash using '. /etc/environmnt'
#  - login csh/tcsh using 'eval `/usr/local/bin/readenv /etc/environment`'
#  - /etc/X11/Xsession.d/10-source-environment using '. /etc/environment'
#  set default locale
export LANG=C
#  set path
export PATH="/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin"
#  xprint
export XPSERVERLIST="`/etc/init.d/xprint get_xpserverlist`"
export XPRINTER=lp0
```

$HOME/.environment:

```
# $HOME/.environment: local environment sourced by
#  - login sh/bash using '. $HOME/.environmnt'
#  - login csh/tcsh using 'eval /usr/local/bin/readenv $HOME/.environment'
#  - $HOME/.Xsession using '. $HOME/.environment'
# add $HOME/bin to the path
export PATH="$HOME/bin:$PATH"
```

/etc/profile:

```
# /etc/profile: system-wide shell profile for
#  - interactive login bash
#  - all sh
#  source the default environment
if [ -f /etc/environment ]; then
  . /etc/environment
fi
#  find out, whether we are a BASH or interactive SH
#  BASH: source /etc/bash.bashrc
#  interactive SH: set prompt
case $BASH in
  *bash*)
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
    ;;
  *)
    case "$-" in
      *i*)
        if [ "`id -u`" -eq 0 ]; then
          PS1='# '
        else
          PS1='$ '
        fi
        export PS1
        ;;
      *)
        ;;
    esac
    ;;
esac
#  set default file permissions
umask 022
```

/etc/bash.bashrc:

```
# /etc/bash.bashrc: system wide shell profile for
#  - interactive non-login bash
#  set default prompt depending on UID and TERM
if [ "`id -u`" -eq 0 ]; then
  DELIM="# "
else
  DELIM="$ "
fi
if [ "$TERM" == "linux" ]; then
  PS1="\[\033[4m\]\h:\[\033[0m\] \w"$DELIM
else
  PS1="\[\033[4m\]\h:\[\033[0m\] "$DELIM
fi
export PS1
#  set shell option "check-window-size"
shopt -s checkwinsize
```

$HOME/.profile:

```
# $HOME/.profile: local shell profile for
#  - interactive login bash
#  - all sh
#  source the local environment
if [ -f $HOME/.environment ]; then
  . $HOME/.environment
fi
#  find out, whether we are called as BASH
#  BASH: source $HOME/.bashrc
case $BASH in
  *bash*)
```

```
    if [ -f $HOME/.bashrc ]; then
      . $HOME/.bashrc
    fi
    ;;
  *)
    ;;
esac
```

$HOME/.bashrc:

```
# $HOME/.bashrc: local shell profile for
#  - interactive non-login bash
#  set xterm titlebar
case "$TERM" in
  *xterm*)
    PROMPT_COMMAND='echo -e -n "\033]2;xterm@$HOSTNAME   $PWD\007"'
    export PROMPT_COMMAND
    ;;
  *)
    ;;
esac
#  set prompt depending on UID and TERM
if [ "`id -u`" -eq 0 ]; then
  DELIM="# "
else
  DELIM="$ "
fi
case "$TERM" in
  linux)
    export PS1="\[\033[1m\]\u\[\033[0m\]:\w "$DELIM
    ;;
  *term)
    export PS1="\[\033[1m\]\u\[\033[0m\] "$DELIM
    ;;
esac
#  set colours and options for 'ls'
eval `dircolors -b /etc/default/dircolors`
alias ls='ls -AF --color=auto'
#  aliases for safety
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

/etc/csh.login:

```
# /etc/csh.login: system-wide shell profile for
#  - login csh/tcsh
#  eval default environtment
if ( -f /etc/environment && -x /usr/local/bin/readenv ) then
  eval `/usr/local/bin/readenv /etc/environment`
endif
#  set default file permissions
umask 022
```

/etc/csh.cshrc:

```
# /etc/csh.cshrc: system-wide shell profile for
#  - all csh/tcsh
#  find out whether we are an interactive tcsh
#  interactive tcsh: set prompt according to UID and TERM
if ( `id -u` ) then
  set delim = ">"
else
  set delim = "#"
endif
if ! ( $?0 && $?tcsh ) then
  bindkey "\e[1~" beginning-of-line  # Home
  bindkey "\e[7~" beginning-of-line  # Home rxvt
```

```
  bindkey "\e[2~" overwrite-mode     # Ins
  bindkey "\e[3~" delete-char        # Delete
  bindkey "\e[4~" end-of-line        # End
  bindkey "\e[8~" end-of-line        # End rxvt
  if ( "$?TERM" == "linux" ) then
    set prompt = "%U%m%u:%B%/%b"$delim" "
  else
    set prompt = "%U%m%u:"$delim" "
  endif
endif
```

$HOME/.login:

```
# $HOME/.login: local shell profile for
#  - login csh/tcsh
#  eval the local environment
if ( -f $HOME/.environment && -x /usr/local/bin/readenv ) then
  eval `/usr/local/bin/readenv $HOME/.environment`
endif
```

$HOME/.cshrc:

```
# $HOME/.cshrc: local shell profile for
#  - all csh/tcsh
#  find out whether we are an interactive tcsh
if ! ( $?0 && $?tcsh ) then
#  set xterm titlebar
switch ( $TERM )
  case *xterm*:
    alias precmd 'echo -n "\033]2;xterm@${HOST}    $cwd\007"'
    breaksw
  default:
    breaksw
endsw
#  prompt
if ( $TERM == "linux" ) then
  set prompt = "%B%n%b %/ > "
else
  set prompt = "%B%n%b > "
endif
#  set colours and options for 'ls'
eval `dircolors /etc/default/dircolors`
alias ls 'ls -F --color=auto'
#  aliases for safety
alias rm 'rm -i'
alias cp 'cp -i'
alias mv 'mv -i'
#--- endif
endif
```

/etc/X11/Xession.d/10-source-environment:

```
# /etc/X11/Xdession.d/10-source-environment: global X session startup script
# source the default environment
if [ -f /etc/environment ]; then
  . /etc/environment
fi
```

$HOME/.Xsession:

```
#!/bin/sh
# $HOME/.Xsession: local X session startup script
#  setup environment
if [ -f $HOME/.environment ]; then
  . $HOME/.environment
fi
#  setup some fancy desktop background
xsetbg /usr/share/backgrounds/propaganda/vol13.5/A-Little-Exercise-1.JPG
```

```
#  start fvwm as window/session manager
fvwm
```

Note: Contrary to popular belief, the existence of the environment variable PS1 does not indicate an interactive shell. If the current shell is the non-interactive child of an interactive shell, it inherits PS1 from the parent shell. We have to test the shell flags for interactivity.

Source of /usr/local/bin/readenv:

```
/* readenv.c
 * a little hack to read a file containing statements like
 * export VAR=VALUE
 * and then print them as
 * setenv VAR VALUE
 * to stdout.
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main (int argc, char **argv)
{
   char string[2048], *var, *val;
   FILE *file;

   if ( argc != 2 ) return EXIT_FAILURE;
   if ( (file = fopen(argv[1], "r")) == NULL ) return EXIT_FAILURE;

   do
   {
      fgets(string, sizeof(string), file);
      if ( !feof(file) )
      {
         val = strstr(string, "export");
         strsep(&val, " ");
         var = strsep(&val, "=");
         if ( var && val ) printf("setenv %s %s;", var, val);
      }
   }
   while (!feof(file));

   fclose(file);

   return EXIT_SUCCESS;
}
```

Note: If we replace export and setenv with alias, this program can also be used to read a shell independent alias definition file.

6. 2. Network Profile Management

Originally, we had installed a nifty package called laptop-net, which was supposed to autodetect to which network our machine was connected to and then automagically reconfigure the system correspondingly. Unfortunately, this package never worked as advertised and the package maintainer did not bother to answer emails. We therefore decided to scrap laptop-net and roll our own replacement.

Similarly as the laptop-net package, we make the distinction between a *network scheme* and a *network profile*. A scheme describes the network interface and the network connection, whereas a profile describes the system configuration, which is selected for a given scheme. The advantage of this setup is that we can this way easily map the same profile to different schemes or different profiles to the same scheme.

The different schemes are defined in the network interface description file /etc/network/interfaces:

/etc/network/interfaces:

```
# List of automatically initialised interfaces:
auto lo
# Definition of the loop-back interface:
iface lo inet loopback
# Definition of the WLAN0 interface and scheme mappings:
mapping wlan0
  script /usr/local/bin/netscheme
# Definition of the WLAN0 scheme "wlan0-home":
iface wlan0-home inet dhcp
  wireless_mode managed
  wireless_nick <nick>
  wireless_essid <essid>
  wireless_channel <channel>
  wireless_enc on
  wlan_ng_authtype opensystem
  wlan_ng_priv_key128 true
  wlan_ng_key0 01:02:03:04:05:06:07:08:09:0A:0B:0C:0D
  wlan_ng_key1 11:12:13:14:15:16:17:18:19:1A:1B:1C:1D
  wlan_ng_key2 21:22:23:24:25:26:27:28:29:2A:2B:2C:2D
  wlan_ng_key3 31:32:33:34:35:36:37:38:39:3A:3B:3C:3D
  up /usr/local/bin/profile-select
  down /usr/local/bin/profile-deselect
# Definition of the ETH0 interface and scheme mappings:
mapping eth0
  script /usr/local/bin/netscheme
# Definition of the ETH0 scheme "eth0-work":
iface eth0-work inet static
  address xxx.yyy.zzz.149
  netmask 255.255.255.0
  broadcast xxx.yyy.zzz.255
  network xxx.yyy.zzz.0
  gateway xxx.yyy.zzz.254
  dns_domain <fqdn>
  dns_nameservers xxx.yyy.1.1 xxx.yyy.1.5
  up /usr/local/bin/profile-select
  down /usr/local/bin/profile-deselect
# Definition of the ETH0 scheme "eth0-local":
iface eth0-local inet static
  address 10.0.0.1
  netmask 255.255.255.0
  broadcast 10.0.0.255
  network 10.0.0.0
  gateway 10.0.0.2
  up /usr/local/bin/profile-select
  down /usr/local/bin/profile-deselect
# Definition of the ETH0 scheme "eth0-dhcp":
iface eth0-dhcp inet dhcp
  up /usr/local/bin/profile-select
  down /usr/local/bin/profile-deselect
```

The script /usr/local/bin/netscheme reads the current network scheme for the given interface, which is saved in the file /var/stete/network/scheme.$INTERFACE.

/usr/local/bin/netscheme:

```
#!/bin/sh
INTERFACE=$1
SCHEMEFILE=/var/state/network/scheme.$INTERFACE
touch $SCHEMEFILE
cat $SCHEMEFILE
```

The two interfaces eth0 and wlan0 and their associated schemes are managed by the two scripts /usr/local/bin/eth0 and /usr/local/bin/wlan0, respectively, which differ only in the definition of the assigned interface.

/usr/local/bin/eth0:

```
#!/bin/sh
prog=`basename $0`
test "X$SUPERCMD" = "X$prog" || exec /usr/bin/super $prog ${1+"$@"}
PATH="/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin"
ADDRESS=""
INTERFACE=eth0
#INTERFACE=wlan0
SCHEMEFILE=/var/state/network/scheme.$INTERFACE
touch $SCHEMEFILE
case "$1" in
   load)
     modprobe $INTERFACE
     ;;
   unload)
     modprobe -r $INTERFACE
     ;;
   up|start)
     echo "$INTERFACE: start"
     ifup $INTERFACE 2> /dev/null
     ;;
   down|stop)
     echo "$INTERFACE: stop"
     ifdown $INTERFACE 2> /dev/null
     rmmod -a; rmmod -a
     ;;
   restart)
     echo "$INTERFACE: restart"
     ifdown $INTERFACE 2> /dev/null
     sleep 1
     ifup $INTERFACE 2> /dev/null
     ;;
   scheme)
     if [ -z "$2" ]; then
       cat $SCHEMEFILE | sed "s/-/ /" | awk '{print $2}'
     else
       echo "$INTERFACE-$2" > $SCHEMEFILE
     fi
     ;;
   *)
     SCHEME=`cat $SCHEMEFILE | sed "s/-/ /" | awk '{print $2}'`
     SCHEMES=`grep iface /etc/network/interfaces | grep $INTERFACE | sed "s/-/ /" | awk '{printf "%s ",
     echo "Usage: $INTERFACE [up|start|down|stop|restart|scheme [xyz]]"
     echo "where xyz is one of:  $SCHEMES"
     echo "Current scheme is:    $SCHEME"
     ;;
esac
```

After the interface was brought up with the command wlan0 up, the script profile-select tries to match the assigned IP number against a list of possible IP numbers to select the appropriate network profile. This profile is the installed by copying configuration files and by starting or stopping system services. After the interface was brought down, the script profile-deselect reverts to a special default profile called *offline*, which does not map to a network scheme.

/usr/local/packages/network/bin/profile-select:

```
#!/bin/sh
PROFILE_DIR="/etc/network/profiles"
PROFILE_STATE_FILE="/var/run/network/profile"
# Load current profile
PROFILE_CUR=`cat $PROFILE_STATE_FILE | awk '{print $3}'`
PROFILE_NEW="offline"
# Select new profile
ADDRESS=`ifconfig $IFACE | grep inet | sed "s/:/ /" | awk '{print $3}'`
for PROFILE in $(cd $PROFILE_DIR; ls -d [a-z0-9]* 2> /dev/null); do
  PATTERNS="$PROFILE_DIR/$PROFILE/patterns"
```

```
  if [ -r "$PATTERNS" ]; then
    for PATTERN in $(cat $PATTERNS); do
      case $ADDRESS in
        ($PATTERN)
          PROFILE_NEW=$PROFILE
          break 2
          ;;
      esac
    done
  fi
done
echo -n "Switching to profile "$PROFILE_NEW": "
# If the new and current profiles are the same, exit here
[ "$PROFILE_NEW" != "$PROFILE_CUR" ] || exit 0
# Run the Stop-Scripts for the current profile
cd $PROFILE_DIR/$PROFILE_CUR/rc.d
for SCRIPT in $(echo "K[0-9][0-9]*"); do
  if [ -n "$SCRIPT" ]; then
    echo -n "."
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT stop &> /dev/null
        ;;
      (*)
        ./$SCRIPT stop &> /dev/null
        ;;
    esac
  fi
done
# Copy all profile specific files for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/files.d
tar cf - . | (cd /; tar xf -)
# Run Start-Scripts for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/rc.d
for SCRIPT in $(echo "S[0-9][0-9]*"); do
  if [ -n "$SCRIPT" ]; then
    echo -n "."
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT start &> /dev/null
        ;;
      (*)
        ./$SCRIPT start &> /dev/null
        ;;
    esac
  fi
done
# Save new profile
echo $PROFILE_CUR" --> "$PROFILE_NEW > $PROFILE_STATE_FILE
echo " done."
```

/usr/local/packages/network/bin/profile-deselect:

```
#!/bin/sh
PROFILE_DIR="/etc/network/profiles"
PROFILE_STATE_FILE="/var/run/network/profile"
# Load the current and new profile
PROFILE_CUR=`cat $PROFILE_STATE_FILE | awk '{print $3}'`
PROFILE_NEW="offline"
echo -n "Switching to profile "$PROFILE_NEW": "
# If the new and current profiles are the same, exit here
[ "$PROFILE_NEW" != "$PROFILE_CUR" ] || exit 0
# Run Stop-Scripts for the current profile
cd $PROFILE_DIR/$PROFILE_CUR/rc.d
for SCRIPT in $(echo K[0-9][0-9]*); do
  if [ -n "$SCRIPT" ]; then
    echo -n "."
```

```
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT stop &> /dev/null
        ;;
      (*)
        ./$SCRIPT stop &> /dev/null
        ;;
    esac
  fi
done
# Copy all profile specific files for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/files.d
tar cf - . | (cd /; tar xf -)
# Run the Start-Scripts for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/rc.d
for SCRIPT in $(echo S[0-9][0-9]*); do
  if [ -n "$SCRIPT" ]; then
    echo -n "."
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT start &> /dev/null
        ;;
      (*)
        ./$SCRIPT start &> /dev/null
        ;;
    esac
  fi
done
# Save new profile
echo $PROFILE_CUR" --> "$PROFILE_NEW > $PROFILE_STATE_FILE
echo " done."
```

In contrast to the laptop-net package, where a special daemon is supposed to watch the ethernet interface for network connects and disconnects, we have to bring up our interfaces manually using the two scripts eth0 and wlan0. This is no great inconvenience, since this ifd daemon is one part of the laptop-net package, which does not work as advertised.

The actual network profiles are defined in the directory /etc/network/profiles. Each profile corresponds to a subdirectory, whose name is the name of the profile. Within this subdirectory, there are two more subdirectories named files.d and rc.d, repectively, as well as one text file name patters.

The patterns file is a simple list of IP numbers (possibly containing the wildcard *, one IP number per line), to which the corresponding profile should apply:

/etc/network/profiles/at-work/patterns:

```
xxx.yyy.zzz.*
XXX.YYY.ZZZ.34
```

The files.d subdirectory contains a tree of more subdirectories and files. This tree is copied as a whole into the root directory / using the tar command just after the network interface has been brought up. Subsequently, all system services named Snn_service in the rc.d directory are started. After the interface has been brought down, all system services named Knn_service are stopped.

The offline profile is special and gets selected if no network scheme is active. This profile may contain the files.d and rc.d directories. The patterns file is not necessary.

Now we want to transfer control of all network related system startup scripts to our network profile management — without confusing the Debian package management of course. To do so, we have to remove the symlinks from the /etc/rcN.d directories and install them in the /etc/network/profiles/XXXX/rc.d, where XXXX is the name of our profiles. Since the Debian boot-script manager update-rc.d will reinstall these links as long as it find the corresponding script in /etc/init.d, we have to divert these to a different location.

Diverting the boot scripts from init:

```
#!/bin/sh
mkdir /etc/network/init.d
PACKAGES="dns-clean fetchmail ippl lprng ntp ntpdate ppp scandetd ssh xinetd"
for p in $PACKAGES; do
  dpkg-divert --rename --divert /etc/network/init.d/$p /etc/init.d/$p
  update-rc.d $p remove
done
```

Now we go to the rc.d directories and create the new symlinks by hand.

Creating the new startup/shutdown symlinks (example):

```
#> cd /etc/network/profiles
#> mkdir -p XXXX/rc.d YYYY/rc.d ZZZZ/rc.d
#> cd XXXX/rc.d
#> ln -s ../../../init.d/ssh S20ssh
#> ln -s ../../../init.d/ssh K20ssh
```

To integrate the VPN tunnel into this network profile management, we define a special network profile vpn, which is not mapped to a scheme but can simply replace the current profile on a given scheme.

To initialise the VPN tunnel and to switch to the corresponding profile, we use a simple script:

/usr/local/bin/vpn:

```
#!/bin/sh
prog=`basename $0`
test "X$SUPERCMD" = "X$prog" || exec /usr/bin/super $prog ${1+"$@"}
PATH="/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin"
ADDRESS=""
export DISPLAY=":0.0"
case "$1" in
   up|start)
     /usr/local/bin/xterm -geometry 156x26-0-0 -e vpnclient connect XXXX &
     while [ "$ADDRESS" == "" ]; do
       ADDRESS=`vpnclient stat tunnel | grep "Client address" | awk '{print $3}'`
       sleep 1
     done
     /usr/local/packages/network/bin/vpn-select
     ;;
   down|stop)
     kill `pidof vpnclient` &> /dev/null
     kill `pidof vpn` &> /dev/null
     /usr/local/packages/network/bin/vpn-deselect
     ;;
   *)
     echo "Usage: vpn [up|start|down|stop]"
     ;;
esac
```

Note: This script sets DISPLAY=:0.0" and assumes that root has access to this display. Note also: XXXX is the name of the configured VPN profile,

We have to register this script with the super package by adding the following stanza to the configuration file:

/usr/local/bin/vpn:

```
vpn        /usr/local/bin/vpn uid=root info="start/stop VPN" \
           <user>
```

The two scripts vpn-select and vpn-deselect have a similar functionality to profile-select and profile-deselect whith the only exception that we don't have to map a profile but select the special profile vpn after the VPN tunnel has been established and revert to the previously active profile after the VPN tunnel has been closed.

/usr/local/packages/bin/vpn-select:

```
#!/bin/sh
PROFILE_DIR="/etc/network/profiles"
PROFILE_STATE_FILE="/var/run/network/profile"
VPN_STATE_FILE="/var/run/network/vpn"
# Load current profile
PROFILE_CUR=`cat $PROFILE_STATE_FILE | awk '{print $3}'`
PROFILE_NEW="offline"
# Select new profile
ADDRESS=`vpnclient stat tunnel | grep "Client address" | awk '{print $3}'`
for PROFILE in $(cd $PROFILE_DIR; ls -d [a-z0-9]* 2> /dev/null); do
  PATTERNS="$PROFILE_DIR/$PROFILE/patterns"
  if [ -r "$PATTERNS" ]; then
    for PATTERN in $(cat $PATTERNS); do
      case $ADDRESS in
        ($PATTERN)
          PROFILE_NEW=$PROFILE
          break 2
          ;;
      esac
    done
  fi
done
echo "PROFILE: " $PROFILE_NEW
# If the new and current profiles are the same, exit here
[ "$PROFILE_NEW" != "$PROFILE_CUR" ] || exit 0
# Run the Stop-Scripts for the current profile
cd $PROFILE_DIR/$PROFILE_CUR/rc.d
for SCRIPT in $(echo "K[0-9][0-9]*"); do
  if [ -n "$SCRIPT" ]; then
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT stop
        ;;
      (*)
        ./$SCRIPT stop
        ;;
    esac
  fi
done
# Copy all profile specific files for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/files.d
tar cf - . | (cd /; tar xf -)
# Run Start-Scripts for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/rc.d
for SCRIPT in $(echo "S[0-9][0-9]*"); do
  if [ -n "$SCRIPT" ]; then
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT start
        ;;
      (*)
        ./$SCRIPT start
        ;;
    esac
  fi
done
# Save new profile
echo $PROFILE_CUR" --> "$PROFILE_NEW > $VPN_STATE_FILE
```

/usr/local/packages/bin/vpn-deselect:

```
#!/bin/sh
PROFILE_DIR="/etc/network/profiles"
PROFILE_STATE_FILE="/var/run/network/profile"
VPN_STATE_FILE="/var/run/network/vpn"
```

```
# Load the current and new profile
PROFILE_CUR=`cat $VPN_STATE_FILE     | awk '{print $3}'`
PROFILE_NEW=`cat $PROFILE_STATE_FILE | awk '{print $3}'`
echo "PROFILE: " $PROFILE_NEW
# If the new and current profiles are the same, exit here
[ "$PROFILE_NEW" != "$PROFILE_CUR" ] || exit 0
# Run Stop-Scripts for the current profile
cd $PROFILE_DIR/$PROFILE_CUR/rc.d
for SCRIPT in $(echo K[0-9][0-9]*); do
  if [ -n "$SCRIPT" ]; then
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT stop
        ;;
      (*)
        ./$SCRIPT stop
        ;;
    esac
  fi
done
# Copy all profile specific files for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/files.d
tar cf - . | (cd /; tar xf -)
# Run the Start-Scripts for the new profile
cd $PROFILE_DIR/$PROFILE_NEW/rc.d
for SCRIPT in $(echo S[0-9][0-9]*); do
  if [ -n "$SCRIPT" ]; then
    case "$SCRIPT" in
      (*.sh)
        /bin/sh ./$SCRIPT start
        ;;
      (*)
        ./$SCRIPT start
        ;;
    esac
  fi
done
# Save new profile
echo $PROFILE_CUR" --> "$PROFILE_NEW > $VPN_STATE_FILE
```

Last but not least we need a small init script, which brings our network profile management into a well defined state upon booting the machine:

/etc/init.d/network_profile:

```
#!/bin/sh
if [ ! -d /var/run/network ]; then
  mkdir /var/run/network
fi
echo "offline --> offline" > /var/run/network/profile
echo "offline --> offline" > /var/run/network/vpn
```

We link this new script into the runlevel directories /etc/rcN.d either by hand or using the update-rc.d command.

6.3. X Fontpath Configuration

First of all, there is no real need to install a font server on our laptop. Font servers are typically needed, where we have a bunch of diskless graphics terminals (X terminals), which need to download the fonts from somewhere. But our laptop will certainly not play the font server for somesuch terminals. If available on the local network, we may want to configure for such an external font server, though.

Our display resolution is 133 dpi. The pixmap fonts of XFree86 come, however, in two flavours only: 75 dpi and 100 dpi. We can immediately forget about installing the 75 dpi fonts and settle on the 100 dpi fonts. But even

these fonts turn out to be somewhat small when displayed at 133 dpi. Scaling bitmapped fonts is a big no-no, though. It's therefore a good idea to install some natively scalable fonts, e.g. Type 1, Speedo and TrueType fonts.

Since that company, whose name reminds us at a brand of toilet paper, has removed its hitherto freely downloadable TrueType fonts, we have to install the msttcorefonts package from the testing distribution, if we want those fonts. Using the Debian font manager package defoma, the installation of the Type 1, Speedo and TrueType fonts is painless. To make use of all these fonts, we only have to adapt the X server configuration file /etc/X11/XF86Config-4:

/etc/X11/XF86Config-4:

```
Section "Files"
# External Fontserver:
#       FontPath   "tcp/fontserver:7100"
# Local Fontserver:
#        FontPath   "unix/:7100"
# Unscaled Bitmap Fonts:
        FontPath   "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
        FontPath   "/usr/X11R6/lib/X11/fonts/misc:unscaled"
# Scalable TrueType and CID Fonts (via DeFoMa):
        FontPath   "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
        FontPath   "/var/lib/defoma/x-ttcidfont-conf.d/dirs/CID"
# Scalable Speedo and Type1 Fonts:
        FontPath   "/usr/X11R6/lib/X11/fonts/Speedo"
        FontPath   "/usr/X11R6/lib/X11/fonts/Type1"

# Scaled Bitmap Fonts:
#       FontPath   "/usr/X11R6/lib/X11/fonts/75dpi"
#       FontPath   "/usr/X11R6/lib/X11/fonts/misc"
EndSection
```

Note: We should replace fontserver with the name of our external fontserver (if any). Removing the entry for the local fontserver, if there is none installed, speeds up things considerably. Scaling bitmapped fonts is ugly.

6. 4. Installing the Cyberbit Unicode TrueType Fonts

As far as we know, the Cyberbit fonts are among the most complete Unicode TrueType fonts (containg nearly 30000 glyphs). It's therefore a good idea to install them.

Installing CyberBit TrueType Fonts:

```
#> mkdir -p /usr/local/share/fonts/truetype
#> cd /usr/local/share/fonts/truetype
#> wget ftp://ftp.netscape.com/pub/communicator/extras/fonts/windows/Cyberbit.ZIP
#> unzip Cyberbit.ZIP
#> mv Cyberbit.ttf CyberBit.ttf
#> rm Cyberbit.ZIP
#> defoma-hints truetype CyberBit.ttf >> localfont.hints
```

Now we have to edit the hint-file manually. Unfortunately, the defoma documentation is somewhat lacking what concerns the installation of large Unicode fonts spanning many codesets. Our hint file is therefore largely based on guesswork.

/usr/local/share/fonts/truetype/localfont.hints:

```
category truetype
begin /usr/local/share/fonts/truetype/CyberBit.ttf
  Family = BitstreamCyberbit
  FontName = BitstreamCyberbit-Roman
  Encoding = Unicode
  Location = English
  Charset = ISO10646-1
  UniCharset = ISO8859-1 ISO8859-2 ISO8859-3 ISO8859-4 ISO8859-5 ISO8859-6 ISO8859-7 ISO8859-8 ISO8859-9
  GeneralFamily = Roman Mincho Symbol
```

```
  Weight = Medium
  Width = Variable
  Shape = Serif Upright
  Foundry = bitstream
  Priority = 20
  X-Foundry = bitstream
  X-Family = cyberbit
  X-Weight = medium
  X-Slant = r
end
```

Finally, we register the fonts with defoma.

Registering the Cyberbit Fonts with defoma:

```
#> defoma-font -v register-all /usr/local/share/fonts/truetype/localfont.hints
```

6.5. Configuring the Syslog Daemon

Debian's default configuration of the syslog daemon is a mess. Fortunately, the rest of Debian (including the configuration of cron) is very well done, so that it is extremely painless to switch to a different configuration.

/etc/syslog.conf:

```
#
# SYSLOG by facility
#
auth,authpriv,security.*    -/var/log/service.auth
cron.*                      -/var/log/service.cron
daemon.*                    -/var/log/service.daemon
kernel.*                    -/var/log/service.kernel
lpr.*                       -/var/log/service.lpr
mail.*                      -/var/log/service.mail
news.*                      -/var/log/service.news
syslog.*                    -/var/log/service.syslog
uucp.*                      -/var/log/service.uucp
local0.*                    -/var/log/service.local0
local1.*                    -/var/log/service.local1
local2.*                    -/var/log/service.ppp
local3.*                    -/var/log/service.local3
local4.*                    -/var/log/service.local4
local5.*                    -/var/log/service.local5
local6.*                    -/var/log/service.local6
local7.*                    -/var/log/service.local7
#
# SYSLOG by level
#
*.debug                     -/var/log/level.debug
*.warning                   -/var/log/level.warning
*.error                      /var/log/level.error
*.crit                       /var/log/level.critical
*.alert                      /var/log/level.alert
*.emerg                     *
```

Reconfiguring the Syslog Daemon:

```
#> /etc/init.d/sysklogd stop
#> rm `syslogd-listfiles -a`
#> dpkg-divert --rename /etc/service.conf
#> vi /etc/syslog.conf
#> /etc/init.d/sysklogd start
```

6. 6. Three Finger Salute

We want to be able to shutdown or reboot our machine from the text console by pressing <CTRL-ALT-DEL> and <CTRL-ALT-INS>, respectively. Both can be configured in /etc/inittab, the former one directly, the latter one by using the kbdrequest feature and defining the key KeyboardSignal as <CTRL-ALT-INS> in the kernel keymaps.

/etc/console/linux-kbdrequest.inc:

```
control alt keycode 110 = KeyboardSignal
```

Modification of /etc/console/uk.kmap:

```
include "linux-kbdrequest"
```

Modification of /etc/inittab:

```
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now
# Action on special keypress (CTRL-ALT-INS)
kb::kbrequest:/sbin/shutdown -t1 -a -r now
```

6. 7. Extending a Logical Volume

Sonner or later we will run out of space on our /home partition. No problem: We just extend the corresponding Logical Volume. Doing so is extremely simple and easy.

Extending the Logical Volume /dev/disk/home:

```
#> umount /home
#> tune2fs -O ^has_journal
#> e2fsadm -L +10G /dev/disk/home
#> tune2fs -O has_journal
#> mount /home
```