CHAPTER 31

Optimizing Your System

One slight problem with SUSE Linux (and all Linux distributions) is that they take a "one-size-fits-all" approach—the default installation attempts to provide services for every kind of user. While this offers the widest range of compatibility, it doesn't always ensure an optimized system.

You may never attach a printer to SUSE Linux, for example, so what's the point of keeping the printing subsystem in memory? You can remove it from your Linux setup and not only free memory, but also speed up boot times, because you no longer need to wait for the printer service to start. While this might save only a couple of seconds, or just a couple of hundred kilobytes of memory, repeating the process and paring SUSE Linux down to the bone can produce an ultra-efficient system.

In this chapter, you'll learn how to target the various subsystems of your Linux system in order to optimize and speed up your computer. We'll look at everything from bootup, to hard disks, to streamlining the kernel itself.

Speeding Up Booting

Let's take a look at what happens when a SUSE Linux-equipped PC boots. Then we'll explore some ways to speed up the process.

Understanding Bootup

When you start your computer, and after the computer's power-on self-tests (POST), the computer's BIOS searches for a boot program on the hard disk. If SUSE Linux is installed on the computer, the boot program then runs the GRUB boot loader program. The job of the GRUB boot loader is twofold. First, it displays a menu from which you can choose which operating system to load. If you've installed SUSE Linux alongside Windows, you'll be able to choose between the two operating systems at this stage. Second, GRUB loads the Linux kernel, if a Linux operating system is chosen from the boot menu.

Once loaded, the kernel then starts the very first program that's run on any SUSE Linux system: `init`. The principal job of `init` is to run a variety of run-level scripts, which load the hardware and software necessary for the full and correct functioning of the system. All of these are located in `/etc/init.d` or its subdirectories.

■ **Note** In this context, *scripts* can be defined as long chains of commands stored in a single file.

The first of these scripts is `/etc/init.d/boot`. This starts and configures the essential system hardware that will be used across all run levels. It also runs some additional startup scripts contained in the `/etc/init.d/boot.d` directory.

Following this, `/etc/init.d/boot.local` is run. This can contain additional hardware-oriented scripts placed there by the user, which are designed to be activated before the run level is entered. If you are running a nonstandard piece of hardware, you might need to use `boot.local`, but this is rare. On my test system, the `boot.local` file was empty, aside from a few comments at the top stating how the file is used.

■ **Note** Comments within scripts are usually preceded by a hash (#) or sometimes a semicolon (;), which tells the computer not to interpret what follows on that line as an actual command. Comments are inserted to help you, as a user, understand what the script is supposed to do.

Following this, the run-level scripts are activated. These are located within the `/etc/init.d` directory but are referenced from symbolic links within the `/etc/init.d/rcX.d` directory, where `X` refers to the run level in use. Figure 31-1 shows the scripts at run level 5.

Figure 31-1. *The scripts for each run level are contained in the /etc/init.d/rcX.d directories, where X is the run-level number.*

Run-level scripts do two things: They start and configure any hardware that's specific to that particular run level. and they start any necessary system software (known as *services*) that is particular to that run level.

■ **Note** Because the run level defines what software and hardware are in use on the computer, the run level could accurately be referred to as the *operating mode* of the computer.

For example, if run level 5 is activated, the X Display Manager (`xdm`) script will be run to start the GUI. `xdm` is missing from run level 3's scripts, because run levels 1 and 3 are usually configured to boot to a command-line prompt.

■ **Note** In other parts of this book, I've referred to the GNOME and KDE Display Manager programs (`gdm` and `kdm`, respectively), rather than `xdm`. As far as run levels are concerned, `xdm` is the script responsible for starting `gdm` and `kdm`, depending on which desktop environment is installed (technically speaking, `xdm` merely starts whichever display manager is listed within `/etc/sysconfig/displaymanager`).

In terms of operating system components, it's during the run-level script stage that things like the printing services are started and network file sharing subsystems are made to run in the background. If the system is configured as a web server, software like Apache will be run at this stage, too.

On SUSE Linux, run levels 2 through 5 are defined as *multiuser*. Technically speaking, this means that they allow more than one user to log on, but for most desktop users, they're better defined as the day-to-day running modes of the computer. Run level 5 is the default run level under SUSE Linux. See Table 31-1 for a list of run levels and what they normally do.

■ **Note** There's no reason why run levels 7, 8, and 9 can't be used under SUSE Linux. However, only run levels 0 through 6 are normally configured.

Table 31-1. *SUSE Linux Run Levels*

Run Level	Description
0	Halt; the computer will be shut down.
1	Single-user mode (root login); very few hardware and software services are activated. Normally used for troubleshooting.
2	Non-GUI; network interface deactivated. Multiple user logins allowed, unlike with run level 1.
3	Non-GUI; standard set of hardware and system services started, including network services.

Continues

Table 31-1. *Continued*

Run Level	Description
4	Unused.
5	GUI; default run level for SUSE Linux. Standard set of hardware and system services started.
6	Reboot; the computer will be shut down and then restarted.
S	Single-user mode; used when booting directly into single-user mode, rather than switching to single-user mode (run level 1).

The job of the run-level scripts is to define the user experience. When they've finished, the computer will be configured and ready for use in some fashion. However, that's not quite the end of things.

Although you'll be able to log on, the KDE or GNOME desktops have yet to start, and these, too, have their own set of initialization processes. They need to start their own set of programs, such as system tray/notification area applets, which provide handy functions like on-screen volume control. Once all that has finished, you can use the computer!

Because so much must take place for your system to come to life, booting SUSE Linux can take some time. On my test system, it averaged between one and two minutes. Certainly, you can shave some time from this.

■ **Note** Run levels don't just contain startup scripts. They also contain "kill" scripts, designed to shut down services should you change into a particular run level from another. For example, run level 3 contains a kill script designed to terminate `xdm`, so that the GUI is no longer active. However, there's no need to worry about kill scripts when considering how to optimize the SUSE Linux boot process.

Reducing the Boot Menu Delay

Getting rid of the GRUB boot menu delay can save some waiting around in the early stages of the boot process. The delay can be reduced to a few seconds, or even eradicated completely. Of course, in such a case, you won't be able to choose which operating system you want to load if you're dual-booting with Windows. Even if SUSE Linux is the only operating system on your computer, without the boot delay, you won't have the chance to boot into recovery mode, as offered on the GRUB menu. So you need to consider whether this is a worthwhile time-saving measure.

YaST offers a way of controlling the GRUB menu delay. Here's the procedure:

1. To start YaST, click Desktop ➤ YaST under GNOME; under KDE, click K menu ➤ System ➤ YaST. Enter your root password when prompted.
2. Click the System icon on the left, and then click the Boot Loader icon on the right.
3. In the Boot Loader Configuration window, click the Boot Loader Installation tab, and then click the Boot Loader Options button.

4. In the Boot Loader Options window, select a setting, in seconds, in the Boot Menu Time-Out box, as shown in Figure 31-2. To prevent the boot menu from appearing, either enter a value of 0 for the time-out or remove the check from the box marked Show Boot Menu.

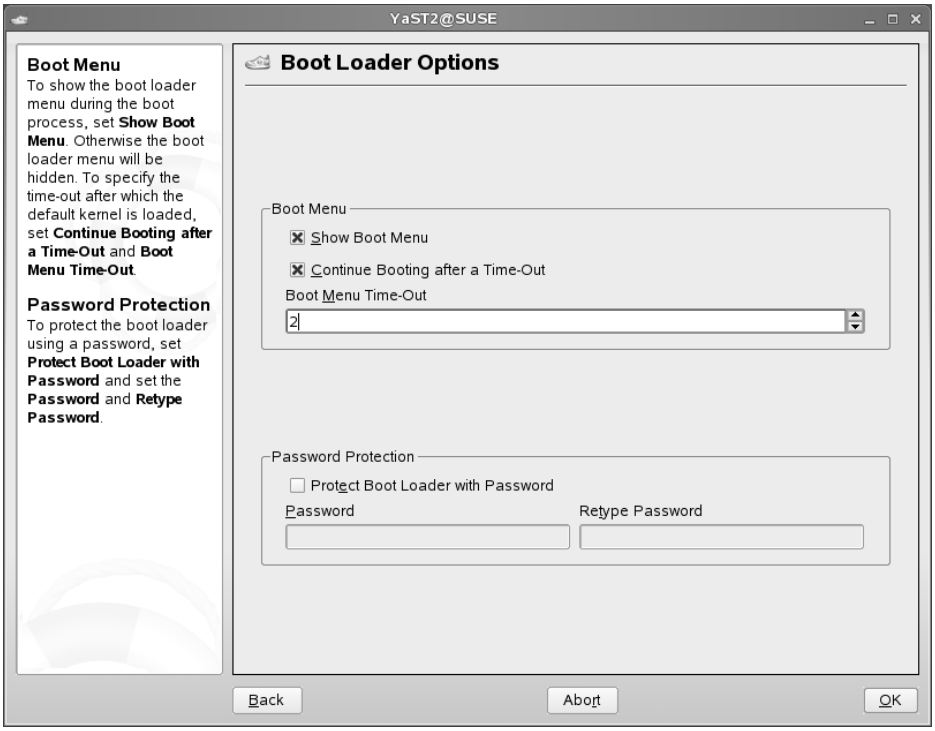

Figure 31-2. *You can stop the GRUB menu hanging around for so long by changing the time-out value in its configuration file.*

5. Click the OK button, and then click Finish to write the changes to disk.

Optimizing Run-Level Services

Perhaps it goes without saying that the majority of bootup time is spent starting the boot and run-level scripts. This is when the entire system comes to life—hardware and essential software services are activated. But this isn't to say that all run-level scripts are essential.

■ **Note** A *service* is a piece of background software that provides something that you, the user, need on a day-to-day basis. Some services manage hardware, such as the graphical interface, printing services, and networking. Some services provide software services, such as logging files or checking the system clock against a time server.

The one-size-fits-all approach of SUSE Linux means that some services that are started up aren't always necessary. Approximately 60 run-level scripts start on a typical boot. By selective pruning, you can easily remove around a quarter of these, but caution is advised. You're altering a fundamental aspect of your system configuration, and one simple mistake can make the difference between a system that works and one that is no longer able to boot.

Creating a Custom Run Level

Because of the risk of seriously damaging your system by pruning run-level scripts, I recommend that you use run level 4 for your experiments. Run level 4 is officially designated as "unused" and is therefore ideal for the purpose.

For this plan to work, you need to make run level 4 into a clone of run level 5, and then configure the computer to boot to run level 4, rather than 5. Here's how to proceed:

1. Open a terminal window. If you're running the GNOME desktop, click Applications ➤ System ➤ Terminal ➤ GNOME Terminal. KDE users should click K menu ➤ System ➤ Terminal ➤ Konsole.

2. Switch to the root user at the command line by typing `su -`.

3. Clear out the existing run level 4 scripts using the following command:

   ```
   rm /etc/init.d/rc4.d/*
   ```

4. Copy the scripts for run level 5 to the directory of run level 4, thereby creating a clone. Because the run level 5 directory contains symbolic links to files in `/etc/init.d`, you need to use the `-P` command-line option with the `cp` command. That way, it copies the links, rather than the linked files. Issue the following command:

   ```
   cp -P /etc/init.d/rc5.d/* /etc/init.d/rc4.d/
   ```

5. The default run level that SUSE Linux boots into is listed in `/etc/inittab`. You need to edit this file to switch the default to run level 4 and also to enable run level 4 as a usable run level within the file, because ordinarily it is disabled.

   - Under the GNOME desktop, type the following to load `/etc/inittab` into the Gedit text editor:

     ```
     gedit /etc/inittab
     ```

   - Under KDE, type the following to load the file into the Kate text editor:

     ```
     kate /etc/inittab
     ```

6. In the text editor, look for the line that reads `id:5:initdefault:`. Change the `5` to a `4`, so that the line now reads as follows:

   ```
   id:4:initdefault:
   ```

7. Scroll down the file to the line that reads as follows:

   ```
   #l4:4:wait:/etc/init.d/rc 4
   ```

8. Remove the hash at the beginning of the line, so that it reads like this:

   ```
   l4:4:wait:/etc/init.d/rc 4
   ```

9. Save the file and quit the text editor.
10. Reboot your computer.

■ **Tip** While rebooting, pressing the Esc key will clear the boot graphic and show the status messages of the various run-level scripts. You can verify that you're entering the newly defined run level 4 by watching for a message along the lines of "Entering run level 4." This will appear around halfway through the boot procedure. When up and running, typing `runlevel` as the root user will display the current run level (two numbers will be quoted—the second indicates the current run level).

Pruning Run-Level Services

YaST includes a tool that allows you to control which run-level services start during bootup. To access it, start YaST (click Desktop ➤ YaST under GNOME, or K menu ➤ System ➤ YaST under KDE), click the System icon on the left, and then click the System Services (Runlevel) icon on the right. When the System Services (Runlevel) window appears, click the Expert Mode button. You should see a window similar to that shown in Figure 31-3.

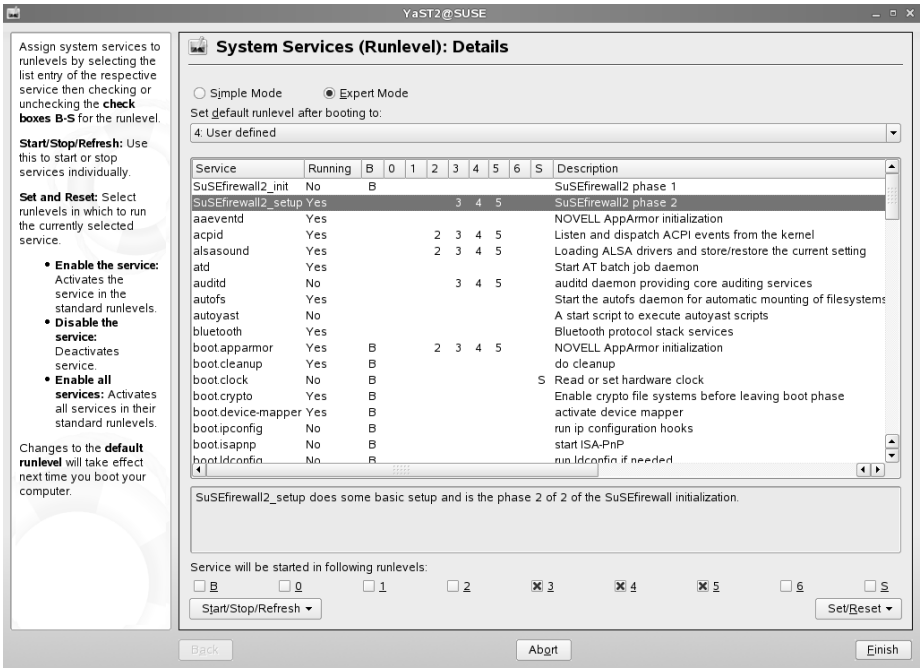

Figure 31-3. *The System Services (Runlevel): Details window shows each available service, whether the service is running, the run levels the service is normally active within, and a description.*

The Expert Mode listing might look confusing at first glance, but it is quite easy to understand. The columns contain the following information:

Services: On the left are listed the various services on the system. This list includes all available services, even if they're not active under any run level and are not used during boot (effectively, these are the services contained in the `/etc/init.d/` directory).

Running: This column lists whether the service is running at the present time. This will be indicated by Yes or No in the list. It's important to note that not all services keep running after they're started. Some start during boot and terminate almost immediately, or after a short while. Therefore Yes or No in this column isn't a good indicator of whether the service is set to run in the current run level.

■ **Note** You may wonder why some services are listed as currently running when they're not enabled in any run level. This is probably because they were started as a subprocess by another service that *is* activated within the current run level. Don't worry too much about these services.

B, 0, 1, 2, 3, 4, 5, 6, and S: These columns indicate within which run levels the services are activated. (See Table 31-1, shown earlier in the chapter, for a description of the purpose of each run level.) The B entry refers to services started by the `boot` script, which precedes the run-level scripts, as mentioned previously. You are interested solely in services that start in run level 4, which is the run level you have created especially to be pruned of unnecessary services.

Description: This column shows brief descriptions of what various run-level services do.

Beneath the list is a text box in which a longer description may appear when a run level is selected.

Take a look through the list and see what the various services do. This should give you an idea of which you might consider removing. In Table 31-2, I've listed some run-level services that I consider it safe to disable. Some of these may not be present on GNOME systems because they are KDE-specific. By removing these services, I managed to trim 10 seconds from the total boot time of my test PC.

You should follow some common-sense rules when removing entries:

- Don't remove a service whose purpose you don't understand. If in doubt, leave it be! Bear in mind that several services might look innocuous but may be essential because they support other, more essential system components.
- Disable services only for run level 4. Don't disable the service for any other run level, especially run level 5, which you will want to use later if your changes prove problematic.
- Avoid disabling any B run-level services. These configure and start various hardware items.

Table 31-2. *Services That You May Want to Disable*

Service	Description	Notes
`cups`	`cups` printing service component	Can be disabled if you do not have a printer attached to your computer and do not intend to print from the computer in the future.
`cupsrenice`	See `cups`	See `cups`.
`fbset`	Component of the graphic boot-time display	Can be disabled.[1]
`nfs`	Configures Network File Sharing (NFS)	Can be disabled if you don't access NFS shares (only corporate/advanced users are likely to do so).
`nfsboot`	See `nfs`	See `nfs`.
`nfsserver`	See `nfs`	See `nfs`.
`nmb`	Part of the SMB file sharing system	Can be disabled if you don't intend to share files or access shared files on other computers (including Windows computers).
`postfix`	Mail Transfer Agent (MTA)	Can be disabled if you use an external SMTP server, as most people do.
`smb`	See `nmb`	See `nmb`.
`smbfs`	See `nmb`	See `nmb`.
`splash`	Component of the graphic boot-time display	Can be disabled.[1]
`splash_early`	See `splash`	See `splash`.
`sshd`	SSH service	Can be disabled if you don't intend to remotely connect to your computer (see Chapter 34).

[1] *To fully disable the boot-time graphic, start YaST, click the System icon on the left, and then click the Boot Loader icon on the right. In the window that appears, select the SUSE Linux entry in the list and click the Edit button. Change the Other Kernel Parameters box entry from* `splash=silent` *to* `splash=0`. *Click OK, and then click Finish.*

Deactivating a service is easy. At the bottom of the YaST window you will see several check boxes corresponding to the various run levels. Simply remove the check in the box marked 4 to stop the service from starting in your custom run level.

When you've finished, click the Finish button. This will write the changes to disk. You should then reboot to test your new settings.

Restoring Run Level 5

If you are too aggressive when pruning run-level services, you may find that you've left your system in an unusable state. If this happens, you can simply switch the default run level back to 5, and then attempt to make repairs to your custom run level.

The following instructions assume a worst-case scenario, which is to say that you're unable to boot to a GUI, but they work under other circumstances as well.

1. Reboot the computer. At the GRUB boot menu, select Failsafe – SUSE Linux 10.1.
2. At the login prompt, log in as the root user.
3. Type the following to open the `/etc/inittab` file in the `pico` text editor:

   ```
   pico /etc/inittab
   ```

4. Scroll down to the line that reads `id:4:initdefault:`. Change the `4` to a `5`, so the line now reads `id:5:initdefault:`.
5. Save the file by pressing Ctrl+O (the letter *O*). Press Enter to confirm the filename.
6. Press Ctrl+X to exit the `pico` text editor.
7. Type `reboot` at the command line to restart your computer.

STOP SEARCHING FOR AN ADDRESS

If you use an Ethernet or Wi-Fi connection to access the network, you'll find that SUSE Linux spends a few seconds each boot acquiring an Internet address. Therefore, one way to provide an instant speed boost is to give your computer a static IP address. (See Chapter 8 for details on how to configure your network interface.)

To assign a static address, you'll need to find out what IP address range your router uses. You can discover this address by looking at the router's configuration software (sometimes this is accessed via a web browser). Look for the configuration section with a heading like DHCP Configuration.

Normally, the addresses are in the 192.168.*x.x* range, where *x.x* can be any series of numbers from 1.1 to 255.255. For instance, the router within my test setup uses the 192.168.1.2-255 range.

In my case, choosing a static IP address that will work with the router is simply a matter of selecting an IP address in this range. However, I know that the router hands out addresses sequentially from 2 upward, so it's best if I choose an address it's unlikely to reach, even if I happen to have many computers connected to the network. Starting at 50 is a good idea, so I assign my test PC the address 192.168.1.50.

Don't forget that, when defining static IP addresses, you'll need to manually supply the gateway, subnet, and DNS addresses. In most cases, the subnet address will be 255.255.255.0. The DNS address will be the same as the gateway address.

Optimizing Hard Disk Settings

The hard disk is one of the key elements in the modern PC. Because most of your PC's data must travel to and from it, speeding up your hard disk means that your entire PC will be faster.

SUSE Linux provides a powerful command-line tool that you can use to control every aspect of your hard disk: `hdparm`. This is a power-user's tool. Not only must it be run as the root user, but you also must be careful not to mistype the commands. All changes are made instantly, so if you make a mistake, your system may crash, or at least suffer from serious

problems. There's even the risk of data loss, although this is minimized by making sure that you have no other programs running at the same time you run `hdparm`.

■ **Tip** When switching to the root user to carry out the steps in this chapter, be sure to use the `su -` command, rather than simply `su`. Adding the dash will switch you to root user and also inherit root's path, so that you can use system configuration commands like `hdparm` that are normally contained in `/sbin`.

The good news is that changes made via `hdparm` will last for only the current session, so there's no risk of permanent damage. Any changes that are beneficial can be made permanent later.

In the context of optimization, `hdparm` lets you both benchmark the disk and change various technical settings, such as the sector `multcount` value. These adjustments can bring speed boosts.

Benchmarking Your Hard Disk

Because experimenting with `hdparm` can cause crashes, and because its benchmarking feature needs almost exclusive access to the hard disk, `hdparm` is best run with as few as possible additional programs up and running. Therefore, switching to run level 3 is a good idea. To do this, close all open programs, then open a terminal window (in GNOME, click Applications ➤ System ➤ Terminal ➤ GNOME Terminal; in KDE, click K menu ➤ System ➤ Terminal ➤ Konsole).

At the prompt, type the following to switch to run level 3:

```
su -
[Enter root password]
init 3
```

■ **Note** Technically speaking, switching to run level 1 is an even better idea, because this will deactivate all unnecessary services. Run level 1 is akin to the Windows Safe Mode, except without the GUI. However, you want realistic benchmark results to test the changes you make via `hdparm`, and it's debatable whether the restricted confines of run level 1 will provide such results.

Let's start by benchmarking your hard disk to see its performance based on the current settings. Type the following (assuming SUSE Linux is installed on the first hard disk in your system; if it's on the second hard disk, change `/dev/hda` to `/dev/hdb`):

```
hdparm -tT /dev/hda
```

This will benchmark your disk in two ways. The first tests the PC's memory throughput, measuring the data rate of the memory, CPU, and cache. The second actually tests the disk's

data rate. The second test affects the outcome of the first, which is why the two are used together. Between them, these two methods of benchmarking present the standard way your disk is used on a day-to-day basis. Figure 31-4 shows the results on my system.

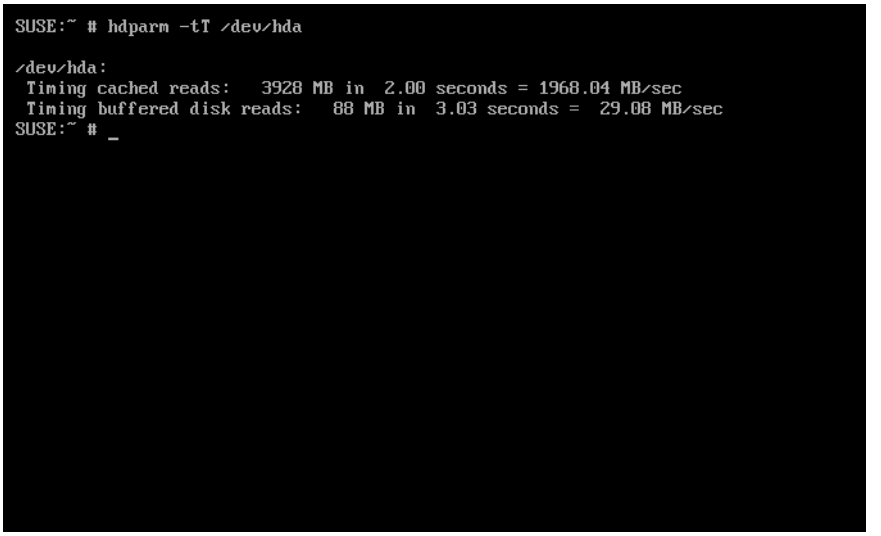

Figure 31-4. *The hdparm program can be used to both benchmark and optimize your hard disk.*

Make a note of the figures so that you can compare them to the results of these tests after you change hard disk settings.

Changing Hard Disk Settings

You can use `hdparm` to view your current hard disk settings by entering the following at the command prompt:

```
hdparm /dev/hda
```

On my test PC, these are the results I got:

```
/dev/hda:
 multcount    = 16 (on)
 IO_support   =  1 (32-bit)
 unmaskirq    =  1 (on)
 using_dma    =  1 (on)
 keepsettings =  0 (off)
 readonly     =  0 (off)
 readahead    = 1024 (on)
 geometry     = 65535/16/63, sectors = 78125000, start = 0
```

Let's take a look at what these settings mean.

The multcount Setting

The first setting, `multcount`, refers to how many sectors can be read from the hard disk at any one time. On many drives, a higher setting here is best, but on other drives, a lower setting is best. You can find out what your drive's maximum `multcount` setting is by issuing the following command:

```
hdparm -i /dev/hda
```

Look for `MaxMultSect` in the results. On my test PC, this read `MaxMultSect=16`.

You can experiment with the `multcount` setting on your hard disk by using the `-m` `hdparm` command option:

```
hdparm -m8 /dev/hda
```

Here, I've chosen a lower value than my drive's original setting (`16`).

You can then follow this by another benchmark to see if there is an improvement:

```
hdparm -tT /dev/hda
```

The IO_Support Setting

The `IO_support` setting refers to the input/output (I/O) mode used by the hard disk controller. This has three possible settings: `0` to disable 32-bit support, `1` to enable 32-bit support, and `3` to enable 32-bit support with a special sync signal.

You can change the `IO_support` setting with the `-c` `hdparm` command option, and the 32-bit support with sync option (`3`) is generally considered the best choice:

```
hdparm -c3 /dev/hda
```

The unmaskirq Setting

The third setting, `unmaskirq`, allows SUSE Linux to attend to other tasks while waiting for your hard disk to return data. This won't affect hard disk performance very much, and generally it's a good idea for the health of your system to activate it if isn't already switched on. This command activates `unmaskirq`:

```
hdparm -u1 /dev/hda
```

The using_dma Setting

The fourth setting refers to whether Direct Memory Access (DMA) is in use. Hard disks are sold on the basis of their DMA modes, such as UltraDMA Burst 2 and the like. DMA is considered an indicator of the speed of a hard disk, but the truth is that, like any specification, it is only a guide.

DMA is activated by default under SUSE Linux, but you can alter the DMA mode using the `-X` command option. However, on most modern PCs, this isn't necessary because the computer's BIOS defaults to the fastest DMA mode.

Other Settings

The last three settings, above the summary of the geometry and sector information of the disk, are those you shouldn't change:

- `keepsettings` refers to the ability of the drive to remember `hdparm` settings over a reboot (although not if the drive loses power).
- `readonly` sets whether or not the hard disk is read-only (so that no data can be written to it). Changing this setting is not advisable!
- `readahead` controls how many hard disk blocks are loaded in advance. It doesn't affect the performance of modern IDE-based hard disks, because the drive electronics contain buffers that perform this task themselves.

Making Disk Optimizations Permanent

When the PC is switched off, any changes you've made with `hdparm` are lost. To make the command run during bootup, you can create a custom service and add it to the current run level. This sounds more complicated than it actually is. Here are the steps:

1. Switch to the root user, using the `su -` command.
2. Enter the `hdparm` settings into a text file, using `vi`, for example. Simply type the command (for example, `hdparm -d1 -m8 /dev/hda`) in the editor window, and then save the file with a name along the lines of `disk_optimize`. (Make sure you add a carriage return after the line when entering it in `vi`.)
3. Make the text file an executable file, with the following command, which effectively turns the file into a run-level script:

   ```
   chmod +x disk_optimize
   ```

4. Save the file with all the other run-level scripts in the `/etc/init.d/` folder:

   ```
   cp disk_optimize /etc/init.d/
   ```

5. Symbolically link the script to the current run level, using the following command, which assumes the default run level is 5:

   ```
   ln -s /etc/init.d/disk_optimize /etc/init.d/rc5.d/S99disk_optimize
   ```

The last step also changes the filename of the symbolically linked file to tell SUSE Linux when it should be run during bootup. A value of `S99` means it will be run at the very end of the boot procedure. Any hard disk performance tweaking must take place as near as possible to the end of booting, because the changes it makes can negatively affect the startup of other services.

Prelinking

As discussed in Chapter 29, a lot of SUSE Linux software relies on other pieces of code to work. These are sometimes referred to as *libraries*, a term that is a good indicator of their purpose: to provide functions that programs can check in and out whenever they need them, as if they were borrowing books from a library.

Whenever a program starts, it must look for these other libraries and load them into memory so they're ready for use. This can take some time, particularly on larger and more-complicated programs. Because of this, the concept of *prelinking* was invented. By a series of complicated tricks, the prelink program makes each bit of software you might run aware of the libraries it needs, so that memory can be better allocated.

Prelinking claims to boost program startup times by up to 50% or more, but the problem is that it's a hack—a programming trick designed to make your system work in a nonstandard way. Because of this, some programs are incompatible with prelinking. In fact, some might simply refuse to work unless prelinking is deactivated. At the time of writing, such programs are in the minority. However, keep in mind that prelinking can be easily reversed if necessary. Alternatively, you might want to weigh whether it's actually worth setting up prelinking in the first place.

■ **Note** Many of the GNOME programs under SUSE Linux aren't compiled in a way that's compatible with prelinking. Therefore, you might not see much of a speed boost using prelinking with the GNOME desktop.

Using Prelinking

If you decide to go ahead with prelinking, you'll need to download and install the relevant software because it isn't supplied on the installation DVD.

Installing the Prelink Software

You can download the `prelink` package from the "factory installation source," an online repository of all official SUSE Linux packages. Here's the procedure:

1. Using your web browser, head over to `http://download.opensuse.org/` and click the link for the Internet Installation Repository. This should then bring up what appears to be a file system view of files and folders.
2. Click the `suse` folder, which should be near the bottom of the list. Then click the `i586` link.
3. Give the file list a minute or two to load, and then search for a file that begins with `prelink`. During my testing, the file `prelink-0.3.6-7.i586.rpm` was the one I needed. While you're there, also download the file whose name begins with `libelf`, which is a dependency of `prelink`. During my testing, this file was `libelf-0.8.5-45.i586.rpm`.
4. Right-click each of the files and select Save Link As. Then save the files to the desktop.

5. Open a terminal window (in GNOME, click Applications ➤ System ➤ Terminal ➤ GNOME Terminal; in KDE, click K menu ➤ System ➤ Terminal ➤ Konsole) and switch to the root user (type `su -`).

6. Switch to the directory where you downloaded the files, and then type the following (assuming the directory does not contain any other installation RPM files):

   ```
   rpm –Uvh *.rpm
   ```

Configuring and Running Prelink

Prelink will run automatically in the background periodically, but first you must activate it as a service. Then it's a good idea to run prelink manually for the first time. Follow these steps:

1. Open a terminal window and switch to the root user (type `su -`).

2. Navigate to the `/etc/sysconfig/` directory and open the `prelink` file in a text editor.

3. Change the line that reads:

   ```
   USE_PRELINK="no"
   ```

   to

   ```
   USE_PRELINK="yes"
   ```

4. Run the SuSEconfig program by typing `SuSEconfig` in the terminal window. This ensures that system configuration files are up-to-date, including those related to prelinking that have been added.

5. To run a prelink scan of your system whenever you want, issue this command (as the root user; type `su -`):

   ```
   prelink -a
   ```

The `prelink -a` command will prelink practically all the binary files on your system and may take some time to complete. You may also see some error output, but you don't need to pay attention to it.

Deactivating Prelinking

Should you find prelinking makes a particular application malfunction or simply stop working, you can try undoing prelinking. To do this, find out where the main binary for the program resides, and then issue the `prelink` command with the `--undo` command option. For example, to remove prelinking from the Gedit text editor program, you could type the following (as the root user; type `su -`):

```
whereis gedit
prelink --undo /usr/bin/gedit
```

However, this may not work because some programs might rely on additional binaries on the system. Therefore, the solution might be to undo prelinking for the entire system, which you can do by typing the following (as the root user):

```
prelink -ua
```

After this, you should remove the `prelink` package to prevent it from running again in future. To do this, type the following as the root user:

```
rpm –e prelink
```

OPTIMIZING THE KERNEL

Using the Linux kernel source code, you can compile and install your own version of the program at the heart of Linux. This gives you total control over the kernel configuration, so you can leave out parts you don't want in order to free memory. You can also set certain optimization settings, such as creating a version of the kernel specifically built for your model of CPU.

Although compiling a kernel is a simple procedure, there are many complex questions that you'll need to answer, and an in-depth knowledge of the way Linux works is necessary.

In addition, compiling your own kernel brings up several issues. The first is that it may not work with any binary modules that you have installed, such as graphics cards or wireless drivers (including `ndiswrapper`, as discussed in Chapter 8). You can opt to install these yourself from scratch, but this adds to the complexity.

The second problem is that SUSE Linux is built around precompiled kernels. Several software packages expect to work with the precompiled kernel and, in addition, SUSE Linux may occasionally download an updated prepackaged kernel automatically as part of the system update feature and override the one you've created.

If there are any security problems with the kernel version you compiled, you'll need to recompile a new kernel from scratch (or patch the one you have). This means you'll need to keep an eye on the security news sites and take action when necessary.

That said, compiling a kernel is an excellent way of learning how Linux works, and the sense of achievement if it all goes well is enormous.

Some people choose to download the kernel source code from the official Linux kernel site, `www.kernel.org`. However, it makes more sense to download the official SUSE Linux release, because this will be tailored for the way your system works. Using the Software Management tool of YaST (see Chapter 29 for details), simply search for and install `kernel-source`.

You can find several guides to compiling your own kernel online. I recommend `www.digitalhermit.com/linux/Kernel-Build-HOWTO.html`.

Freeing Disk Space

After using SUSE Linux for some time, you might find that the disk begins to get full. You can keep an eye on disk usage by using the following command in a terminal window (in GNOME, click Applications ➤ System ➤ Terminal ➤ GNOME Terminal; in KDE, click K menu ➤ System ➤ Terminal ➤ Konsole):

```
df -h
```

This will show the free space in terms of megabytes or gigabytes, and also expressed as a percentage.

If the disk does start to get full, you can take some steps to make more space available.

Emptying the /tmp Folder

An easy way to regain disk space is to empty the `/tmp` folder. As with the Windows operating system, this is the folder in which temporary data is stored. Some applications clean up after themselves, but others don't, leaving behind many megabytes of detritus.

Because the `/tmp` folder is accessed practically every second the system is up and running, to empty it safely, it's necessary to switch to run level 1. This ensures few other programs are running and avoids the risk of deleting data that is in use. The following series of commands will switch to run level 1, empty the `/tmp` folder, and then reboot afterwards (this will stop the GUI, so make sure you close all running programs beforehand):

```
su -
[Enter root password]
init 1
[Enter root password again]
rm -rf /tmp/*
rm -rf /tmp/.*
reboot
```

■ **Tip** On a similar theme, don't forget to empty the desktop Trash. This can hold many megabytes of old data. If you find permission errors are reported when emptying this folder, open a command-line prompt and type the following: `sudo rm -rf .Trash/*`. You'll need to type your root password.

Removing Unused Software

If you still need disk space, consider removing unused programs. As you learned in Chapter 29, you can manage software through YaST's Software Management tool (start YaST, click the Software icon, and then click the Software Management icon).

The best way of managing software is to switch to the Selection or Package Group view, by selecting the corresponding filter from the Filter drop-down list. The Selection filter lets you see which groups of programs you have installed, rather than viewing individual titles (removing individual programs might not free up too much space). The Package Group filter categorizes titles by what they do, as shown in Figure 31-5. For example, you might choose to remove software under the Games heading. Alternatively, if you don't use any office programs on your system, you might choose to deselect titles under the Productivity group heading.

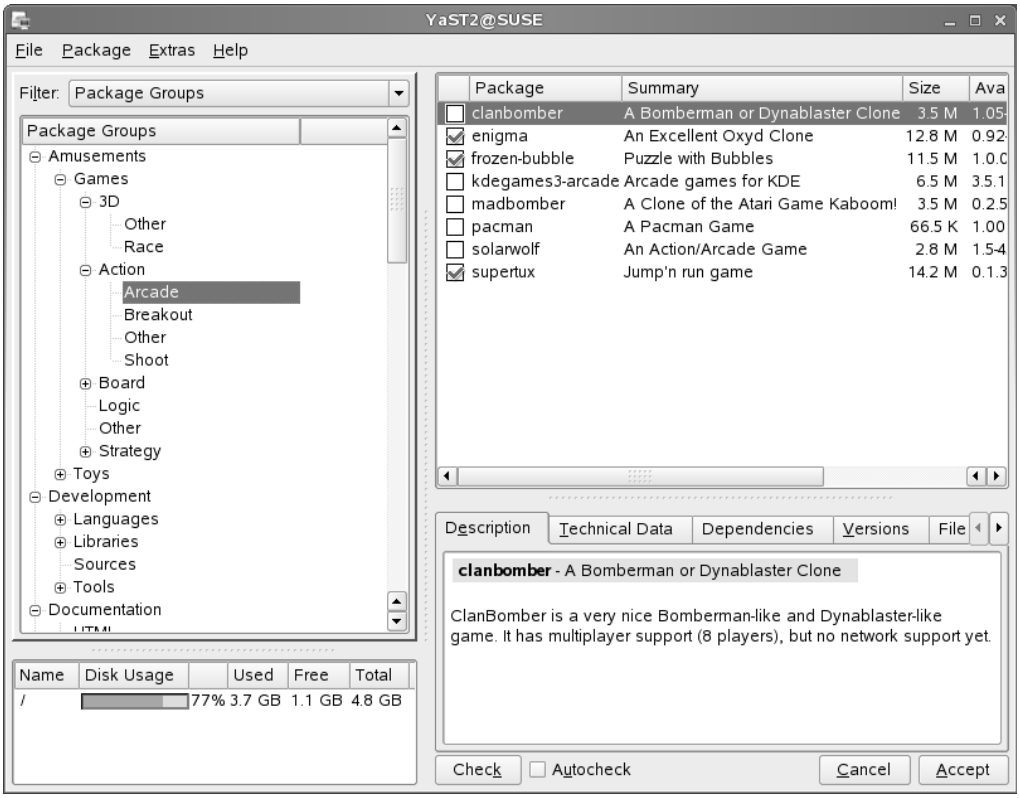

Figure 31-5. *Using the Package Group filter helps identify classes of programs that you might want to remove to free space, such as games.*

Caution As always, removing software can create dependency problems, so you might find yourself limited in what software you can actually remove.

Adding Another Disk Drive

Another solution to the problem of running out of disk space is to add another hard disk drive, perhaps as a slave on the primary IDE channel. Using a new disk within SUSE Linux is very easy and can be done from the command-line shell.

Partitioning the Disk

Once the disk has been fitted, follow these steps to partition the disk:

1. Boot into SUSE Linux and open a terminal window (in GNOME, click Applications ➤ System ➤ Terminal ➤ GNOME Terminal; in KDE, click K menu ➤ System ➤ Terminal ➤ Konsole).
2. Use the `cfdisk` command to initially partition the disk. Assuming that you've added the new disk as a slave on the primary channel, issue the following command as the root user (type `su -`):

   ```
   cfdisk /dev/hdb
   ```

■ **Tip** Working out how SUSE Linux refers to the hard disks installed on the system isn't hard. Usually, they're given letters from `a` through to `d`. So, `/dev/hda` is the primary master, `/dev/hdb` is the primary slave, `/dev/hdc` is the secondary master (usually the CD/DVD-ROM drive), and `/dev/hdd` is the secondary slave. If your system uses SCSI drives, you'll find they're named `/dev/sda`, `/dev/sdb`, and so on.

---

3. To create a new partition, within the `cfdisk` program, use the cursor keys to highlight New, and then press Enter. The default partition size should automatically be all of the disk space, so press Enter again to confirm this.
4. With the new partition created, highlight Write on the menu and press Enter. This will write the new partition information.
5. Highlight Quit and press Enter.
6. Reboot the system to ensure the new partition is made available.
7. To make your partition accessible, when SUSE Linux is back up and running, open a terminal window and issue the following command as root user (again assuming that the new hard disk is `/dev/hdb`):

   ```
   mkfs -t reiserfs /dev/hdb1
   ```

   Note that you need to specify the partition number in this instance. Because there's only one partition on the disk, this is number 1. If you create two or more partitions, each will be numbered consecutively (1, 2, 3, and so on).

You've created a reiserfs-formatted partition, because this is the preferred standard used within SUSE Linux. Other versions of Linux might use different file systems, such as ext3.

Configuring SUSE Linux to Use the Drive

Now the new drive is ready for use, but you need a way of making it available within the SUSE Linux file system. Therefore, you need to create a mount point and also configure the system so that the disk is mounted automatically at boot.

As discussed in Chapter 14, creating a mount point is simply a matter of creating an empty folder. Therefore, you can create a directory in the root of the hard disk (or anywhere else) and call it something like `second_disk`. This directory must then be made writable, as follows (all commands should be entered as the root user):

```
mkdir /second_disk
chmod a+w /second_disk
```

Then you must edit the `/etc/fstab` file in order to make the new disk mount automatically. All you need to do is add a line at the end of the file, such as this:

```
/dev/hdb1 /second_disk reiserfs default 0 2
```

Note that it's important that you add a carriage return (press Enter) after the line.

You can test your new hard disk by rebooting. When SUSE Linux returns, you should find that the new disk is available by accessing the `/second_disk` directory. You can check its capacity by typing `df -h`.

Summary

In this chapter, we looked at streamlining your installation of SUSE Linux. This involved speeding up the boot procedure by decreasing the boot menu delay and deactivating various unnecessary run-level scripts that get loaded at boot time. We also looked at optimizing your hard disk settings to allow for greater efficiency in loading and saving files.

Additionally, we investigated prelinking programs so that they load faster, recompiling the kernel so that it's optimized for your system, freeing disk space by various means, and adding a second hard disk.

In the next chapter, you'll learn how to perform backups to safeguard your data.