Multimedia applications and the realtime priority

GETTING REAL

Linux provides tools and patches for speeding up the priority of multimedia applications. So if you're not getting the performance you expect, try shifting into overdrive. **BY OLIVER FROMMEL**

Despite the presence of multiple Gigahertz clock speeds and multiple Gigabytes of memory, TV on the computer screen sometimes doesn't work as smoothly as you might like. In some cases, the slow performance of a multimedia application may result from the priority assigned to the application.

Actually, one of the biggest advantages of Linux often causes problems when it comes to prioritizing multimedia. Because Linux is designed as a multi-user operating system, the emphasis is on keeping many different applications running concurrently. By default, Linux does not assign the full power of the system to a single program. Today's Linux computers, however, often belong to a single user, and that user can sometimes benefit from dialing up the priority for resource-hungry multimedia applications.

Modern Linux kernels allow users to assign higher priorities to individual applications, giving them preferential treatment when it comes to CPU cycles. There are many different solutions for revving up the application priority. If you have the latest version of your preferred distribution, you may find that many of the features are enabled in the default kernel (such as the Rlimits feature, discussed later in this article, which is enabled in kernel version 2.6.14 or newer).

If you are thinking about experimenting with re-assigning system priority levels, you probably want to enable kernel preemption when compiling the kernel. You can check the kernel configuration file, typically stored in the */boot/grub* directory, to see if your Linux distribution adds kernel preemption by default. The file for Ubuntu is titled *config-2.6.12-9-k7*, for example. Use Grep to search *PREEMPT*. If you discover that *CONFIG_PREEMPT* is not set or *PREEMPT_NONE*

Ford Motor Company

is set, you can recompile your kernel with modified settings (Figure 1). However, *CONFIG_PREEMPT_VOLUNTARY = y* is acceptable, and *PREEMPT* is better still.

If you really want to crank up the priority of a Linux multimedia application, the realtime priority level is the ultimate destination. Realtime priority gives the application almost complete control of system resources, which could have the effect of bringing other applications to a standstill.

If your kernel supports realtime priority, you still need to find out how to enable this feature. Normally, only root is allowed to execute a program with realtime priority. After all, you don't want to give every single user the ability to bring the computer to its knees by running his or her software with realtime priority. However, running end-user multimedia applications as root doesn't make much sense either. For a responsible single user, therefore, the best solution is to set up Linux so that it will support realtime speeds without requiring root privileges. This article discusses some techniques for running your applications in realtime without root access.

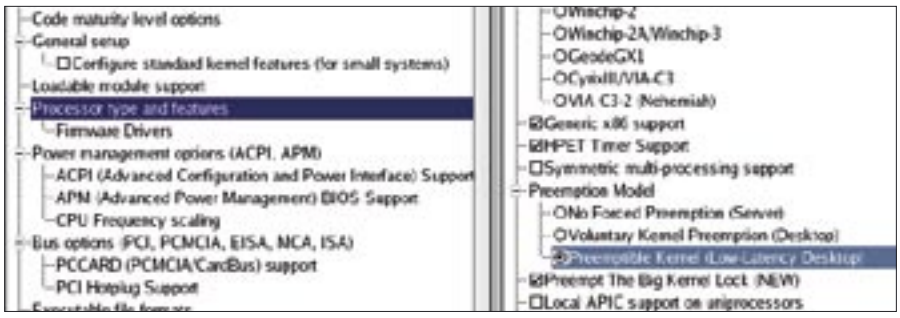

Figure 1: If you are looking for a system with quick response, you will want to enable the "CONFIG_PREEMPT" setting when you build the kernel.

Realtime without Root

Serious audio users have long since devised an approach to the realtime requirements of the Jack [2] audio server: the realtime-lsm [3] kernel module. The kernel developers have not added the module to the standard kernel due to various misgivings; but it has proved its value in practical audio applications. What the module aims to do is give realtime priority to programs not running with root privileges. This is important for Jack, as it has to run under the same user ID as the audio programs that access it – and it is not a good idea to run every single synthesizer program as root.

Realtime LSM

Ubuntu has the realtime-lsm kernel module in its repository, and users of other distributions can download the source code from the website. To use realtime-lsm, you need Capabilities as a module, not built-in to the kernel. This is the case with Suse 9.3, but not with Fedora Core 4, so Fedora users will need to recompile the kernel and set up Capabilities as a module (Figure 2). Users with other distributions will require the kernel sources, or at least the headers, depending on their choice of distribution.

After satisfying the requirements, go on to compile and install realtime-LSM by running *make* and *make install*. If you then additionally run *depmod -a*, you can enter *modprobe realtime-lsm* to load the module. realtime-LSM understands a few parameters that you can set to influence the module's behavior. For example, *any = 1* tells the module to assign realtime priority to any requesting program. For more control, the gid parameter lets you specify the group ID of authorized programs. For example, if your audio programs belong to the *audio* with a GID of *33*, you can load the kernel module as follows: *modprobe realtime-lsm gid = 33*. This lets you run Jackd in realtime mode with the *-R* option without root privileges.

Rlimits

Instead of the workaround I just described, most kernel developers prefer to use the Rlimit mechanism to assign realtime priority, and this is why Rlimit became part of standard kernel 2.6.14. The R in Rlimits does not stand for realtime, however, but for resource. It lets you specify the use of various system resources, such as memory usage, and, more interestingly for this article, the priority.

The approach to Rlimits control that most developers would like to see involves the PAM (Pluggable Authentication Module) subsystem, which handles password management in many distributions (Fedora, Suse, …). Unfortunately, only the latest (0.79 or newer) and patched PAM versions actually support Rlimits. If you happen to have a PAM that is up to the job, you can specify the users and/or groups allowed to assign realtime priority in the */etc/security/limits.conf* file. Each entry starts with the group, followed by the resource, and then by the value. The second column typically specifies whether the user can modify the preset value (*soft* if so, *hard* if not). Resources include realtime priority *rt_priority*, the nice level *nice*, and the amount of memory to lock *memlock*.

```
@audio hard rt_priority 80
@audio hard nice -10
```

If you prefer not to update the whole PAM subsystem on your current distribution, you can try a special utility that only handles realtime priority: the aptly named *set_rtlimits* tool [4]. To avoid any user assigning priorities at will, the administrator can use the */etc/set_rtlimits.conf* configuration file to specify which users and groups will have this ability. Additionally, the file actually lists the programs permitted to assume realtime priority. The following line lets a user called *joey* give the Jack server */usr/bin/jackd* realtime priority:

```
joey /usr/bin/jackd -1 80
```

The next number in this entry sets the maximum nice value (see the manpage for *nice*), the second number sets the maximum realtime priority. A negative number means that the user is not permitted to change the assigned value. The program detects authorized user groups based on a leading @, for example *@audio*. You can then run the required program by passing the absolute pathname to the command line tool:

```
set_rlimits -r=80 ➥
/usr/bin/mythfrontend
```

As Rlimits is now part of the standard

kernel, the next versions of most Linux distributions will support it. Let's hope this leads to less complex tools that let users prioritize applications in a simpler way.

Con Kolivas' Patchset

Not long ago, an Australian doctor by the name of Con Kolivas caused a stir on the Linux kernel mailing list, by learning C programming in next to no time, then moving on to kernel hacking, and finally by optimizing the kernel scheduler. The scheduler is the component that assigns CPU time to active programs. The less time the scheduler needs to do this, the quicker programs can run, quasi-simultaneously. Con Kolivas scheduler is now part of the standard kernel, and Con has a number of other performance boosting patches on his homepage.

Among other things, the patch set includes a number of new priority classes for realtime, for example, *SCHED_ISO* and *SCHED_BATCH*. Batch priority is useful for processes that take a long time to complete on server machines. This class allows server programs to run awhile longer, before the scheduler interrupts them. This in turn means less time lost on task switching. On the downside, the server will not respond as quickly to user input, but this is probably something you can live with on a server machine.

Realtime Lite

The *SCHED_ISO* priority class is something like a "lite" version of the realtime priority. It does not require root privileges, but neither does it assign the highest priority. If a user without root privileges attempts to assign realtime priority to an application, the Kolivas scheduler automatically puts the program in the *SCHED_ISO* class.

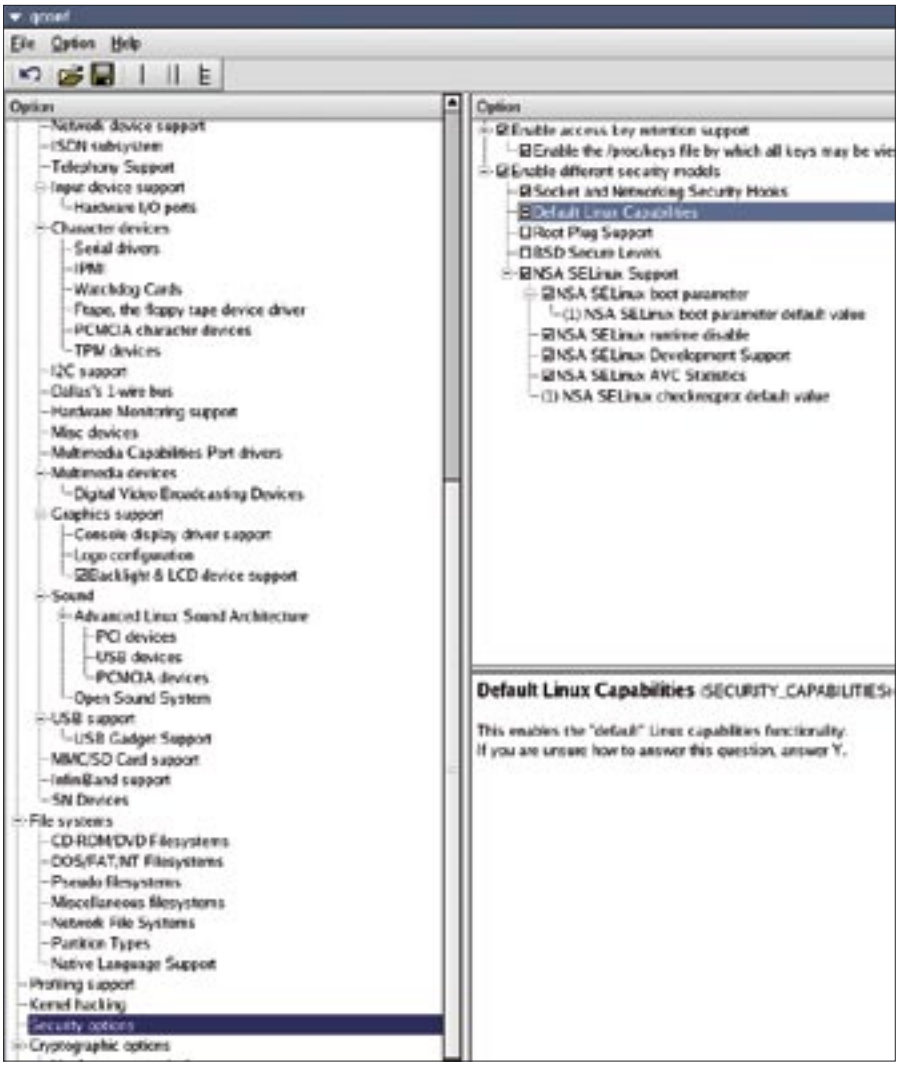

Figure 2: You need Capabilities compiled as a module (opposed to compiled into the kernel) in order to use realtime-lsm.

In our lab, the *schedtool* command line program proved useful for setting priorities. To use the tool, just launch the program with the required priority, as follows:

```
schedtool -R -p 50 -e ➥
mplayer file.avi
```

The *-R* switch sets the realtime priority; *-p 50* is the value. If the *-e* is missing, Schedtool expects the ID of the active process. The *schedutils* package by Robert Love [6] does a similar thing to Schedtool, but it doesn't support the full set of scheduler priorities.

Working on the Kernel

In our lab, the patch set by Con Kolivas proved to be the most effective approach. Using the patch, MythTV played a decoded DVB stream without video or sound dropouts on a machine with a heavy system load. As the changes are not yet part of the standard kernel, users still need to compile the optimized kernel themselves. Fedora users can check out [7] for binary packages, which are not quite up to date.

Conclusion

Of course, there are limits to the techniques described in this article, and the results this kind of tweaking will give you depend on the current system load. Realtime priority should be considered a matter of making the "best effort," but it doesn't always clear up all your problems. This said, and assuming normal circumstances, the techniques described

INFO

[1] Con Kolivas' patch set: *http://members.optusnet.com.au/ckolivas/kernel/*

[2] Jack audio server: *http://jackit.sourceforge.net*

[3] Realtime-LSM: *http://sourceforge.net/projects/realtime-lsm*

[4] set_rtlimits: *http://www.physics.adelaide.edu.au/~jwoithe*

[5] schedtool: *http://freequaos.host.sk/schedtool*

[6] schedutils: *http://rlove.org/schedutils*

[7] Desktop kernel for Fedora: *http://apt.bea.ki.se/kernel-desktop*

[8] Ingo Molnar's Voluntary Preemption: *http://people.redhat.com/mingo/realtime-preempt*