

Real-time Audio with Linux 2.6

Real-time audio work is of course a relative idea, latency is inherent when processing audio in the digital domain, however it is certainly possible to obtain latencies far below perceptible levels. The steps needed to get such results is exactly what this article will address.

The Kernel

As you most likely already know, much of the configuration needed to be able to reach low latency scheduling is done within the Linux kernel. For the sake of being recent and for ease of use, this article will use the latest 2.6 kernel source. If you are new to building custom kernels it would be wise to go read up on the steps needed to build a kernel that will boot on your system with necessary drivers.

Get, Extract, Patch

Obviously the first step in building a new kernel is to get the source to it. This can be had from <http://www.kernel.org> (at the time of writing the latest stable version is 2.6.22). Next you will need to obtain the latest real-time kernel patch by Ingo Molnar, which is available from his website: <http://people.redhat.com/mingo/realtime-preempt/>. You will want to download the latest version for your choice of kernel version. I.E: patch-2.6.22-rt9 for kernel version 2.6.22. Next un-gzip the patch in the directory of the kernel source and apply it. This is usually done with “patch -p1 -i patch-2.6.xx.x-rtx”. If all goes well you will be ready to configure your kernel.

Config, Build, Boot

The next step is to configure the newly patched kernel. This is easily done with the “make menuconfig” command. Users can also use “make xconfig” for an x11 interface to the configuration. You will want to first make sure that all necessary drivers to boot your computer are built in, usually this is an ide driver and ext3 file system support. Also you will want to go ahead and configure other drivers such as network and video as well as usb/firewire/parallel drivers. I prefer leaving the ALSA drivers for my sound card out of the initial kernel configuration, as I like to download the latest version from the ALSA website and build them later, this is just a personal preference. You will still want to enable sound card support within “Device Drivers”. Building and installing the ALSA drivers will be covered later.

Now for the good stuff. Navigate to the “Processor type and features” category. There will be an option titled “Preemption mode”. Set this to “Complete Preemption (Real Time)”. It is also commonly recommended to enable the high resolution timer option. There are also some additional tweaks which are commonly recommended by the Linux Audio Users (<http://lad.linuxaudio.org/subscribe/lau.html>) community. These include selecting deadline as the default IO scheduler (Block Layer -> IO Schedulers), enabling HPET timer support (Processor type and features), turning on enhanced real time clock support (Device Drivers -> Character Devices), turning off priority inheritance debugging (Device Drivers -> Character Devices), and enabling the high precision event timer (Device Drivers -> Character Devices).

Once you have your options and drivers configured you can safely exit and save the configuration. Next you will want to compile or build the kernel image. Your reading on kernel compiling and installation should have made you familiar with this, however a quick `make-kpkg kernel_image` in Debian or Ubuntu will start the process. After your kernel is built and installed it's time to reboot into the new kernel. If everything boots properly then welcome to your new real-time capable system.

System Tuneage

Even though your kernel may be real-time capable, you will most likely still not be able to run jackd with a frame size of 16 (.667 ms latency with 2 periods/buffers). This is where a little software tweaking is needed.

First you will want to build the ALSA sound drivers (and libraries if necessary) for your new kernel. The latest version can be had from the ALSA project website (<http://alsa-project.org>). Note: If you are installing ALSA for the first time you will also want to install the libraries and utilities from the ALSA website as well. After extracting the drivers package navigate into the new directory (alsa-driver-1.0.14 at the time of writing) and issue the following command:

```
./configure --with-sequencer=yes --with-oss=yes --with-cards=<drivers>
```

you will want to replace <drivers> in the above command to include a comma separated list of the drivers for the sound card(s) on your system. `./configure --help` will give you a full list of drivers and options, you can also find out which driver you need through the ALSA sound card matrix.

Once the ALSA system is configured simply issue the “make” command and watch as your new driver system is built before your eyes. After it's built, as root issue “make install” to install the modules into the `/lib/modules/<kernel version>` directory. After this you can install the libraries and utilities if necessary with simple “`./configure`” and “`make&&make install`” commands. When all this is done you should be able to adjust settings and volumes with the `alsamixer` utility. A few `modprobes` may be necessary to get all the drivers loaded properly but they are usually loaded automatically.

After ALSA is installed we will need to modify the PAM security system to allow regular users to operate software at a high priority (real time) level. The file

that will allow this is “/etc/security/limits.conf” (in Debian based systems). You will need to add the following lines as root to limits.conf.

```
@audio - rtprio 95
@audio - memlock 512000
@audio - nice -19
```

This example is from Florian Schmidt's website (http://tapas.affenbande.org/wordpress/?page_id=73) and is a recommended default only, though it should work nicely for any system. The “rtprio” setting is the maximum priority a user of the “audio” group can run a task. The “memlock” setting is the maximum amount of memory that a member of the “audio” group can lock with a realtime task. This should be less than your maximum physical amount of memory, some recommend it to be half. “nice” is the minimum “nice level” a task can be run as (the willingness of a task to give up it's cpu time). These settings will get jackd, ardour and other RT capable programs running at higher priorities, but there is still a bit more to do.

The next and final step is to set a few thread priorities. The real time patch should have created several IRQ threads as well as added a “hrtimer” thread. These can be viewed with a quick `ps -e`. You will need the `chrt` command to set thread priorities (in Debian this is part of the `schedutils` package). It doesn't really matter which order you set the priorities just that it is done properly. Scan the output of the `ps -e` command looking for threads like “softirq-timer” and “softirq-hrtimer”, making note of the pid of each one (on my dual core system there are two of each, one for each cpu). Also you will want to find the pid of the IRQ thread that your sound card resides on. Looking at the output of `cat /proc/interrupts` should reveal this. Then all you have to do is issue a few commands and you're done. The basic format for these are: `chrt -f -p 99 <pid>` So your commands might be as follows.

```
$: chrt -f -p 99 6
$: chrt -f -p 99 12
$: chrt -f -p 99 818
```

These will set the threads identified (6, 12, and 818) to a real-time priority level (99), of course that high of a priority may not be necessary and a bit insecure, feel free to tweak as you see fit. These commands are the last settings that need to be changed and allow the kernel timers to operate on a much higher priority, thus insuring more satisfactory scheduling latencies. The priorities of these processes will need to be reset after each reboot, so a simple init script would suffice for doing this as the pids shouldn't change.

Start recording

The only thing left to do of course is to start recording and making music. You should be able to run jackd with much lower frame sizes than before and much fewer xruns. Enjoy!

Complements of Dana Simmons (dcsimon), Waxy Ear Recordings 2007.