Notes on Kernel OSS-Emulation

Jan. 22, 2004 Takashi Iwai <tiwai@suse.de>

Modules

ALSA provides a powerful OSS emulation on the kernel. The OSS emulation for PCM, mixer and sequencer devices is implemented as add-on kernel modules, snd-pcm-oss, snd-mixer-oss and snd-seq-oss. When you need to access the OSS PCM, mixer or sequencer devices, the corresponding module has to be loaded.

These modules are loaded automatically when the corresponding service is called. The alias is defined `sound-service-x-y`, where x and y are the card number and the minor unit number. Usually you don't have to define these aliases by yourself.

Only necessary step for auto-loading of OSS modules is to define the card alias in `/etc/modprobe.d/alsa.conf`, such as:

```
alias sound-slot-0 snd-emu10k1
```

As the second card, define `sound-slot-1` as well. Note that you can't use the aliased name as the target name (i.e. `alias sound-slot-0 snd-card-0` doesn't work any more like the old modutils).

The currently available OSS configuration is shown in /proc/asound/oss/sndstat. This shows in the same syntax of /dev/sndstat, which is available on the commercial OSS driver. On ALSA, you can symlink /dev/sndstat to this proc file.

Please note that the devices listed in this proc file appear only after the corresponding OSS-emulation module is loaded. Don't worry even if "NOT ENABLED IN CONFIG" is shown in it.

Device Mapping

ALSA supports the following OSS device files:

```
PCM:
        /dev/dspX
        /dev/adspX

Mixer:
        /dev/mixerX

MIDI:
        /dev/midi0X
        /dev/amidi0X

Sequencer:
        /dev/sequencer
        /dev/sequencer2 (aka /dev/music)
```

where X is the card number from 0 to 7.

(NOTE: Some distributions have the device files like /dev/midi0 and /dev/midi1. They are NOT for OSS but for tclmidi, which is a totally different thing.)

Unlike the real OSS, ALSA cannot use the device files more than the assigned ones. For example, the first card cannot use /dev/dsp1 or /dev/dsp2, but only /dev/dsp0 and /dev/adsp0.

As seen above, PCM and MIDI may have two devices. Usually, the first PCM device (`hw:0,0` in ALSA) is mapped to /dev/dsp and the secondary device (`hw:0,1`) to /dev/adsp (if available). For MIDI, /dev/midi and /dev/amidi, respectively.

You can change this device mapping via the module options of snd-pcm-oss and snd-rawmidi. In the case of PCM, the following options are available for snd-pcm-oss:

dsp_map
: PCM device number assigned to /dev/dspX (default = 0)

adsp_map
: PCM device number assigned to /dev/adspX (default = 1)

For example, to map the third PCM device (`hw:0,2`) to /dev/adsp0, define like this:

```
options snd-pcm-oss adsp_map=2
```

The options take arrays. For configuring the second card, specify two entries separated by comma. For example, to map the third PCM device on the second card to /dev/adsp1, define like below:

```
options snd-pcm-oss adsp_map=0,2
```

To change the mapping of MIDI devices, the following options are available for snd-rawmidi:

midi_map

MIDI device number assigned to /dev/midi0X (default = 0)

amidi_map
: MIDI device number assigned to /dev/amidi0X (default = 1)

For example, to assign the third MIDI device on the first card to /dev/midi00, define as follows:

```
options snd-rawmidi midi_map=2
```

PCM Mode

As default, ALSA emulates the OSS PCM with so-called plugin layer, i.e. tries to convert the sample format, rate or channels automatically when the card doesn't support it natively. This will lead to some problems for some applications like quake or wine, especially if they use the card only in the MMAP mode.

In such a case, you can change the behavior of PCM per application by writing a command to the proc file. There is a proc file for each PCM stream, `/proc/asound/cardX/pcmY[cp]/oss`, where X is the card number (zero-based), Y the PCM device number (zero-based), and `p` is for playback and `c` for capture, respectively. Note that this proc file exists only after snd-pcm-oss module is loaded.

The command sequence has the following syntax:

```
app_name fragments fragment_size [options]
```

`app_name` is the name of application with (higher priority) or without path. `fragments` specifies the number of fragments or zero if no specific number is given. `fragment_size` is the size of fragment in bytes or zero if not given. `options` is the optional parameters. The following options are available:

disable
: the application tries to open a pcm device for this channel but does not want to use it.

direct
: don't use plugins

block
: force block open mode

non-block
: force non-block open mode

partial-frag
: write also partial fragments (affects playback only)

no-silence
: do not fill silence ahead to avoid clicks

The `disable` option is useful when one stream direction (playback or capture) is not handled correctly by the application although the hardware itself does support both directions. The `direct` option is used, as mentioned above, to bypass the automatic conversion and useful for MMAP-applications. For example, to playback the first PCM device without plugins for quake, send a command via echo like the following:

```
% echo "quake 0 0 direct" > /proc/asound/card0/pcm0p/oss
```

While quake wants only playback, you may append the second command to notify driver that only this direction is about to be allocated:

```
% echo "quake 0 0 disable" > /proc/asound/card0/pcm0c/oss
```

The permission of proc files depend on the module options of snd. As default it's set as root, so you'll likely need to be superuser for sending the command above.

The block and non-block options are used to change the behavior of opening the device file.

As default, ALSA behaves as original OSS drivers, i.e. does not block the file when it's busy. The -EBUSY error is returned in this case.

This blocking behavior can be changed globally via nonblock_open module option of snd-pcm-oss. For using the blocking mode as default for OSS devices, define like the following:

```
options snd-pcm-oss nonblock_open=0
```

The `partial-frag` and `no-silence` commands have been added recently. Both commands are for optimization use only. The former command specifies to invoke the write transfer only when the whole fragment is filled. The latter stops writing the silence data ahead automatically. Both are disabled as default.

You can check the currently defined configuration by reading the proc file. The read image can be sent to the proc file again, hence you can save the current configuration

```
% cat /proc/asound/card0/pcm0p/oss > /somewhere/oss-cfg
```

and restore it like

```
% cat /somewhere/oss-cfg > /proc/asound/card0/pcm0p/oss
```

Also, for clearing all the current configuration, send `erase` command as below:

```
% echo "erase" > /proc/asound/card0/pcm0p/oss
```

Mixer Elements

Since ALSA has completely different mixer interface, the emulation of OSS mixer is relatively complicated. ALSA builds up a mixer element from several different ALSA (mixer) controls based on the name string. For example, the volume element SOUND_MIXER_PCM is composed from "PCM Playback Volume" and "PCM Playback Switch" controls for the playback direction and from "PCM Capture Volume" and "PCM Capture Switch" for the capture directory (if exists). When the PCM volume of OSS is changed, all the volume and switch controls above are adjusted automatically.

As default, ALSA uses the following control for OSS volumes:

OSS volume	ALSA control	Index
SOUND_MIXER_VOLUME	Master	0
SOUND_MIXER_BASS	Tone Control - Bass	0
SOUND_MIXER_TREBLE	Tone Control - Treble	0
SOUND_MIXER_SYNTH	Synth	0
SOUND_MIXER_PCM	PCM	0
SOUND_MIXER_SPEAKER	PC Speaker	0
SOUND_MIXER_LINE	Line	0
SOUND_MIXER_MIC	Mic	0
SOUND_MIXER_CD	CD	0
SOUND_MIXER_IMIX	Monitor Mix	0
SOUND_MIXER_ALTPCM	PCM	1
SOUND_MIXER_RECLEV	(not assigned)	
SOUND_MIXER_IGAIN	Capture	0
SOUND_MIXER_OGAIN	Playback	0
SOUND_MIXER_LINE1	Aux	0
SOUND_MIXER_LINE2	Aux	1
SOUND_MIXER_LINE3	Aux	2
SOUND_MIXER_DIGITAL1	Digital	0
SOUND_MIXER_DIGITAL2	Digital	1
SOUND_MIXER_DIGITAL3	Digital	2
SOUND_MIXER_PHONEIN	Phone	0
SOUND_MIXER_PHONEOUT	Phone	1
SOUND_MIXER_VIDEO	Video	0
SOUND_MIXER_RADIO	Radio	0
SOUND_MIXER_MONITOR	Monitor	0

The second column is the base-string of the corresponding ALSA control. In fact, the controls with `XXX [Playback|Capture] [Volume|Switch]` will be checked in addition.

The current assignment of these mixer elements is listed in the proc file, /proc/asound/cardX/oss_mixer, which will be like the following

```
VOLUME "Master" 0
BASS "" 0
TREBLE "" 0
SYNTH "" 0
PCM "PCM" 0
...
```

where the first column is the OSS volume element, the second column the base-string of the corresponding ALSA control, and the third the control index. When the string is empty, it means that the corresponding OSS control is not available.

For changing the assignment, you can write the configuration to this proc file. For example, to map "Wave Playback" to the PCM volume, send the command like the following:

```
% echo 'VOLUME "Wave Playback" 0' > /proc/asound/card0/oss_mixer
```

The command is exactly as same as listed in the proc file. You can change one or more elements, one volume per line. In the last example, both "Wave Playback Volume" and "Wave Playback Switch" will be affected when PCM volume is changed.

Like the case of PCM proc file, the permission of proc files depend on the module options of snd. you'll likely need to be superuser for sending the command above.

As well as in the case of PCM proc file, you can save and restore the current mixer configuration by reading and writing the whole file image.

Duplex Streams

Note that when attempting to use a single device file for playback and capture, the OSS API provides no way to set the format, sample rate or number of channels different in each direction. Thus

```
io_handle = open("device", O_RDWR)
```

will only function correctly if the values are the same in each direction.

To use different values in the two directions, use both

```
input_handle = open("device", O_RDONLY)
output_handle = open("device", O_WRONLY)
```

and set the values for the corresponding handle.

Unsupported Features

MMAP on ICE1712 driver

ICE1712 supports only the unconventional format, interleaved 10-channels 24bit (packed in 32bit) format. Therefore you cannot mmap the buffer as the conventional (mono or 2-channels, 8 or 16bit) format on OSS.