The Linux Sound User's Guide

Version Alpha-1, 6 April 1994.

This book is a guide to using sound under Linux. It describes the user visible aspects of the VoxWare sound drivers used under Linux and several other Intel-based Unix compatible operating systems. It gives background information on sound cards and digital sound and music technology in general, a detailed installation procedure for the Linux sound drivers, and a comprehensive survey of applications supporting sound that run under Linux. A glossary of terms and an extensive list of references to additional material is also included.

Contents

Chapter 1

Preface

Last Christmas, I bought a sound card. I found that the process of setting it up under Linux and locating applications that support sound was somewhat confusing, due to the lack of documentation and a lot of unfamiliar terminology.

After a period of time sorting things out, I was surprised to find that there is a considerable base of applications supporting sound under Linux. In an effort to help others who were struggling with the same problems, I wrote the Linux Sound HOWTO.

This provided some minimal documentation, but I still thought there was a need for a larger document covering sound application issues in more detail. I was also looking for some kind of Linux related project, both as a way of paying back the Linux community, and to achieve one of my other objectives: learning LaTeX. After suggesting the idea to the Usenet news groups, I received enough positive feedback that I decided to undertake the project.

I hope this document is useful to you, and that it will encourage the development of more sound and multimedia applications that run under Linux.

Jeff Tranter
`Jeff_Tranter@Mitel.COM`
Kanata, Ontario, Canada
April, 1994

Chapter 2

Introduction

This document is a user's guide to using sound under Linux. It gives background information on sound card technology, and describes the user visible aspects of the sound drivers for the Linux kernel (including drivers for sound cards as well as the PC speaker). The steps involved in installing a sound card and configuring the kernel are described in detail.

A number of applications available for Linux that support sound are reviewed, and future enhancements to the sound drivers are discussed.

Appendix A contains an extensive list of references to point the reader to sources of additional information. Appendix B is a glossary defining many of the terms related to sound and music technology used in this book.

2.1 Scope

This document is applicable to the Linux operating system, and to a lesser extent, any of the other operating systems that use the VoxWare sound drivers.

The scope is limited to the user visible aspects of sound applications. The technical issues of programming are documented in the Hacker's Guide to VoxWare.

2.2 Disclaimer

I believe one of the factors for the success of Linux is the spirit of cooperation and positive feedback. I don't profess to be an expert on sound hardware and programming. If you feel something is wrong or should be added to this document, *let me know*! I'll try to incorporate your ideas in the next revision.

The rapid pace of Linux development means that some of the information in this document may not be up to date by the time you read this. Check the documents listed in the bibliography and always read the appropriate README files when building the kernel and sound applications.

2.3 Acknowledgments

I would like to thank Hannu Savolainen and the many other contributors to the VoxWare sound drivers and applications.

This book is dedicated to the thousands of Linux activists around the world who are responsible for making Linux a success.

2.4 Copyright

1 Laurie Court
Kanata, Ontario
CANADA K2L 1S2
`Jeff_Tranter@Mitel.COM`

2.5 About This Document

This document was created entirely on the author's 386DX-40 PC running the Slackware release of Linux, kernel version 1.0. The text was entered using GNU emacs and typeset using LaTeX.

Chapter 3

Introduction to Sound Card Technology

In the past, Unix based systems traditionally have not provided much hardware or software support for sound. In the personal computer arena, sound has become increasingly popular, first for games and more recently for commercial multimedia applications.

With the availability of Linux and low cost sound hardware for Intel based PCs, a sound capable Unix system is within the reach of most computer hobbyists.

Possibly because sound support was lacking in Unix systems, many new users are confused by the technical jargon specific to sound and electronic music, and the many sound cards available. This guide will attempt to explain what can be done with sound under Linux, unravel some of the technical terms, and point the reader to sources of more information.

3.1 What Can A Sound Card Do?

The typical sound card hardware provides the capability for some or all of the following functions:

- playing and recording digitized sound samples
- playing music from an audio Compact Disc in a CD-ROM drive
- generating sounds using an internal FM synthesizer
- controlling external MIDI devices
- providing miscellaneous functions, such as a joystick interface, SCSI disk interface, volume and tone controls, and facilities for mixing of audio inputs

Two of Linux's strengths are the ability to perform efficient multitasking and the provision of a well defined interface between applications and kernel drivers (device files). These, combined with low cost sound hardware, make it an ideal platform for developing and running sound applications.

The next few sections in this chapter give background information on sound hardware and terminology. Later chapters will cover Linux specific issues.

3.2 Digital Sampling

Digital sampling is a technique whereby analog waveforms are digitized — essentially stored as numbers. For digitized sounds, there are two basic parameters that have the most influence on sound quality: sampling rate and sample size.

The sampling rate is the speed at which the analog waveform is converted to digital "samples" or numbers. This is expressed in samples per second, or more often (and less accurately), Hertz. The sample size indicates the number of data bits which are stored for each sample; the more bits, the more accurately the sample represents the original waveform. Sounds can also be recorded with one channel (mono), two channels (stereo), or even four (quad).

Various coding schemes are used to represent the sample as a numerical value. Linear coding stores waveforms using values that are proportional to the sound samples. Companding techniques such as μ-law use a variation of a logarithmic scale to allow a greater range of values to be stored in a given number of bits, at the expense of some accuracy.

As an example, a low cost sound card can produce single channel, 8 bit samples at 8000 samples per second. This provides sound quality comparable to the telephone network. By comparisom, a 16 bit sound card producing stereo sound at 44100 samples/second is equivalent to Compact Disc audio quality (ignoring issues such as noise and distortion).

Sound hardware uses a Digital to Analog Converter (D/A converter or DAC) to produce sound from digital samples. Most sound cards also provide an Analog to Digital Converter (A/D converter or ADC) to perform the reverse operation: converting analog sound waveforms to digital samples. Low end sound cards typically provide 8 bit D/A and A/D converters, while high end sound cards are usually 16 bit (although not all 16 bits may be accurate due to lack of precision).

Apart from sampling rate and size, sound cards can be rated on other parameters such as noise, frequency response, distortion, and output power level. They may also provide hardware support for PCM companding or other compression schemes.

Sound samples are typically stored as computer files. Several file formats are in common use. The Audio Formats FAQ describes these formats in detail.

Getting back to the examples given before, let's calculate how much disk storage would be needed for some sound files. A 60 minute audio CD, stored as a raw sound file would require:

$60min \times 60secs/min \times 44100samples/sec \times 2channels \times 2bytes/sample$
$= 635,040,000bytes$

Thus, storing your collection of Music CDs on a hard disk drive would not be too practical! Small sound samples at low sampling rates are more reasonable though:

$10secs \times 8000samples/sec \times 1channel \times 1byte/sample$
$= 80,000bytes$

Digital sampling is now the most popular technique for music synthesis. By digitizing the sounds of real musical instruments, very realistic music can be produced. It also has the flexibility to support speech, or any variety of synthetic sounds. This has only been made affordable by recent advances and cost reductions in hardware, especially memory and D/A converters.

3.3 FM Synthesis

Many sound cards provide hardware for producing sounds using FM synthesis. This is an older technique than digital sampling and is based on modifying waveforms and combining them using various mathematical operations.

The synthesizer hardware provides a number of voices — the number of simultaneously playable notes — typically between 6 and 20. Each voice is defined in terms of several waveform oscillators ("operators"), which are combined to produce a specific sound. FM synthesizers are characterized by the number of voices and operators, and often provide different operating modes that trade off one against the other.

The basic idea is to start with oscillators that can produce sine, triangle, or other simple waveforms with specific frequencies and amplitudes. These are then modulated with an "envelope" defined by parameters such as attack, decay, sustain, and release. Several of these waveforms are then combined in various ways (e.g. added or subtracted). Other effects such as feedback, chorus, echo, and delay can then be added.

The synthesizer parameter settings for a specific sound are known as a patch, and libraries of these are stored on disk as patch files.

If this sounds like a convoluted way to produce the sound of a musical instrument, you are getting the right idea. Most users will use a predefined set of standard patches, and never change them. A typical patch file defines 128 melodic and 47 percussion instruments, providing quite a wide range of sounds. If you want to learn more about FM patches, check out the Glib package described later.

The advantages of FM synthesis are that it requires reasonably simple hardware and little software overhead to produce. The main disadvantage is that determining the patch parameters is something of a "black art". Producing realistic instrument sounds is difficult, and the results often leave something to be desired.

Professional music systems have generally moved to sampling techniques, but FM synthesis is still useful for backwards compatibility with older equipment and for providing basic sounds with little CPU overhead (e.g. sound tracks for games).

3.4 Wavetable Synthesis

The current state of the art for sound hardware is Wavetable Synthesis. This technique combines the best of sampling and FM synthesis. Sampled voices are typically downloaded into memory on the sound card. The sound hardware then provides facilities for modifying these sound samples as needed (e.g. frequency, attack, decay) in real time, to play different notes.

Thus, this technique provides the flexibility and realism of digital sampling, while achieving the reduced software complexity and overhead of FM synthesis. Sound cards using Wavetable Synthesis can easily simulate FM synthesizers. The disadvantage of this technique is that cards supporting wavetable synthesis are currently more expensive, and are not fully compatible with the older sound cards.

3.5 MIDI

MIDI stands for Musical Instrument Digital Interface, and as the name suggests, is a standard hardware and software protocol for allowing electronic musical instruments to communicate. Events such as key presses are sent over the MIDI bus, allowing the control of devices in real time. This unidirectional[1] serial interface is commonly used to connect professional music equipment such as synthesizers, music keyboards, drum machines, synthesizers, and even lighting equipment.

Many sound cards provide a MIDI port (sometimes as an option). Using the port you can communicate with and control external MIDI devices. You can do such things as "play" synthesizers using MIDI commands, or capture music keyboard events and store them as a MIDI file for later playback or editing. MPU-401 is an industry standard for smart MIDI interface cards; some sound boards are compatible with this, others provide only a subset of the functionality.

MIDI files are a standardized way of storing MIDI events in a computer. They can represent the "notes" of a song, or the setup of MIDI devices. MIDI files can often also be played using the onboard facilities of a sound card, in which case no MIDI bus is involved. The "Standard MIDI" specification also defines a number of instruments or patches that conforming devices need to provide (128 melodic instruments and 47 percussion sounds).

3.6 Adagio

MIDI is not meant to be a human readable format. Adagio is a music description language developed at Carnegie Mellon University. Musical scores can be composed using Adagio, and played using the appropriate tools. Programs to convert between Adagio and MIDI files are also available. For more information, see the manual that is included with the Linux Adagio package.

3.7 MOD Files

MIDI and Adagio describes songs in terms of note events, but do not completely describe the sounds used. MOD files (an abbreviation for module files) are a self contained format for describing a melody as well as the sampled instruments used to play the song. The format was originally developed on and tailored for the Commodore Amiga computer, but MOD file players are now available for other platforms as well[2].

[1] The signals are unidirectional on each MIDI cable, but devices typically provide IN and OUT ports.

[2] I have been unable to find any MOD file editors that run under Linux, though.

There is a wealth of free MOD files available on electronic bulletin board systems and internet archive sites, produced by amateur and semi-professional musicians.

The advantage of the MOD file format is that it completely describes a song in a flexible and reasonably compact format (typically 30K to 300K bytes in size). The main disadvantage is that generating the effects needed for MOD files in real time requires a lot of CPU processing power, unless they can be performed in hardware (e.g. using wavetable synthesis).

3.8 MPEG Audio

MPEG is a working group formed under ISO to define standards for digital video and audio compression. The audio portion of MPEG uses sophisticated algorithms optimized for digital audio that allow quite high compression rates (theoretically up to 22:1 but typically about 6:1). It is a lossy algorithm, meaning that the original data is not reproduced exactly, but a listener typically cannot hear the difference.

There are several levels of MPEG audio standards under development. The lower levels require considerable processing to perform compression and playback, but are within the reach of PCs and workstations and some commercial and freeware software is now available. Higher levels require specialized encoder/decoder hardware.

MPEG is slated to be used for applications such as CD-I, digital radio, HDTV, and DBS satellite transmission.

A group known as the Internet Underground Music Archive (IUMA) is using the internet to distribute music in MPEG audio format. Their goal is to maintain a publicly accessible, fast internet site that archives the music, artwork and text of any musician, group or band that wishes their music to be internationally and freely distributed. With the internet experiencing tremendous growth (20 million current users and hundreds of thousands users added each month) they expect the Internet Underground Music Archive to be accessible to the majority of people with computers by the late 1990s.

3.9 Internet Talk Radio

Internet Talk Radio, “the first station in cyberspace”, is a professionally produced source of news and information distributed freely on the internet.

It’s run like a radio station, but publishes sound files (8 bit, 8000 sample/second μ-law encoded); typically anywhere from 30 to 90 minutes per day[3]. The files can be retrieved by anonymous FTP access on the internet and played using a sound card. They claim to reach 100,000 people in 30 countries.

[3] 90 minutes of audio in this format requires approximately 45 megabytes of storage.

3.10 CD-ROM

Many sound cards also provide a CD-ROM port, and the ability to mix the analog output of the CD-ROM drive with the sound card input. This allows playing audio CDs through the sound card. Perhaps surprisingly, only a very few CD-ROM drives allow getting digital access to the sound data for audio CDs.

Some of the SoundBlaster type cards provide a proprietary CD-ROM interface that is only compatible with certain drives. Others provide a full SCSI interface that can be used for other types of devices, such as hard disk drives, as well.

3.11 Miscellaneous Functions

Many sound cards provide additional features which can be quite useful.

A facility for selecting between several audio input sources and mixing of the sources is often provided, as well as an external volume control. Some cards also provide software driven tone controls.

A standard joystick port is useful for games[4]. A MIDI port, previously described, is sometimes supported using the same connector with a special cable.

Most cards also provide facilities for controlling settings such as I/O port, IRQ number, and DMA channels. On some cards these are jumpers, and on others they can be changed under software control.

In order to use a sound card you will also need speakers; these are often the magnetically shielded, amplified type which are designed for use with sound cards.

For recording live sounds, a microphone is also handy.

3.12 Sound Card Comparison

Table 3.1 lists a number of popular sound cards, and compares their basic features. The intent here is to provide some basis of comparison, not to recommend any one sound card over another. What's right for you depends on several factors, including your budget and your intended application. It is even conceivable that you might want to install more than one sound card. When interpreting the data in the table, keep in mind that some cards exist in different versions or with options that have slightly different capabilities, and that parameters such as the number of operators and voices vary depending on the operating mode selected. Many features have been omitted.

[4] Make sure you don't have a conflict between joystick ports on a sound card and a game or multi-function i/o board.

Sound Card	**SB**	**SB Pro**	**SB 16**	**GUS**	**PAS16**
Digital Playback:					
bits	8	8	16	16	16
max rate	44KHz	44KHz	44KHz	44KHz	44KHz
channels	1	2	2	2	2
Digital Recording:					
bits	8	8	16	8	16
max rate	12KHz	44KHz	44kHz	44KHz	44KHz
channels	1	2	2	2	2
FM Synthesis:					
voices	11	11 or 20	11 or 20	-	11 or 20
operators	2	2 or 4	2 or 4	-	2 or 4
Wavetable Synthesis:					
voices	-	-	-	32	-
Other Features:					
MIDI	N	Y	Y	Y	N
CD-ROM	N	Y	Y	N	N
SCSI	N	N	Y	N	Y
joystick port	Y	Y	Y	N	N

Table 3.1: Sound Card Comparison

Chapter 4

The VoxWare Sound Driver

VoxWare is the name of the kernel sound drivers used under Linux. This chapter describes the drivers in detail, and explains how to install, configure, and test your sound card.

4.1 Overview

The kernel sound drivers are the software that talks to the sound hardware. Application programs are insulated from these low level details and only need know about the application layer interface at the device file level. This makes it possible for applications to be independent of the specifics of any one sound card. Very briefly, the device files provided by the kernel drivers are the following:

- `/dev/audio` (Sun workstation compatible audio device; read/write)
- `/dev/dsp` (digital sampling device; read/write)
- `/dev/mixer` (sound mixer; read/write)
- `/dev/sequencer` (MIDI, FM, and GUS synthesizer access; read/write)
- `/dev/midi` (MIDI device; not yet implemented in current sound driver)
- `/dev/sndstat` (displays sound driver status when read)
- `/dev/audio1` (for second sound card)
- `/dev/dsp1` (for second sound card)

Not all device files are supported by all sound cards. The VoxWare drivers currently support these cards:

- Roland MPU-401 MIDI interface
- AdLib

- SoundBlaster and compatibles (including ThunderBoard and ATI Stereo F/X)
- SoundBlaster Pro
- SoundBlaster 16
- ProAudioSpectrum 16 (and the compatible Logitech SoundMan 16)
- Gravis UltraSound

Some other MPU-401 compatible cards may also work (WaveBlaster, Turtle Beach Maui, etc...). Support for other popular sound cards will likely be added in the future.

The Linux kernel also supports the SCSI interface provided on some sound cards (e.g. ProAudioSpectrum 16) and the proprietary CD-ROM interface provided on the SoundBlaster Pro and SoundBlaster 16. There are also joystick drivers available as kernel patches. All of these are separate from the sound card drivers.

4.2 Installation and Configuration

Setting up Linux to support a sound card involves the following steps:

1. Installing the sound card.
2. Configuring and building the kernel with the sound drivers.
3. Creating the sound device files.
4. Testing the installation.

We'll describe each of these steps in detail in the next few sections. If you are building the VoxWare drivers on an operating system other than Linux, then the procedure will be slightly different.

4.2.1 Installation

The first requirement, if you have not already done so, is to install the sound card[1]. Follow the instructions provided by the manufacturer. Be sure to note down the jumper settings for IRQ, DMA channel, and so on; if you are unsure, use the factory defaults. Try to avoid conflicts with other devices (e.g. ethernet cards, joystick port) if possible. You will also need to connect the speakers, cables, and a microphone if you want to do any live recording. Incidently, a math coprocessor is useful for some sound applications (e.g. changing file formats, adding effects, or speech synthesis), but the kernel sound driver itself does not do any floating point calculations.

[1] Or have your dealer install it if you are not comfortable working with hardware.

4.2.2 Configuring the Kernel

The next step is to configure the Linux kernel. If you are using a recent version (0.99 patch level 14 or later), the sound drivers are included with the kernel release. Follow your usual procedure for building the kernel. When you configure the kernel, enable the sound driver, and answer the questions about sound card settings when prompted by the configure program.

Here is a example of configuring the sound driver for a SoundBlaster Pro card:

```
% cd /usr/src/linux
% make config
Kernel math emulation (CONFIG_MATH_EMULATION) [y] y
...
Sound card support (CONFIG_SOUND) [n] y
...
Compiling Sound Driver v 2.4 for Linux

Configuring the sound support

Do you want to include full version of the driver (n/y) ? n
Do you want to DISABLE the Sound Driver (n/y) ?n
The SoundBlaster, AdLib and ProAudioSpectrum
cards cannot be installed at the same time

Select at most one of them:
- ProAudioSpectrum 16
- SoundBlaster / SB Pro
        (Could be selected with PAS16 also
  since there is a SB emulation on it)
- AdLib

Don't enable SoundBlaster if you have GUS at 0x220!

  ProAudioSpectrum 16 support (n/y) ? n
  SoundBlaster support (n/y) ? y

The following cards should work with any other cards.
CAUTION! Don't enable MPU-401 if you don't have it.
  Gravis Ultrasound support (n/y) ? n
  MPU-401 support (NOT for SB16) (n/y) ? n

Select one or more of the following options
  SoundBlaster Pro support (y/n) ? y
  SoundBlaster 16 support (y/n) ? n
  digitized voice support (y/n) ? y
```

```
  MIDI interface support (y/n) ? y
  FM synthesizer (YM3812/OPL-3) support (y/n) ? y

IRQ number for SoundBlaster?
The IRQ address is defined by the jumpers on your card and
7 is the factory default. Valid values are 9, 5, 7 and 10.
Enter the value: 5
SoundBlaster IRQ set to 5

DMA channel for SoundBlaster?
For SB 1.0, 1.5 and 2.0 this MUST be 1
SB Pro supports DMA channels 0, 1 and 3 (jumper)
For SB16 give the 8 bit DMA# here
The default value is 1
Enter the value: 1
SoundBlaster DMA set to 1

Select DMA buffer size (4096, 16384, 32768, 65536 bytes)
32768 is recommended value for this configuration.
Enter the value: 32768
The DMA buffer size set to 32768
The sound driver is now configured.
...
```

There are a few areas of possible confusion when configuring the sound driver.

Answering "yes" to the "Do you want to include full version of the sound driver" question results in compiling in support for *all* of the sound cards. This is probably not what you require unless you are building a generic kernel that will be configured at boot time using command line options.

The question "Do you want to DISABLE the Sound Driver" is used to compile sound support into the kernel, but disable it (presumably to be enabled by kernel command line options). If you don't want the sound driver at all, the best way is to give a negative answer to the earlier question "Sound card support (CONFIG_SOUND)".

If you have a SoundBlaster of any type, answer yes to the question "SoundBlaster support (n/y) ?". You will specify which model later.

The question "MPU-401 support (NOT for SB16) (n/y) ?" refers to an MPU-401 MIDI interface card, and not the interface provided on some sound cards.

The "digitized voice support (y/n)" question is referring to support for sampled sound (`/dev/dsp` and `/dev/audio`).

The "MIDI interface support (y/n) ?" question refers to the MIDI interface provided on a sound card.

The values for IRQ and DMA that are listed as factory defaults may not necessarily be the default values for your sound card. Check the manufacturer's documentation to be sure.

The sound driver supports more than one sound card at a time, with some constraints. The configure program should tell you if you choose any invalid or incompatible options.

If you have upgraded from an earlier kernel, you should ensure that the sound related header files are correct. The correct header files for sound applications are `soundcard.h` and `ultrasound.h` in the directory `/usr/include/linux`. If you have files by this name in `/usr/include/sys`, they should only `#include` the above files. For example, the file `soundcard.h` should contain only this line:

```
#include <linux/soundcard.h>
```

This will ensure that when you upgrade the kernel, the appropriate header files are also updated.

4.2.3 Creating Device Files

Once the kernel is configured, you need to create the sound device files. The easiest way to do this is to cut the short shell script from the end of the file `/usr/src/linux/drivers/sound/Readme.linux`, and run it as root. Alternatively, you can create the device files manually using the following commands (as root):

```
mknod -m 666 /dev/mixer c 14 0
mknod -m 666 /dev/sequencer c 14 1
mknod -m 666 /dev/midi c 14 2
mknod -m 666 /dev/dsp c 14 3
mknod -m 666 /dev/audio c 14 4
mknod -m 666 /dev/sndstat c 14 6
mknod -m 666 /dev/mixer1 c 14 16
mknod -m 666 /dev/dsp1 c 14 19
mknod -m 666 /dev/audio1 c 14 20
```

4.2.4 Testing

Now that the kernel is configured and the device files created, you can verify the sound hardware and software. Follow your usual procedure for installing and rebooting the new kernel (keeping the old kernel around in case of problems, of course). Verify that sound card is recognized during kernel initialization. You should see a message such as the following on powerup:

```
snd2 <SoundBlaster Pro 3.2> at 0x220 irq 5 drq 1
snd1 <Yamaha OPL-3 FM> at 0x388 irq 0 drq 0
```

This should match your sound card type, IRQ, and DMA settings. The driver may also display some error messages and warnings during boot up. Watch for these when booting the first time after configuring the sound driver.

Assuming your sound card has digital sampling capability, you should now be ready to play a sound file[2], and send it to the sound device as a basic check of sound output, e.g.

```
% cat endoftheworld >/dev/dsp
% cat crash.au >/dev/audio
```

If you have sound input capability, you can do a quick test of this using commands such as the following:

```
# record 4 seconds of audio from microphone
% dd bs=8k count=4 </dev/audio >sample.au
# play back sound
% cat sample.au >/dev/audio
```

If these tests pass, you can be reasonably confident that the sound hardware and software are working. If you experience problems, read the chapter on troubleshooting.

[2]The snd-data package described in section 7 contains some sample sound files that you can use for this purpose.

Chapter 5

The PC Speaker Driver

For those who do not (yet) have sound hardware, there are a couple of other options. With a little hardware, a sound interface can be built using a parallel printer port. For a zero cost solution, there is even a sound driver for the internal speaker of your PC. The driver is compatible with the sound card driver, but the quality may leave something to be desired.

The PC speaker sound driver is not part of the standard kernel distribution; you will have to obtain it separately (at time of writing, the most recent version was `pcsndrv-0.6`).

The PC speaker sound driver uses the following device files:

- `/dev/pcaudio` (equivalent to `/dev/audio`)
- `/dev/pcsp` (equivalent to `/dev/dsp`)
- `/dev/pcmixer` (equivalent to `/dev/mixer`)

The following sound output devices are supported:

- Stereo-on-One (a DAC circuit that plays stereo sound on a single parallel printer port, designed by Mark J. Cox)
- PC Speaker (the default if no Stereo-on-One was found)
- Mono DAC on one parallel port
- Stereo DAC on two parallel ports

Instructions for building the parallel port DAC circuits are included with the sound driver. You can select which output device to use and change parameters such as the sampling rate with the supplied "pcsel" program.

The sound driver is compatible with the real sound card drivers, and most applications should work (excluding features such as MIDI, FM synthesis, and sound recording). The sound quality from the PC speaker may leave something to be desired – it seems to depend heavily on the speaker. The Stereo-on-One and DACs provide quite good quality sound, but like the speaker driver, there is a lot of CPU processing overhead. You may find it slows your machine down too much for some games, and slower machines, such as 386SX class, will probably not be able to play MOD files in real time.

The basic steps involved in installation are the following:

1. Back up your kernel sources.
2. Untar the distribution into `/usr/src`.
3. Apply the supplied kernel patch.
4. Configure and build the kernel.
5. Create the device files using the "pcpsinstall" script.
6. Build the "pcsel" utility.
7. Test the installation.

If you don't have a real soundcard, then you should create symbolic links for `/dev/audio`, `/dev/dsp`, and `/dev/mixer` that point to the corresponding PC speaker device files. If you do have a real sound card, the PC speaker can coexist with the normal sound card driver.

For more details, read the documentation included with the driver. This package was put together by Michael Beck (`beck@informatik.hu-berlin.de`).

Chapter 6

Troubleshooting

This chapter lists a number of potential problems related to the sound driver and suggests likely solutions. You should also check the README files in the sound driver source directory to find out about any known bugs, limitations, or incompatibilities.

6.0.5 No sound card detected

If no sound card is detected when booting, there are a couple of possible reasons. The configuration of the driver could be incorrect and the driver was not able to detect your card in the given I/O address. Another common error is not having the sound driver in the kernel, because you booted with an old kernel instead of the one that was just compiled.

Reading the sound driver status device file provides additional information on whether the sound card driver initialized properly. Sample output should look something like this:

```
% cat /dev/sndstat
Sound Driver:2.4 (Fri Mar 11 11:53:15 EST 1994 root@fizzbin)
Config options: 1aa2

HW config:
Type 2: SoundBlaster at 0x220 irq 5 drq 1
Type 1: AdLib at 0x388 irq 0 drq 0

PCM devices:
0: SoundBlaster Pro 3.2

Synth devices:
0: Yamaha OPL-3

Midi devices:
0: SoundBlaster
```

```
Mixer(s) installed
```

If the cat command displays "`cat: /dev/sndstat: No such device`", then the sound driver is not active in the kernel. If the printout contains no devices (PCM, Synth or Midi), then your soundcard was not detected. Verify that you entered the correct information when configuring the sound driver. Also check that you are running the kernel that you compiled with sound support (type "`uname -a`" and check the compile date and time). Make sure that any IRQ and DMA settings match those set on any jumpers on the sound card.

6.0.6 Incomplete playing of sound files

Another symptom of trouble is being able to play about a second of digitized audio and then having the sound stop, possibly with an error message about "missing IRQ" or "DMA Timed out". This is most likely caused by having IRQ or DMA channel jumper settings that do not match the values entered when configuring the sound driver. You should verify that the kernel configuration matches the sound card jumper settings and that they do not conflict with some other device (e.g. network or SCSI interface).

6.0.7 Pauses while playing MOD files

On slower machines, there may be pauses or dropouts when playing MOD files. This is due to the considerable CPU processing required. You can correct this by:

- eliminating other processes
- using a lower sampling rate or mono mode
- upgrading to a more powerful sound card (e.g. Gravis Ultrasound)
- upgrading to a faster computer

In general, any 386 machine should be able to play samples or FM synthesized music on an 8 bit soundcard with ease. Playing MOD files, however, requires considerable CPU resources. Some experimental measurements have shown that playing at 44KHz requires more than 40% of the speed of a 486/50 and a 386/25 can hardly play faster than 22 KHz (these are with an 8 bit card sound such as a SoundBlaster). A card such as the Gravis Ultrasound performs more functions in hardware and will require less CPU resources.

6.0.8 Problems with FM synthesis on an older SB Pro

If you have one of the older SoundBlaster Pro cards (the SB Pro 1), it does not have the OPL-3 chip. This will cause problems when using the FM synthesizer. The problem is that both the SB Pro 1 and the SB Pro 2 return the same DSP version number so the driver has no way to distinguish between these two models. It enables the OPL-3 mode, which is incorrect for the SB Pro 1. A simple fix is to comment out one line in the kernel sound driver file `sb_dsp.c`. The line is 688 (in version 2.4) and contains the call to `enable_opl3_mode()`:

```
686 if (major > 3 || (major == 3 && minor > 0)) /* SB Pro2 or later */
687   {
688     enable_opl3_mode (OPL3_LEFT, OPL3_RIGHT, OPL3_BOTH);
689   }
```

6.0.9 Problems with a SoundBlaster "compatible" card

Some users experience problems when using a sound card that is not made by Creative Labs but is claimed to be "SoundBlaster compatible". The sound driver may recognize the sound card and work intermittently or it may not recognize the card at all.

The sound driver only supports SoundBlaster clones that are 100% register compatible. If the card is not detected under Linux, it is likely not register compatible and will simply not work.

Some clones are mostly compatible but have proprietary "enhancements" which the original SoundBlaster doesn't have, such as a mixer. In some cases, booting DOS and loading the vendor's sound driver will initialize the card properly such that it will then work under Linux (until the next time it is powered down).

6.0.10 Compile errors when building applications that support sound

The version 1.0c and earlier sound driver used a different and incompatible `ioctl()` scheme. Your application may have been written for these drivers. If this is the case, you should obtain newer source code or make the necessary changes to adapt it to the new sound driver (see the sound driver README files for details).

You should also ensure that you have used the latest version of `soundcard.h` and `ultrasound.h` when compiling the application. See the section on installation for details.

6.0.11 Segmentation violations when running sound application binaries

This again is probably an incompatibility problem due to being compiled with the older sound drivers. The application should be recompiled.

Chapter 7

Sound Applications

To make use of the sound support in the Linux kernel, applications are needed. This chapter describes a number of important sound applications that run under Linux.

A word of warning — because the sound driver has evolved over a period of time, you may encounter some applications that do not compile or execute properly. They may have been written for an earlier version of the sound drivers (the version 1.x drivers were not fully compatible with the 2.x versions). If you have problems, you may be able to find a newer version of the application that works properly. Alternatively, you may be able to make the minor changes needed to the source code in order to get the application to work yourself. If you still have problems and have read all of the README files, you can pose a question to Usenet or the SOUND channel of the activists list.

Because the pace of development is so fast, I have not listed version information or archive sites for these applications. You should consult sources such as the Linux Software Map or "archie" to get the latest information (see the reference section for details).

I have personally tried *most* of these applications, but not all of them (some require hardware that I don't have[1], such as a CD-ROM).

If you develop or come across any new sound applications which are not listed here that you feel should be, let me know and I'll add them to the next version of this book.

7.1 Applications for Digitized Voice

These applications are those that make use of the digital sampling capability of sound cards.

7.1.1 Snd-Driv

This package is the VoxWare sound driver. Since the driver is included with the standard Linux kernel distribution, this is really only of use to those who are using the driver with an operating

[1] yet :-)

system other than Linux. It was written by Hannu Savolainen (`hsavolai@cs.helsinki.fi`).

7.1.2 Snd-Util

This package includes a number of useful sound utilities that run under Linux. Included are:

- etc.magic.add (entries to add the the /etc/magic file to enable the "file" command to recognize some common types of sound files)
- soundinstall (a script to create the sound device files)
- mixer.c (a sample mixer program)
- aumix (a curses based mixer program; also available separately)
- str (a MOD file player)
- srec (a simple sound recording program)
- splay (a simple sound playing program)
- vplay (an improved sound recorder)
- xvmixer (a mixer program that uses an XView GUI)
- gmod (a MOD file player for GUS cards)

Many of these utilities are now obsolete and have been superseded by improved versions that are available as separate packages. The snd-util package was put together by Hannu Savolainen (`hsavolai@cs.helsinki.fi`).

7.1.3 Snd-Data

This package complements "snd-util" and provides some useful data files related to sound. It includes:

- a digitized audio file (`endoftheworld`)
- some MOD files (both 15 and 32 voice)
- some MIDI files
- a large collection of FM patch files (over 400)

This package was put together by Hannu Savolainen (`hsavolai@cs.helsinki.fi`).

7.1.4 Sox

"Sox" (for SOund eXchange) is a powerful utility for converting sound files from one format to another and performing effects. It understands all of the common sound formats, and provides the ability to manipulate non-standard types. The program "play" is an alias for sox that can play sound files under Linux.

This is one of the few utilities that runs much more quickly if a math coprocessor is present.

Sox was written by Lance Norskog (`thinman@netcom.com`); the Linux extensions (known as LSox) were done by Greg Lee (`lee@uhunix.uhcc.hawaii.edu`).

7.1.5 Wavplay

This package contains the "play" and "record" utilities for playing and recording .wav files. Also included is "xplay", an X-based utility for playing or recording of .wav files. It allows marking and zooming in on parts of the file, which is shown in graphical format.

If you are running the newer sound drivers (2.x), you also need to install a patch to the source. The package was written by Andre Fuechsel (`af1@irz.inf.tu-dresden.de`).

7.1.6 Aumix

"Aumix" is an audio mixer program with a colourful interface built with ncurses. Also included in the package is an experimental dB meter program. It was written by Savio Lam (`lam836@cs.cuhk.hk`).

7.1.7 Xmix

This is a mixer program for Linux that runs under X11. It was written by Hal Brand (`brand@netcom.com`).

7.1.8 Maplay

Maplay is an MPEG audio player which runs under Linux (with some patches). Because of the processing requirements, it only supports mono output, and playback is said to be a little choppy even on a 66MHz 486.

The original author is Tobias Bading; the Linux patches are by Louis P. Kruger (lp-kruger@tucson.princeton.edu).

7.2 Applications for MIDI and Music

7.2.1 Adagio

This package contains several utilities for playing and manipulating Adagio and MIDI files. Much of the program code is from a toolkit written at Carnegie Mellon University, with changes for Linux. It contains the following programs:

- mp (plays MIDI files or converts them to Adagio scores)
- xmp (an X11 GUI version of "mp")
- ad (plays an Adagio score or creates a file of MIDI information than can be sent to the tracks program)
- tracks (adds header information to one or more files created with "ad" to form a standard MIDI file)
- fmkeys (plays from a keyboard connected to the external MIDI port)
- midt (disassembles MIDI files)
- tm (assembles one or more MIDI dumps produced by "midt")
- setfm (downloads patch files to the sound driver)

It also includes the Carnegie Mellon MIDI Toolkit manual and some patch files. The package was put together by Greg Lee (`lee@uhunix.uhcc.hawaii.edu`).

7.2.2 Glib

"Glib" is a text based librarian and patch editor for electronic synthesizers. Xgl is an X Windows (XView) based interface, which provides the means to modify wave envelopes graphically.

It has support for the SoundBlaster (2 operator) cards, 4 operator OPL-3 cards (e.g. SoundBlaster Pro), limited support for the Gravis Ultrasound, and the Kawaii K1.

Included is a document on programming the AdLib/SoundBlaster FM music chips by Jeffrey Lee (`jlee@smylex.uucp`).

It was originally written by Tim Thompson (...twitch!glimmer!tjt), and many others have contributed to it. The Linux package was put together by Greg Lee (`lee@uhunix.uhcc.hawaii.edu`).

7.2.3 Tracker

"Tracker" is a utility for playing Amiga MOD files under Linux (it also runs on many other machines and operating systems). It is command line based. Most MOD files produced by the programs soundtracker, noiseplayer, and protracker can be played.

The author is Marc Espie (`espie@ens.fr`).

7.2.4 Gmod

"Gmod" is a MOD file player specifically for the Gravis Ultrasound card. Supported formats are 4/6/8 channel MOD, 669, MultiTracker, and UltraTracker.

Gmod was written by Robert Sanders (`gt8134b@prism.gatech.edu`) and Andrew J. Robinson (`robinson@cnj.digex.com`), based on code by Hannu Savolainen (`hsavolai@cs.helsinki.fi`).

7.2.5 Mod

"Mod" is yet another MOD file player for the Gravis Ultrasound Card. It may support more sound file formats in the future. The author is Mikael Nordqvist (`mech@df.lth.se`)

7.2.6 Mp

"Mp" is a MIDI file player (this appears to be an older version and is superseded by the one included in the Adagio package).

7.2.7 Cdplayer

"Cdplayer" is a very simple command line based utility to play audio CDs in a CD-ROM drive. It provides the ability to play, start from a given track, stop, pause, and show status information.

The author is Mark Buckaway (`mark@datasoft.north.net`).

7.2.8 XMitsumi

Xmitsumi is a simple X GUI interface to control audio playback on a Mitsumi LU-005 CD-ROM drive. It may or may not work on other Mitsumi or other low end drives. It was written by Liem Bahneman (`roland@cac.washington.edu`).

7.2.9 WorkBone

WorkBone is a simple text mode interactive control interface for playing audio CDs under Linux. It makes use of the numeric keypad as a control panel for playing, stopping, pausing, and moving track to track.

Workbone was based on the WorkMan program. The author is Thomas McWilliams (`thomas.mcwilliams@f615.n109.z1.fidonet.org`)

7.2.10 WorkMan

WorkMan is a GUI based interface for playing audio CDs that runs under X. It provides a similar interface and most of the features (e.g. shuffle, programming) of audio CD players. Also included is a database of over 750 music CDs, including track names. Workman runs on several different computer platforms.

Workman was written by Steven Grimm (`koreth@hyperion.com`). The Linux port is by Chris Newbold (`ctne_ltd@uhura.cc.rochester.edu`), who also supports it.

7.3 Applications for Network Sound Support

X provides a portable, network transparent windowing system, but the sound support is essentially limited to making the speaker beep. Multimedia applications would benefit from a standard protocol and toolkit for distributed, network sound support. Several protocols are currently available, but there is not yet any one standard. Several of these run under Linux, or are being ported.

7.3.1 RPlay

The "rplay" package implements the RPLAY and RPTP protocols for supporting both local and remote (over a network) sound control. It comes with the sound server, some sample applications, a library of support routines, and patches for some common applications to add support for rplay.

The applications include the Open Look virtual window manager (olvwm) and the games XTank, XPilot, Crossfire, cbzone, and XBoing.

As well as Linux, Rplay works on a number of common Unix platforms. It was developed by Mark Boyns (`boyns@sdsu.edu`).

7.3.2 NetAudio

Network Audio is a device independent protocol for supporting audio over a network. It uses a similar model to the X window system (and works well with X). It was developed by Network Computing Devices Corporation, and is freely available. Implementations exist for Sun and SGI workstations and NCD X terminals.

Some applications that support NetAudio are the games xboing, xpilot, and the Sun workstation version of the commercial game SimCity.

You can get more information on NetAudio by anonymous FTP from `ftp.x.org` in the directory `/contrib/netaudio` or by sending email to `netaudio-request@ncd.com`

Work is currently underway to port NetAudio to Linux and FreeBSD.

7.3.3 AudioFile

Another network sound system is Digital Equipment Corporation's AudioFile. It is supported on Digital RISC systems running Ultrix, Digital Alpha AXP systems running OSF/1, Sun SPARCs, and SGI AL-capable systems (e.g. Indigo, Indy). The source kit is located at ftp site `crl.dec.com` in the directory `/pub/DEC/AF`.

As far as I know, no one is currently working on a Linux port.

7.3.4 Mtalk/Mtalkf/Atalk/Ztalk

It's possible for two computer users to communicate using sound cards and a "chat" program that transfers audio over a network (TCP/IP, SLIP, or even over the internet). Several similar experimental programs have appeared recently; each use slightly different compression schemes and protocols. Due to hardware limitations, audio is one way at a time (making it more like CB radio than a telephone). All of these packages are ALPHA releases.

"mtalk" was written by `misch@elara.fsag.de`. It uses 2-bit sampling and runs over TCP (it also supports running over "term"). Using compression, it transfers 8000 sample/sec μ-law encoded audio at a data rate of about 1 kilobyte per second.

"mtalkf" is a modified version of mtalk by Scott Doty (`scott@cs.santarosa.edu`). It supports higher bandwith audio.

"atalk", written by Thomas Boutell (`boutell@netcom.com`), was inspired by mtalk. It uses adaptive delta modulation and allows selecting from several levels of sound quality. It notifies you of incoming calls and tells you who to answer, and is supported under Linux or Sun workstations.

"ztalk" incorporates GSM encoding, which is a European standard for lossy speech compression using RPE/LTP (residual pulse excitation/long term prediction). This is said to provide good compression and speech quality, but is more CPU intensive (you may need a math coprocessor). This implementation is by Scott Doty (`scott@cs.santarosa.edu`), using GSM library code by Jutta Degener and Carsten Bormann of Technische Universitaet Berlin (`toast@tub.cs.tu-berlin.de`).

W. Richard Jhang (`feinmann@cs.mcgill.ca`) has extended ztalk to add an address book, alias, and on-line help facility.

Liem Bahneman (`roland@cac.washington.edu`) and Andy Burnett (`burnett@baldrick.cecer.army.mil`) have added an X graphical user interface to ztalk and called it "xztalk".

At the time of writing, these tools are changing rapidly (on a daily basis). Over time, some standardization should occur and the applications should become more reliable.

7.4 Applications for Speech Synthesis

Speech synthesis has been a topic of research for a number of years; some of this has resulted in freely available software which has been ported to Linux.

7.4.1 Speak

This is a Linux port of a speech synthesizer program for Sun and NeXT workstations. It uses an English text to phoneme conversion algorithm and sampled phonemes (.au files). It consists of the program "scat" which converts its input into speech, and a library of speech synthesis routines that can be called from application programs.

I found it to be fast and the text to phoneme algorithm is quite good, but because it uses a small number of fixed phonemes, the resulting speech is very difficult to understand. You may find it amusing. Speak was originally written by John A. Wasser (`WASSER%VIKING.DEC@decwrl.ARPA`) and ported to Linux by Rob Hooft (`hooft@EMBL-Heidelberg.DE`).

7.4.2 Rsynth

This package uses a sophisticated algorithm that models the human vocal tract, as well as an English text to speech dictionary.

I found the resulting speech was quite understandable, but the sound generation was quite slow – on my 386DX40 system, it took almost 2 minutes of processing before it could speak "Hello, world" at an 8kHz sampling rate. It should be considerably faster on a system with a math coprocessor. This package was written by Nick Ing-Simmons (`nicki@lobby.ti.com`) with changes to run under Linux by Rob Hooft (`hooft@embl-Heidelberg.DE`).

7.4.3 OGI Speech Tools

This package contains tools to do speech processing (e.g. graphical display, neural networks for speech recognition, vector quantization, and a library of often used functions). It was developed at the Center for Spoken Language Understanding (`tools@cse.ogi.edu`). The Linux package was put together by Tilo Schuerer (`tilo@cs.TU-Berlin.DE`)

The LSM entry states "If you are into computational linguistics, or are interested in speech compression, recognition, or applications of neural networks, this is one great research package to check out. The documentation in this package consists of man pages, and TeX source. Bugs in the tools lyre and auto_lyre have been fixed from the previous version".

7.4.4 Linus Speaks

While not speech *synthesis* per se, another useful application for your sound card is answering the ever popular question "How do you pronounce *Linux*"? Two 8 KHz sound samples of Linus Torvalds speaking in both English and Swedish can be uploaded from these archive sites:

```
ftp.funet.fi:/pub/OS/Linux/PEOPLE/Linus/SillySounds
ftp.cs.helsinki.fi:/pub/Software/Linux/Kernel/SillySounds
```

7.5 Games

Who can forget another important application for sound – games! Here are a few that run (or will run) under Linux.

7.5.1 Bdash

This is loosely based on the game BoulderDash[2]. You negotiate a maze of rocks, trying to collect diamonds, without being crushed by rocks or killed by other obstacles such as water or beetles. It uses the svgalib library for 256 colour graphics and supports digitized sounds. You can design your own levels using data files. Bdash was written by Karsten Ballueder (`kballued@charon.physik.uni-osnabrueck.de`).

7.5.2 Xboing

XBoing is a blockout type game where you have a paddle which you use to bounce a ball around the game area, blowing up blocks with the ball. You win by obtaining points for each block destroyed and each level completed.

It runs under X. A colour display with a screen resolution of at least 1024x768 is recommended. The author is Justin C. Kibell (`jck@citri.edu.au`).

7.5.3 DOOM

Rumour has it that this popular game is being ported to Linux (it currently runs under MS-DOS). This is an action game with impressive 3-dimensional graphics (some go so far as to call it virtual reality). DOOM is a commercial product sold by id software, with a freely available demo version.

[2] I remember playing this on my Apple //e computer

Chapter 8

Future Directions

The VoxWare sound drivers are still evolving. Hannu Savolainen is working on a version 3.0 release that will have more features. Some of the future capabilities that are planned (or rumoured) are:

- implementing MIDI bus support (`/dev/midi`)
- patch manager support
- sound driver documentation (Hacker's Guide)
- support for more sound cards
- miscellaneous bug fixes

Because the VoxWare drivers are supported on other Intel operating systems (Interactive, SCO, BSD, and SVR4), it should not be difficult to write applications that are portable across all of these platforms. Perhaps this will create a large enough market to stimulate the development of some powerful and unique sound applications.

Appendix A

References

There are many documents freely available from internet archive sites that provide more information on specific issues related to sound and music. Three of the major Linux archive sites on the internet are `sunsite.unc.edu`, `tsx-11.mit.edu`, and `nic.funet.fi`.

The following FAQ (Frequently Asked Question) documents can all be found on the site `rtfm.mit.edu` in the directory `/pub/usenet/news.answers`:

- Generic PC Soundcard FAQ (`PCsoundcards/generic-faq`)
- Comp.sys.ibm.pc.soundcard FAQ (`PCsoundcards/soundcard-faq`)
- Gravis Ultrasound FAQ (`PCsoundcards/gravis-ultrasound/faq`)
- Audio file format descriptions (`audio-fmts/part1,part2`)
- Alt.CD-ROM FAQ (`cdrom-faq`)

The following Usenet news groups discuss issues relevant to sound:

- `alt.binaries.sounds.misc` (Digitized sounds and software)
- `alt.binaries.sounds.d` (Discussion and follow-up group)
- `alt.binaries.multimedia` (Multimedia sounds and software)
- `alt.sb.programmer` (SoundBlaster programming topics)
- `comp.multimedia` (Multimedia topics)
- `comp.music` (Computer music theory and research)

If you have internet mail access, you can subscribe to the SOUND channel of the Linux Activists mailing list. To find out how to join the mailing list, send mail to `linux-activists-request@joker.cs.hut.fi`.

The Soundsite Newsletter is an electronic publication produced every two months and available by anonymous FTP from `sound.usach.cl` in the directory `/pub/sound/newsletters`. The editor is David Komatsu (`davek@uhunix.uhcc.hawaii.edu`).

The README files included with the kernel sound driver source code contain useful information about the latest sound card drivers. These can typically be found in the directory `/usr/src/linux/drivers/sound`.

A good introduction to MIDI is "The USENET MIDI Primer" by Bob McQueer. You can get instructions on how to obtain this and other MIDI files by sending email to `listserv@auvm.bitnet`. The official MIDI Specification is available for a nominal fee from the International MIDI Association[1]:

International Midi Association
5316 West 57th Street
Los Angeles, CA 90056
(415) 321-MIDI

Information on MPEG, including a FAQ, can be obtained on the internet from the sites `ftp.cs.tu-berlin.de` or `quepasa.cs.tu-berlin.de` in the directory `/pub/msdos/dos/graphics`.

Information on the Internet Underground Music Archive can be obtained by anonymous FTP from `sunsite.unc.edu` in the directory `/pub/electronic-publications/IUMA`.

You can obtain the latest information on Internet Talk Radio by sending mail to these addresses:

- `info@radio.com` for general information
- `announce-request@radio.com` to subscribe to the mailing list
- `sites@radio.com` for a list of known FTP sites
- `questions@radio.com` to reach a human

A large archive of Sun audio files (8 KHz 8-bit μ-law encoded) is maintained at the University of California at San Diego, hostname `sounds.sdsu.edu`, in the directory `/sounds`.

The Linux Software Map (LSM) is an invaluable reference for locating Linux software. Searching the LSM for keywords such as "sound" is a good way to identify applications related to sound hardware. The LSM can be found on various anonymous FTP sites, including `sunsite.unc.edu` in the file `/pub/Linux/docs/LSM.gz`. The map is maintained by Jeff Kopmanis (`jeffk@cyberspace.org`).

The Linux Documentation Project (LDP) has produced several free books on Linux covering such issues as installation, networking, and the internals of the Linux kernel. These are available from internet archive sites. Linux HOWTO documents are shorter manuals covering specific issues; there is a Linux Sound HOWTO. The coordinator of the project is Matt Welsh (`mdw@sunsite.unc.edu`).

[1] There are also some unofficial MIDI specifications floating around on internet archive sites.

Hannu Savolainen is working on a Hacker's Guide to VoxWare. This will describe the VoxWare application program interface and the internals of the sound driver. A draft of this document is currently available from Linux archive sites (draft 2, dated February 21, 1994). Other than the kernel source code, this is the only reference for the sound driver API.

Note that at the time this book was written, the sound driver was at version 2.4, and was included as part of the Linux kernel version 1.0.

Appendix B

Glossary of Terms

ADC - Analog to Digital Converter; an electronic device that converts signals from analog to digital form

Analog - electronic signals that can take on continuously varying values

API - Application Program Interface

CD - Compact Disc

CD-I - Compact Disk Interactive

CD-ROM - Compact Disk Read-Only Memory

Companding - *comp*ressing/exp*anding*; a method of reducing the number of bits needed to store digital samples by using a modified logarithmic scale

DAC - Digital to Analog Converter; an electronic device for converting signals from digital to analog form

dB - decibel; a logarithmic unit of measurement

DBS - Direct Broadcast Satellite

Digital - electronic signals that can only take one of a number of discrete values

DSP - Digital Signal Processor

FAQ - Frequently Asked Questions

Fizzbin - a card game played on the Star Trek episode "A Piece of the Action"; also the name of my home Linux system

FM - Frequency Modulation; a technique for synthesizing sounds

GUS - Gravis Ultrasound

HDTV - High Definition Television

ISO - International Organization for Standardization

LDP - Linux Documentation Project

MIDI - Musical Instrument Digital Interface

Mixer - An electronic circuit that combines two or more analog signals

MPU-401 - An industry standard "smart" interface card for the MIDI bus

MOD - a sound file format originally developed for the Amiga computer

MPEG - Motion Picture Experts Group

Operator - a sine wave oscillator used for FM synthesis

OPL-2 - A 2 operator FM synthesizer integrated circuit

OPL-3 - A 4 operator FM synthesizer integrated circuit

PAS16 - Pro Audio Spectrum 16

Patch - a set of parameters defining the setup of an FM synthesizer

PCM - Pulse Code Modulation; an encoding scheme that represents analog waveforms as digital pulses; commonly used in telecommunications

SB - SoundBlaster

Voice - an independent sound generator; the number of voices defines the number of simultaneous sounds or notes that can be played

VoxWare - the kernel sound card drivers used by the Linux operating system

YM3812 - An FM synthesizer integrated circuit (same as OPL-3)

Index