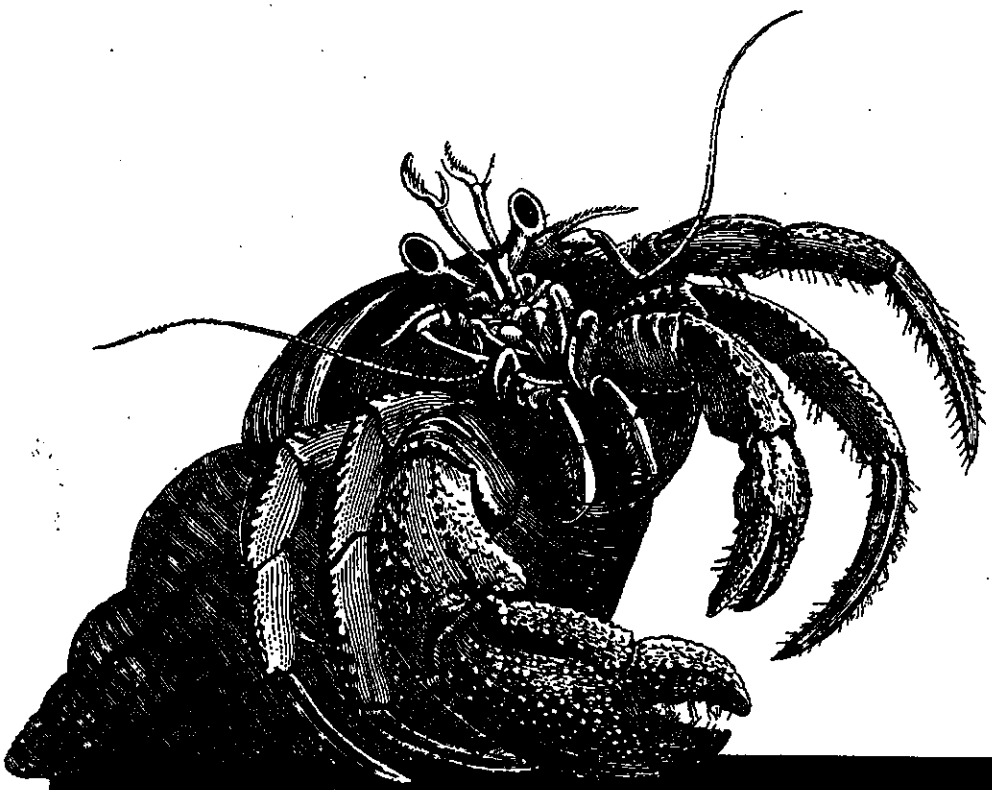
Mastering the MP3 Audio Experience



MP3

The Definitive Guide

O'REILLY®

Scot Hacker

MP3

The Definitive Guide

MP3

The Definitive Guide

Scot Hacker

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

MP3: The Definitive Guide
by Scot Hacker

Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

Editor: Simon Hayes

Production Editor: Maureen Dempsey

Cover Designer: Hanna Dyer

Printing History:

March 2000: First Edition.

Library of Congress CIP data can be found at *http://www.oreilly.com/catalog/mp3/.*

ISBN: 1-56592-661-7
[M]

Table of Contents

Preface

This book has a simple premise: People want to build MP3 collections of the music they like and respect. To do justice to that music requires that the MP3 files constituting a personal music collection be of a high audio quality. But MP3 is generally considered to be a convenience format, not an audiophile format—its main advantages are its flexibility and its portability.

While the press generally refers to MP3 audio as being "near CD quality," audiophiles often point to anomalies in the fidelity of the typical MP3 download. But there's a big difference between the average MP3 file downloaded from the Internet and a file you encode yourself, at a decent bitrate, from your own source material, using the encoder you feel yields the highest quality. MP3 is very much capable of achieving CD quality—you just have to pay a little attention to the variables. As I began to research the MP3 scene in earnest, I found that only a small fraction of available resources were paying close attention to MP3 quality issues. As a hobbyist audiophile, I found this dissatisfying, and felt that it was important to provide readers with a "no-compromise" approach to MP3—you *can* have your convenience factors and a quality audio experience at the same time.

While this book provides plenty of introductory material that will coach any reader through the basic mechanics of MPEG audio, it puts quite a bit of emphasis on fidelity issues, in addition to some of the peripheral topics not covered in depth in other books and online resources. Beyond the basics, for example, we'll be taking a close look at the many legal issues surrounding the MP3 scene, the challenges of building your own MP3 playback hardware, the technical details involved in setting up your own MP3 streaming server, and more.

It was also important to me that this book not be overly Windows-centric. Microsoft Windows may be king in terms of both the number of users and the number of MP3 applications available, but I'm not convinced it's the best possible

MP3 playback and creation platform, for reasons we'll go into elsewhere in the book. The number of MacOS users is increasing once again, Linux use is rising at an incredible clip, and BeOS is highly optimized for media content creation and consumption, with lots of built-in MP3-specific goodies. Accordingly, I've tried to balance coverage of non-Windows operating systems evenly throughout this book. Even if you use only one operating system, I hope you'll find reading about some of the alternative approaches illuminating.

It practically goes without saying that the amount and variety of available MP3 playback and creation software is growing at an incredible rate, as are the number of MP3 hardware options available. I don't pretend to have covered everything available in this book, and plenty of new applications and gear not covered here will undoubtedly be available by the time you read this. I've tried to structure the coverage of available products with an eye toward concepts, rather than specifics, so that the provided coverage will (hopefully) be applicable even to products that have yet to be invented. Please regard the coverage in this book, even where application-specific, as a guide to MP3 creation and playback principles in general.

MP3 is a truly amazing codec and a great feat of engineering. In conjunction with the huge array of "peripheral" technologies and tools available, MP3 has single-handedly ushered in a new era of file-based digital music distribution. It is my hope that this book will help you get the most out of the codec and its surrounding technology, so you can get back down to what this is all supposed to be about: enjoying the music you love.

Conventions in This Book

The following typographical conventions are used in this book:

`Constant width`
: Indicates command-line elements, computer output, and code examples.

Italic
: Introduces new terms and URLs, commands, file extensions, filenames, directory or folder names, and UNC pathnames.

Indicates a tip, suggestion, or general note. For example, we'll tell you how to increase performance or save space, or we'll list links to useful web sites.

Indicates a warning or caution. For example, we'll warn you about easy-to-overwrite traps, crucial plug-ins you should not delete, or where it is important to re-encode your material.

How to Contact Us

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
(800) 998-9938 (in the U.S. or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (FAX)

You can also send us messages electronically. To be put on the mailing list or request a catalog, send email to:

info@oreilly.com

To ask technical questions or comment on the book, send email to:

bookquestions@oreilly.com

We have a web site for the book, where we'll list examples, errata, and any plans for future editions. You can access this page at:

http://www.oreilly.com/catalog/mp3/

For more information about this book and others, see the O'Reilly web site:

http://www.oreilly.com

Acknowledgments

As with any book of this scope, I did not work alone. I am much indebted to my editor, Simon Hayes, for helping to get this project off the ground and for his guidance in structuring and shaping this book in the "big picture." I am also most grateful for the many contributions made by our pool of technical editors:

- mp3tech.org's Gabriel Bouvigne, who possesses a nearly encyclopedic knowledge of MPEG's technical arcana and who helped to flesh out the details of this book in numerous ways.
- Lifelong audiophile Mike Knapp, who can build high-end amplifiers in his sleep and who contributed immeasurably to Hi-Fi issues throughout the book.
- Bruno Prior, who literally built a house around an extensive room-to-room home MP3 network, and who seems to have used every MP3 tool on the planet extensively. Prior also contributed much on the topic of encoding from analog sources.
- MP3.com's "High Geek" Sander van Zoest, who offered much behind the scenes information, especially on the broadcasting and streaming side of things, and who turned me on to MP3 products and technologies before they happened.

In addition, I'd like to thank the members of the WinAmp, mp3stereo, SHOUTcast, and icecast mailing lists, as well as the community inhabiting various MP3 USENET groups. The following individuals have also offered assistance: John Hedtke, Malcolm Humes, Michael James, Henry Kingman, Bruce Lash, Marco Nelissen, Peter Urbanec, Rob Voisey, and Franc Zijderveld.

This book is dedicated to my wife, Amy Kubes, who cheerfully put up with the endless stream of music (both good and bad) flowing from my office over the past year, and for her unwavering support during the course of this project.

In this chapter:
- MP3 Basics
- Rights, Piracy, and Politics

1

The Nuts and Bolts of MP3

In April of 1999, the term "MP3" surpassed "sex" as the most-searched-on term at some of the Internet's top search engines—a phenomenal achievement for a complicated digital music encoding algorithm devised over the course of a decade by a few scientists and audiophiles in an obscure German laboratory.

What is it about MP3 that inspires such unprecedented levels of enthusiasm? For some, it's the prospect of being able to store vast quantities of music on a computer's hard drive, and to shuffle and rearrange tracks from that collection around at a moment's notice. For others, it's the promise of an entirely new model for the music universe—one that allows creative artists to publish their own work without the assistance of the established industry. But for millions of users, the thrill of MP3 is more simple than that: it's the possibility of getting their hands on piles of high-quality music, free of charge.

In this chapter, we'll get a bird's-eye view of the format and the MP3 phenomenon: what it is, how it works, how to download and create MP3 files, and how to listen to them. Then we'll take a look at some of the many issues surrounding MP3, including piracy, politics, digital rights, and the recording industry's stance on the matter. Finally, we'll examine the correlation between the MP3 and open source software movements, and find out why file-based digital music distribution is here to stay.

MP3 Basics

If you're new to the MP3 game, you'll want to know exactly what MP3 files are, where to get them, how they work, and how to make the most of a growing MP3 collection. As you read through this brief overview, keep in mind that these topics are covered in much greater detail elsewhere in this book.

What Is MP3?

Simply put, MP3 is an audio compression technique. Raw audio files—such as those extracted from an audio CD—are very large, consuming around 10 MB of storage space per minute. But MP3 files representing the same audio material may consume only 1 MB of space per minute while still retaining an acceptable level of quality. By drastically reducing the size of digital audio files, it has become feasible for music lovers to transfer songs over the Internet, for users to build enormous digital music collections on their hard drives, to play them back in any order at any time, and to move them around between different types of playback hardware. These possibilities have far-reaching ramifications not just for music lovers, but for artists and the recording industry as well. We'll explore the politics and philosophical issues raised by MP3 in the second part of this chapter.

Why the term "MP3?"

"MP3" is the quick way of referring to an encoding algorithm called "MPEG-1, Layer III," developed primarily by a German technology group called Fraunhofer and Thomson and now officially codified by the International Standards Organization, or ISO. The name, of course, corresponds to the extension found on MP3 files: *After_the_Goldrush.mp3*, for example. More on Fraunhofer and Co. can be found in the section "About the Codec," later in this chapter.

Small is beautiful: How MP3 works

Raw audio does not compress well via traditional techniques: if you try to zip up a WAV file, for instance, you'll find that the resulting archive is only marginally smaller than the uncompressed original.

MP3 takes a different tack on the compression problem. Rather than just seeking out redundancies like zip does, MP3 provides a means of analyzing patterns in an audio stream and comparing them to models of human hearing and perception. Also unlike zip compression, MP3 actually discards huge amounts of information, preserving only the data absolutely necessary to reproduce an intelligible signal. The amount of data preserved is configurable by the person doing the compressing, so an optimal balance between file size and quality can be achieved. The tool or software used to achieve the compression is called an "encoder," while the playback software is called a "decoder" or, more simply, an "MP3 player."

By running uncompressed audio files through an MP3 encoder, files can shrink to around one-tenth of their original size, while still retaining most of their quality. By compressing a little less (to around one-eighth of the original size), MP3 quality can be virtually indistinguishable from that of the original source material. As a result, a three-minute song can be transformed into a 3 MB file, which is something most people can find room for on their hard drives, and that most web

surfers can download in a reasonable time frame. In other words, a 640 MB compact disc stuffed full of MP3 files rather than uncompressed audio can store around 10–11 hours worth of music. And since DVDs store around eight times as much as compact discs, a recordable DVD could hold nearly five days worth of continuous music on a single 5" platter.

The mechanics of the MP3 codec and perceptual encoding principles can be found in Chapter 2, *How MP3 Works: Inside the Codec.*

Working with MP3 Files

If you know how to download files from the Internet, have a grasp of basic file management concepts, and aren't afraid to experiment with new applications, you can probably get started on your own MP3 collection without much coaching. However, there are a lot of options and considerations to take into account, including the quality and efficiency of MP3 encoders and players, advanced features and functions, techniques used for organizing and customizing large MP3 collections, and so on. We've dedicated all of Chapter 3, *Getting and Playing MP3 Files*, and Chapter 4, *Playlists, Tags, and Skins: MP3 Options*, to these topics. For now, here's a brief tour of the basics.

Downloading MP3s

In order to start playing MP3 files, you'll need to get your hands on some, of course. There are two ways to do this: You can either download MP3s that other people have created, or you can create them from the music you already own.

Before you start downloading MP3s, you should know that the vast majority of files available out there are distributed illegally. Many people encode music they legally own, and then make it available on the Internet to people who do not own that music, which is illegal (see Chapter 7, *The Not-So-Fine-Print: Legal Bits and Pieces*, for more information). Whether you choose to download pirated music is a moral choice that only you can make. The wide availability of pirated music, however, should not stop you from seeking out legal MP3s. While there are far fewer of these available, you'll be surprised by the quality of the gems you'll find hiding out in the haystacks. A great place to find legal MP3s is *MP3.com*, though that site is certainly not the only source of legitimate files. If you use a commercial MP3 tool like RealJukebox (Chapter 3), you'll probably find a button or link in the interface that will take you directly to an MP3 download site.

Finding MP3 files

While most users start out by simply typing "MP3" into their favorite search engine, that probably isn't the most efficient way of going about things. You might want to start instead at a major site dedicated to indexing or distributing MP3 files, such as *mp3.lycos.com*, *www.listen.com*, *www.scour.net*, or *www.rioport.com*. Search engines can, however, be very useful for finding smaller sites run by individuals—but be prepared to encounter lots of broken links and unresponsive sites. Because many user-run sites are quickly shut down by Internet Service Providers (ISPs) under pressure from record labels, search engines often index links to sites that no longer exist.

The Web isn't the only way to find MP3 files—you'll also find plenty of files on FTP servers, in binary Usenet groups, and in IRC channels. Details on using these venues for MP3 downloading can be found in Chapters 3 and 4.

Users looking to swap MP3 files easily with music fans all over the world may want to check out Napster (*www.napster.com*), which is a sort of combined IRC, FTP, and search client with a twist. Rather than searching the Web, you'll be searching the hard drives of other Napster users for songs you like. Since you'll only see files on the systems of people currently using the service, you won't have to worry about broken links and downed servers. Log in to the Napster server, register your collection with a specific genre, and you'll be able to search for files on other people's systems by song name or artist. Find a song or songs you like and transfer them to your hard drive, while other people do the same with your music collection. Meanwhile, you can chat with other music lovers in the background as your transfer proceeds. Great idea, but the potential for copyright abuse inherent in this product is extreme, and none of the music we found during testing was legitimate. Nevertheless, Napster has single-handedly ushered in a whole new era of user-to-user file sharing, and has the music industry more worried than ever.

Creating your own MP3 files

Creating your own MP3s is only slightly more difficult than downloading them, but the payoff is worth it. You know for a fact that the music in your collection is the music you like, you can personally control the quality of the encodings, and you don't have to worry about whether any of your tracks are illegal.

Encoding tracks from your CD collection is a two-step process. First, bits from an audio CD must be transferred to your system as uncompressed audio, typically as a WAV file. This extraction process is known as *ripping*. The uncompressed audio is then run through an MP3 encoder to create an MP3 file. However, there are

dozens of tools available that take care of all the hard work behind the scenes, ripping and encoding transparently in a single step. You'll meet a handful of ripper/encoder combination tools in Chapter 5, *Ripping and Encoding: Creating MP3 Files.*

Playback basics

Think of an MP3 file like any other document you might store on your computer and open in an application. You can open a document by using an application's File → Open menu, by double-clicking a document icon, or by dragging a document onto the application's icon. MP3 files are no different, and can typically be played in any of these ways. There are hundreds of MP3 players available for virtually all operating systems, and all of them are capable of playing all MP3 files. As a user, you have tons of options when it comes to picking your tools. In Chapter 3, you'll meet some of the most popular MP3 players available for Windows, MacOS, Linux, and BeOS, and be introduced to the fundamental principles of MP3 playback.

Playlists

One of the most liberating aspects of working with file-based music (as opposed to music stored on media such as CDs, tapes, or LPs) is the fact that you suddenly gain the ability to organize, randomize, and mix the tunes in your music collection in an infinitude of ways. If you've ever created custom mixed-music cassette tapes, you know how fun—and how time consuming—this can be. MP3 playlists let you enjoy the fun part while skipping the time-consuming part.

The vast majority of MP3 players include a "playlist" window or editor, into which you can drag any random collection of tracks. Any playlist can be saved for posterity, to be played again at a later date. A playlist can be as short as a single song or as long as your entire collection (some people have playlists referencing months of nonrepeating music). A playlist can reference all the music in a folder or an entire directory structure, or can be composed by querying your system for all songs matching a certain criteria. For example, you can create playlists of all country music written prior to 1965, or all of your acid jazz tracks, or all of your schmaltzy disco. Playlist creation and manipulation is covered in detail in Chapter 4.

Playlists are simple text files listing references to the actual locations of MP3 files on your system or on a network. As such, they consume almost no disk space. Because playlists reference songs on your system, it is usually not useful to trade them with other users. There are, however, playlists comprised only of URLs to MP3 files on the Internet, and these will, of course, work on anyone's system.

ID3 tags

MP3 files are capable of storing a certain amount of "meta-data"—extra information about each file—inside the file itself. Data on track title, artist, album, year, genre, and your personal comments on the track can all be stored in an MP3 file's *ID3 tags*. These tags will be inserted automatically by most tools as you rip and encode, or can be added or edited later on, often directly through your MP3 player's interface. ID3 tags become more important as your collection grows, especially when you start using database-oriented MP3 organizers, as described in Chapter 4.

Internet radio

Some people have neither the time nor the inclination to create and manage a huge MP3 collection. Fortunately, they don't have to. Thanks to the rise of outfits like SHOUTcast (*www.shoutcast.com*) and icecast (*www.icecast.org*), thousands of users are *streaming* MP3 audio from their computers to the Internet at large, running live broadcasts much like a radio station. There are several key differences between MP3 downloads and MP3 streaming:

- MP3 broadcasts aren't saved to the listener's hard disk, unlike MP3 downloads. When you tune in to a broadcast, the only thing that's saved to disk is a tiny text file containing some meta-data about the broadcast in question, including the server's address and a playlist. This file is passed to the MP3 player, which in turn receives and handles (buffers) the ongoing broadcast.
- Broadcasts are synchronous, while downloads are asynchronous. In other words, when you tune in to a broadcast, you hear exactly what's being played from a given server at that moment in time, just like the radio. When you download a file, you get to listen to it any time you want.
- Because of bandwidth constraints on most listeners, broadcasts are typically of a lower fidelity than MP3 downloads. MP3 broadcast servers usually send out MP3s that have either been down-sampled to a lower frequency, encoded at a lower-than-normal bitrate, or sent as a mono rather than stereo stream.

Full details on tuning in to MP3 broadcasts can be found in Chapter 4. The process of running your own Internet radio station is described in Chapter 8, *Webcasting and Servers: Internet Distribution.*

Beyond the computer

While you'll almost certainly create all of your MP3 files on your computer, and will most likely begin your MP3 explorations by playing them back through your computer as well, part of the magic of file-based digital audio is the flexibility. There's no reason an MP3 file can't be transferred to any device that includes a

storage and playback mechanism. And sure enough, a whole new class of devices has arisen to meet this need: portable units similar to the classic Sony Walkman but geared for MP3 playback, rather than tape or CD, are becoming hugely popular. Meanwhile, we're beginning to see the emergence of a whole new range of home stereo MP3 components, capable of storing gigabytes of digital audio and being operated just like any other home stereo component. Of course, the technology is being applied to car stereos as well. Even hand-held computers such as the Handspring Visor are gaining MP3 playback capabilities.

Users with some technical know-how and a soldering iron are hacking out techniques for building MP3 playback hardware of their own, free from SDMI and other security mechanisms (see Chapter 7 for more about MP3 security and legal issues) that ultimately limit the functionality of commercial MP3 hardware. Chapter 6, *Hardware, Portables, Home Stereos, and Kits*, includes comparative analysis of MP3 portables, an early look at a few MP3 home stereo components, and introduces the concepts of building your own MP3 hardware from scratch.

About the Codec

So, what exactly is MPEG audio compression, and MP3 specifically? Technically, that's a bit of a long story, so we'll go into great detail on that in Chapter 2. You don't need to know how MP3 works in order to start playing with it, but to shed a little light on the subject now, MPEG audio compression is a "psychoacoustic" technique that exploits various limitations in both the human ear and the mind's ability to process certain kinds of sounds at very high resolutions. MPEG encoders store "maps" of human auditory perception in a table, and compare an incoming bitstream to those maps. The person doing the encoding gets to specify how many bits per second will be allocated to storing the final product. Taking note of that restriction, the encoder does its best to strip away as much data as possible (within the specified data storage limitation, or "bitrate") while still retaining the maximum possible audio quality. The more bits per second the user allows, the better described the final output will be, and the larger the resulting file. With fewer bits per second, the user will get a smaller file (better compression), and a corresponding decrease in audio quality. Again, we'll go into the process in greater detail in Chapter 2.

The MPEG family

MPEG is not a single standard, but rather a "family" of standards defined by the Moving Picture Experts Group, which was formed in 1998 to arrive at a single compression format for digital audio and avoid a standards war between various competing technologies. All of the MPEG standards are used for the coding of audio-visual data into compressed formats.

Coding in this sense of the word refers to the process of running a stream of bits through an algorithm, or set of rules. *Encoding* is the process of taking an uncompressed bitstream and running it through the algorithm to generate a compressed bitstream or file. *Decoding* is, naturally, the opposite—taking a compressed bitstream and turning it into an uncompressed file or an audible signal. The term *codec* is short for compressor/decompressor,* and refers to any algorithm capable of performing this bidirectional function.

The MPEG family is broken down into major classes (MPEG-1, MPEG-2, MPEG-4), which are further broken down into sub-classifications called *layers*. Each major class and layer is optimized for specific real-world applications, such as compressed movie soundtracks, broadcast, or file-based musical coding. Each successive layer is more complex than the preceding layer. For example, a layer III decoder will be 2.5 times more complex than a layer I decoder. The MPEG "layers" are described in sub-documents of each class, with audio coding schemes described in a document labeled "ISO/IEC11172-3." The MPEG coding technique that interests us in this book is MPEG-1/MPEG-2 Layer III, referred to throughout this book simply as "MP3."

Technically, MPEG-1 Layer III and MPEG-2 Layer III are both referred to as MP3, as are the rather obscure MPEG 2.5 extensions. MPEG-1 Layer III is used for 32, 44.1, and 48kHz sampling rates, while MPEG-2 Layer III is for 16, 22.05, and 24kHz sampling rates. The MPEG 2.5 extensions allow for 8 and 11kHz. MP3 players can play any of these, and the specs are very similar.† The vast majority of files you'll encounter in the wild are simple MPEG-1 Layer III.

Do not confuse MPEG-1 Layer III (MP3) with MPEG-3—there is no such animal. There was once an MPEG-3 classification in development, which was intended to address high-quality video. However, MPEG-2 was shown to deliver sufficiently high quality, so MPEG-3 was conjoined with the existing MPEG-2 specification. The spec now skips from MPEG-2 to MPEG-4.

* In some circles, the term stands for enCOder/DECoder, though this interpretation has lost favor to compressor/decompressor.

† MPEG-2 also allows for multichannel extensions of up to five channels, though few people have ever actually seen this in action. Multichannel efforts are concentrated on MPEG-4, covered in Chapter 9, *Competing Codecs and Other File Formats.*

The MP3 patent

The fact that the MP3 spec is maintained by the MPEG Working Group doesn't mean they invented the technology. The working group merely codifies standards to guarantee interoperability between various applications, operating systems, and implementations. One of the very first tasks of the working group was to circumscribe the conditions of the ownership of intellectual property under the umbrella of international standards. Their conclusion was that patented technologies are allowed to be codified as standards, but that those patents must be fairly and equitably licensable to all comers, so that no single company could gain a monopoly on a specific audio/video compression technology.

The MP3 codec itself was devised by the Fraunhofer Institute of Germany and Thomson Multimedia SA of France (referred to throughout this book simply as "Fraunhofer"), who originally published the standard in 1993.* Fraunhofer and Co. own the intellectual copyright on any technology capable of creating "an MP3-compliant bitstream." While Fraunhofer publishes low-grade sample code that can be used as a basis for more sophisticated MP3 coding tools, Fraunhofer still requires developers of MP3 encoders to pay hefty licensing fees (full details on that can be found in Chapter 5).

To learn more about the MPEG working group and MPEG specifications in general, there is no better starting point than *www.mpeg.org*. To learn more about Fraunhofer and MP3 licensing issues, see *www.iis.fhg.de*. The official web site of the MPEG Consortium is *drogo.cselt.stet.it/mpeg/*.

Rights, Piracy, and Politics

The flexibility and portability of MP3 has left the recording industry wondering where to turn, unsigned musicians newly empowered, signed artists with mixed reactions, and fans making out like bandits. The debate centers on a quest for the right balance between exploiting the promotional power of this new medium and protecting the intellectual copyright of artists and labels.

MP3's Impact on the Recording Industry

In July of 1999, the International Federation of Phonographic Industries (IFPI) estimated that around three million tracks were downloaded from the Internet every day, most of them without the permission of their copyright holders. The Recording Industry Association of America (RIAA) claims to have lost as much as $10 billion through music piracy in 1998. It's not just record company executives and

* Fraunhofer did not work alone; other companies and organizations (notably AT&T) contributed to the development of the encoder as well.

artists who stand to lose; the digital music revolution has implications for everyone in the channel: record store owners, CD pressing plants, and even truck drivers. Of course, most signed artists resent having their intellectual property illegally distributed as well. Well-known artists ask the RIAA every day to clamp down on pirate sites hosting their music (although it's also the case that many signed artists are much more supportive of MP3 than are their labels). In the rest of this chapter, we'll take a look at some of the many difficult issues currently being faced by the industry and music lovers alike, and take a look at some of the techniques the industry is proposing to deal with the situation. The legal nitty-gritty of MP3 is discussed in more detail in Chapter 7.

File-based digital audio changes the game

For nearly a century, the record industry has held the distribution of musical content in a hammerlock. If you wanted to own music, you had to do it on their terms, purchasing music distributed on the media they had officially blessed, and only through their approved channels. While the industry's stranglehold on music distribution slipped for the first time in the 1950s with the advent of reel-to-reel tape decks, and even more in the '70s with the popularization of the cassette tape, tape technologies had a major Achilles' heel: analog copies always lose a little quality as successive copies are made—a third-generation copy of a well-recorded LP doesn't sound so well-recorded anymore. In addition, the person making the copy is burdened with having to create a new physical instance for each person to whom she wants to distribute her tunes.

I don't have to tell you that MP3 has changed all of that. Digital copies (of anything) are virtually bit-perfect, so no quality is lost in successive generations—a 74th-generation copy sounds every bit as good as the original. And then there's the Internet. Because digital music can be file-based rather than media-based, a single file representing any kind of content can be placed on a web or FTP server and made available to the entire world at once. The burden of making physical copies, which naturally limited the rampancy of tape-based copies to a large extent, has vanished.

But until recently, there's been another "gating factor" that has limited the spread of file-based audio distribution: size. While it's always been possible to rip an audio track from a compact disc and make it available on the Internet, doing so was impractical because uncompressed audio consumes around 10 MB per minute of storage space. Few people had the available bandwidth or storage space to be whipping 30 MB pop songs around.

If all of us had Internet connections with unlimited bandwidth and hard drives large enough to store terabytes or petabytes of data, MP3 would be unnecessary. Limited bandwidth and small hard drives are the only reason MP3 even exists (or, at least, the only reasons it's become popular). Ironically, the MP3 phenomenon hit at a time when both of these issues were being addressed at a rapid clip. More and more, people are having DSL or cable modem connections installed in their homes, and 36 GB hard disks are available for a few hundred bucks at this writing. If the trend continues, and there's no reason to think that it won't, one can almost imagine audio compression in general becoming obsolete due to a lack of demand. But for the foreseeable future, limited bandwidth and modest hard drives are a reality, so digital audio compression is a necessity.

If you can't beat 'em, join 'em

Will the recording industry be able to put an end to MP3? If not, how will it cope with the phenomenon? There are several factors at work here. While people commonly claim that the cassette revolution had little impact on the industry, the truth is that, for whatever reason, sales today aren't as great as they were in the '70s. Of course, there are several reasons for this, such as the fact that CDs cost twice what LPs used to cost, and the fact that we no longer seem to have anything like the giant mega-stars of the '70s. Pink Floyd's "Dark Side of the Moon" stayed in Billboard's Top 100 for the better part of a *decade*. While we still have stars, the days of the Beatles and the Stones are, most likely, gone for good. Of course, the industry feels that illegal MP3 downloads are making these problems worse.

But a 1999 report from industry analyst Jupiter Communications concluded that only three percent of consumers would be purchasing downloaded music by the year 2003. While the industry has good reason to be concerned, some see that last factoid as a wake-up call to the industry—either adapt to a world in which downloadable copyrighted music is a reality, or be left out altogether. If this projection turns out to be true, the industry will either have to find a way to crush unprotected MP3 distribution (unlikely), or accept the fact that MP3s and other digital audio files will continue to be distributed without hope of a significant revenue return.

In the short term, the RIAA launched a campaign to start shutting down pirate sites. For the most part, this has consisted of cease-and-desist letters being sent to Internet service providers, warning them to remove illegal files from users' sites or face prosecution (see Chapters 7 and 8). ISPs are generally quick to oblige (and are bound by law to do so). But even with the best lawyers in the land at their

disposal, the industry has found that trying to crush pirate sites in the midst of a phenomenon this large is like playing a vast and endless game of Whack-a-Mole—eliminate one site and six others spring up in its place. Even with the cooperation of ISPs, the task is futile and the industry knows it, though they are still obliged to continue trying.

Beat 'em: The Secure Digital Music Initiative (SDMI). In the face of this apparent futility, the industry has decided to approach the problem from another angle: Chop it off at the knees. Since the sprawling and largely ungovernable Internet cannot be easily controlled, the industry has decided to work with its partners and make it harder to create copy-able digital music files to begin with. By colluding with the makers of compact discs, sound cards, and software vendors, and by embedding special codes into newly created CDs, the industry hopes to get as many people using copy protection-enabled equipment and source material as possible.* According to a quote made by SDMI Executive Chairman Dr. Leonardo Chiariglione in *Billboard* magazine:

> You will be able to play your MP3 files on the portable devices of today, but at a certain point in time, which may happen quite soon, the record companies will start embedding some signals into their future content so that it can become playable only on SDMI devices.

As you'll see throughout this book, it is logically and practically impossible to create a 100% secure system, since savvy users can always trap music as it's heading out of the computer and toward the sound card. The industry's goal, then, is to create a fence high enough that the vast majority of users will lack the technical know-how or wherewithal to try and jump over it. This plan, known as the Strategic Digital Music Initiative (SDMI), is already much-delayed in its implementation at this writing, and is, many feel, doomed for failure. Only time will tell. More on SDMI can be found in Chapters 6 and 7.

Join 'em: MP3 and electronic commerce. If illegal MP3 distribution cannot be stopped or even significantly curtailed, then the industry may learn to embrace the new paradigm and accept the fact that its role is changing in the digital world. Major labels will most likely come to appreciate the "buzz" effect that can be created by posting tracks on their own web sites. Releasing a few good tracks from an upcoming album by a big-name star is likely to result in more sales and more word-of-mouth. Already, we're beginning to see the first glimmerings of this phenomenon, as major stars such as Tom Petty, Billy Idol, The Grateful Dead, and Alanis Morissette embrace the format.

---

* Embedded codes in music, known as *watermarks*, are created such that their presence persists even when transformed from one format to another, even when "jacked" out of an audio port.

The industry faces a major obstacle in persuading hardware vendors to collude with them in making computers secure against music piracy: Consumers *want* the ability to copy music freely. If one hardware vendor adopts SDMI and another does not, many consumers are simply going to buy equipment from the one who does not. That's quite a disincentive to vendors considering implementing SDMI in hardware, and without global and enforced legislation (unlikely), the industry faces an uphill battle. Nevertheless, Sony has already developed a pair of technologies, called MagicGate (for use in devices) and OpenMG (for use in PCs), that guarantee that a signal can be moved from one place to another, but not copied. Of course, Sony is also a record label, so their desire to move on this front is understandable. But unless they can get other vendors to adopt the same or similar technology, savvy consumers will simply avoid these products.

But the industry doesn't have to *give* away MP3 files. What's wrong with selling them? One can imagine a future in which fans can log into a label's site and download tracks for a buck a pop, selecting only the songs worth purchasing and (happily) ignoring the duds. Whether this is managed via micropayments or ongoing accounts with labels, there's a big problem here: Once a fan has purchased a file, what's to stop him from hacking it into an unprotected version, placing that file on his own site, and making it available for download—enabling infinite copying of the unprotected version to the rest of the world? SDMI and audio formats with built-in security (see Chapter 9) will help here, but again, it only takes one user to break or somehow get around the security mechanism, and the file is once again in the clear and released into the wild. Right or wrong, most people are going to download the free, unprotected version rather than the 99¢ version when given a choice. One begins to appreciate the industry's dilemma.

So, if the industry decides to go for it and start selling music online in a big way, how will they do it? First of all, due to its lack of security, MP3 is very badly suited for the job. Online sales of digital music are likely to come in some other file format, and more likely in several of them. No big deal—the flexibility of software makes it easy for users to store lots of formats on their hard drives, and probably even to play them all back through the same player. Regardless, you can practically rest assured that whatever formats are used will be SDMI-compliant. Taking that as a given, here are the models:

Micropayments

Because credit card charges always incur a transaction fee, low-cost items (such as individual songs) are more awkward to sell online than are sweaters or beer. The notion of micropayments is that users maintain an account with

an independent organization. Users may load up the account with money in advance, as with a phone card, and charge small purchases against it. Record labels may also run their own micropayment schemes.

CD distribution

Because the production of compact discs is relatively inexpensive, users may be able to order up a bunch of songs at once and have them pressed to an audio CD, which will then be mailed to the user. Because the cost will be higher, standard credit card transactions will be feasible. This model, in fact, has already been used by companies such as Liquid Audio (*www.liquidaudio.com*), though they have not managed to reach a significant cross-section of the population. Such a model run by a major label would likely have more success.

Subscription

In this model, users would pay a flat monthly or yearly fee and be entitled to download the latest hits from a variety of artists selected by the user or the label, in whichever format the label chooses to work.

Temptation

Perhaps the simplest model, and the one most akin to the shareware model computer users are familiar with, is to simply give away a track or two from upcoming albums, with the hope that users will like the tracks enough to purchase the entire CD, either online or from a local record store.

Advertising

Controversial for good reason, the MP3 advertising model is an extension of the banner ads on web pages with which we're all familiar. Each downloaded MP3 file comes with a brief advertisement embedded into the first few seconds, which is the first thing the user hears. While this is essentially similar to what we get on the radio every day, most users dislike the notion of having their personal MP3 collections riddled with ads. Nevertheless, this model is already in use by Amp3.com, who has managed to sign a number of big-name artists on to the program.

"Guilt sites"

This rather odd idea, which has been tossed around by many people in many forums, has yet to see a working model. The concept is that many users want to enjoy the free trade and exchange of MP3 files, but still want to pay a royalty back to the labels and artists who made the music possible. "Guilt sites" would allow people to check in anonymously and say, "I've decided to actually keep and listen to 24 of the 113 files I downloaded this month. Here's a list of the songs. Please accept my $24 in payment." A fascinating idea, but one which the recording industry is unlikely to bless.

Is MP3 Legal?

One of the questions most frequently asked by people concerned about MP3 piracy is, "Why not just make MP3 illegal?" By making MP3 illegal, they reason, it would be easy to arrest music pirates and deter others from taking part in the ongoing plundering of the traditional music business. There are several problems with this line of thinking:

- MP3 has lots of legitimate uses, and is intended for legitimate use. The mere fact that it can also be used for illegitimate purposes does not in and of itself provide sufficient reason to outlaw it.
- MP3 is a codec. Nothing more, nothing less. Nothing about MP3 is inherently dangerous. You can use a crowbar to break into a drug store, chicken manure to make bombs, or a laptop computer to crack security systems, but that doesn't mean any of those things should be illegal. You can kill yourself or others by drinking too much alcohol, swinging a machete in the wrong vector, or by eating too much bacon, but that doesn't mean those things should be illegal.
- If we were to make MP3 illegal, we would also have to outlaw all other computer file formats, from *.DOC* to *.JPEG*, since all of them can be used to store and distribute intellectual property illegally as well.

Short answer: Yes, MP3 is legal—it's just an innocent codec. Issues surrounding the protection of intellectual and creative copyright are completely separate from the mechanism of distribution. However, MP3 differs from other audio compression techniques; it provides no built-in means to prevent unrestricted copying. This is an issue the industry is struggling to solve, but MP3 itself is but one player in a larger problem, not the problem itself.

The Artist's Turn

Great talent does not automatically end up at the top of the music business. All over the world, millions of talented, creative artists are playing music for their friends, or in small clubs, or even on tour, trying to make a name for themselves. You think that a talented artist will automatically end up with a recording contract? Think again. The record business isn't necessarily looking for talent... but that's another story.

Unless you're scouring the local newspaper and heading off to local clubs night after night looking for something fresh, you may never hear thousands of great musicians and songwriters. Unless an artist is being spotlighted by the recording industry, mass exposure is almost impossible for an artist to get.

The next wave of self-publishing

The MP3 revolution addresses this concern head-on. Because any musician with a computer can encode their own songs to MP3, they can potentially expose themselves to the world at large without ever having to sign a recording contract. They can go directly to the people, bypassing the industry as we know it. This possibility is analogous to the great desktop publishing revolution of the '80s and the web publishing revolution of the 1990s—suddenly, everyone has the ability to publish their own music (no matter how bad or how good) without help from "the man."

Unsurprisingly, this—perhaps the single most liberating and legitimate aspect of the MP3 revolution—is seldom if ever mentioned by record industry executives when talking about piracy problems. An artist signed to a label generally does not have the right to post his own songs to the Internet, because the label owns the copyright to those songs. But an independent, unsigned artist can do whatever she likes with her own tunes. And while that represents a lesser threat to the industry, it represents a threat nonetheless; the prospect of a burgeoning "industry" that runs itself, outside of the RIAA's purvey, is at hand. And unlike piracy issues, there isn't a thing the RIAA can do about the rise of self-publishing musicians. Sites like MP3.com represent this concept in its full glory. Rather than requiring users to surf the Web for scattered, legitimate downloads by unsigned artists, users can search through archives of thousands of unsigned artists all in one place.

While MP3.com is focused on and dedicated to the promotion of unsigned artists, the site does indeed enjoy an arrangement with The American Society of Composers, Authors, and Publishers (ASCAP), which allows MP3.com to stream (not offer for download) the works of musicians signed to labels. MP3.com's relationship with ASCAP is outlined at *www.mp3.com/ascap/*.

Do the math: A good deal for artists

There's more than just exposure in it for the artist. Let's do the math. According to one artist who's been struggling to make it in the business for years, a band who sells 150,000 CDs through a medium-size label will still not be generating profit for themselves—yes, there's that much overhead in being associated with a label.* Furthermore, only around 5% of signed artists end up turning a profit—signing with a label is not necessarily a "ticket to ride." But if an artist presses his or her own CDs, and sells them for $10 each over the Internet, they can make a profit by

* It is for reasons such as this that the relatively successful group TLC was forced to file for bankruptcy in 1999.

selling only 15,000 copies. This artist would make $5 per CD in profit, netting $75,000 in sales. And, in fact, this is exactly what MP3.com allows artists to do, using their D.A.M. CD service (see Chapter 8 for details).

Presumption of Guilt

There's another, possibly more frightening aspect to the rise of self-publishing: Anti-piracy measures instituted by the industry—such as SDMI—may have a negative impact on the rise of this new distribution mechanism. As you'll see in more detail in Chapter 7, SDMI and similar mechanisms operate on a presumption of guilt. Because so much music in MP3 format is pirated, security mechanisms tend to make the assumption that anything in MP3 format is probably pirated. This is nothing but a slap in the face to the unsigned, unrepresented artist using the MP3 codec for legitimate purposes, to willfully and legally publish his own music. If security formats, technologies, and hardware are pushed on users by the industry, unsigned artists will be left right where they were before MP3 promised them an alternative.

Legitimate uses of MP3 are not constrained to music. One doesn't have to look too far to find other creative, legal, and legitimate uses of the technology. *The New York Times* makes spoken-word versions of its news available for download from its site. MP3LiT.com features hundreds of published and unpublished authors reading excerpts from their books, going on political rants, reading or improvising poetry, and offering self-help material, all in MP3 format. These and other legitimate uses of MP3 will be adversely affected by any attempt by the record industry to squash unprotected MP3 files in general.

Why MP3 Is Here to Stay

As you've probably heard, MP3 isn't the only digital audio compression format out there. As you'll see in Chapter 9, there are numerous alternatives to the format available to the public. In fact, many of them achieve even better compression ratios and/or better sound quality than MP3.

Not the best, but good enough

So why hasn't the public moved toward competing formats in a wild rush? Even MP3's most staunch supporters readily admit that MP3 is a less-than-ideal solution from an audiophile perspective. But like Betamax or the Macintosh, history proves that the best technology doesn't always win. The VQF and AAC formats are great examples of this in action: both offer superior quality, and often, smaller file sizes than MP3 as well.* Nevertheless, MP3 is entrenched—you might say that MP3 is to

* Many MP3 players also handle VQF and other audio formats; you can switch between them seamlessly.

the Internet audio industry as Windows is to the operating system market—functional and workable, but not the ideal solution, even though the depth of its penetration practically guarantees that it isn't going anywhere anytime soon.

More often than not, standards are set by the technology that establishes a strong user base first. And establishing that user base has as much to do with marketing, wide availability, and ease-of-use as it has to do with quality. And history also shows that a superior technology doesn't just have to be a little better in order to turn heads—it has to be way, way better. While some of MP3's competing formats are better than MP3, the fact of the matter is that MP3 is good enough. At decent bitrates, it sounds great, and the compression ratios are perfectly acceptable to most people. While some people make the point that the public will migrate to another technology as soon as something better comes along, this is not proving to be the case. Excellent alternatives do exist, but they're not taking hold (at least, they're not as of early 2000). For another format to surpass the momentum of MP3, it would have to have all of the following attributes:

- Smaller file sizes
- Superior audio quality
- Be free and unprotected

Ironically, it's that third point that makes all the difference. The first and second points are already satisfied by other file formats and codecs. But the public wants what the public wants, and they have spoken very clearly: they want lots of choice, available source code, and freedom from copying restrictions. The fact that this also means a huge headache when it comes to protecting intellectual property is another matter.

The source, Luke, use the source

Another important key to MP3's popularity is the fact that it's highly available. Rather than hanging from the awnings of a single company, literally hundreds of MP3 encoders and players are available for virtually every operating system under the sun. And, of course, there are millions of MP3 files out there. Most people don't have the time or the inclination to evaluate all the competing formats—they go with what works.

But perhaps most importantly, Fraunhofer and Thomson have made sample code available to the world. As you'll see in Chapter 7, that doesn't mean the MPEG codec is open source exactly (technically, developers still owe licensing fees to the codecs copyright holders), but it does mean that it's been possible for competition to take root and for a multiplicity of approaches to flourish. From the most com-

plex and exacting command-line applications to dirt-simple, all-in-one GUI applications, a quick search of any MP3-related site will uncover as many approaches to MP3 encoding and playback as there are types of computer users. That kind of diversity has a way of generating a deep, grassroots kind of spirit around a technology—one need look no further than the amazing success of Linux to witness the kind of results that can arise in an open, "biologically diverse" environment.

Among the contenders to MP3 is Microsoft's own digital audio compression format, Windows Media. To be sure, Windows Media does deliver good sound at much smaller file sizes than MP3 (at the lower bitrates).* Windows Media offers security features for artists and labels who want to take advantage of it. It's built-in streaming features make it competitive with MP3 and RealAudio simultaneously. But the real kicker, of course, is the fact that Windows Media is built into later versions of Windows, giving it a level of direct access with which no one can hope to compete. But at this writing, Windows Media was not gaining any kind of serious buy-in from users (though that doesn't mean it never will). Think about that: if not even Microsoft can break into a market it desperately wants to occupy, then it's a pretty safe bet MP3 will be sticking around for a good long time. MP3 is in a position relative to digital audio compression that Microsoft itself is in relative to operating systems—it got in on the game early and established a strong user base, making it the standard. And standards have a way of remaining standards (imagine what would happen if you purchased a new toaster and it came with an electrical plug that didn't fit your wall sockets). Windows Media and other competitive codecs are covered in Chapter 9.

"Open source" music

To be sure, there are a handful of bands who have adopted the "open source" approach to music. The Grateful Dead is probably the most famous example of this force in motion. For decades, attendees at Dead shows were not patted down for hidden microphones; rather, taping of live shows was actively encouraged by the band and its management. The Grateful Dead realized that when you encourage a meme to spread and make your fans feel like they're an integral part of the process, the fans become loyal to the family; the Dead always sold out their concert venues. While industry execs still fear that giving things away means cannibalizing their own business, the important lesson is that if what you give away is good, then people want more of it, not less.

* Though the consensus is that the format creates noticeable artifacts, especially with certain types of sounds—see Chapter 9 for more information.

Today, with the Dead a thing of the past, this spirit lives on at sites like *www.deadabase.com*, where users can trade MP3 versions of all those live shows. The only restriction placed on people downloading tracks is that they aren't allowed to profit from them, even by selling ad space on their sites. In subsequent years, bands such as Phish, Metallica, the Black Crowes, and the Dave Matthews Band have all jumped on the "open source music" bandwagon, although this always applies to tapes of live shows rather than of albums.

Comparison between the MP3 movement and the open source software movement doesn't stop there. Some argue that the entire history of music is open-source-like, noting that Homer's *Iliad* and *Odyssey* were composed collectively over the years by "skilled Greek emcees," and that the modern practice of resampling, remixing, and audio collage forms a perfect audio analog to the open source movement.*

Give the people what they want

Finally, there's the fact that MP3 is unprotected. Ironically, it is this fact that labels and signed artists fear the most—any MP3 file can be duplicated a bazillion times, traveling from one user to the next like an unrestrained virus. Right or wrong, millions of MP3 users are interested in the technology simply because it's so easy to get "free" music. While the record industry struggles to introduce protected formats that will allow us only to copy our own music, for our own use, the public apparently wants just the opposite—a simple format that works the way they expect the rest of our computer documents to work, not one that imposes restrictions and limitations on who can get at the contents of a particular file, and on what machine.

Protected audio compression formats are going to make some headway, because the industry is going to make sure that they do. But there are too many people who appreciate MP3 precisely for its openness for it ever to go away. While some songs may eventually become available only in protected formats, people will continue to use available MP3 tools.

Biting the hand that feeds

The question is, does the freedom afforded by MP3 ultimately threaten the very industry it so enjoys? The industry certainly made this argument at the dawn of cassette tapes, and again at the dawn of video tapes, only to be proven wrong for the most part. But as mentioned earlier, infinite digital copying really is a horse of a different color, and this time around, the industry has a right to be afraid. But is that really such a bad thing? MP3 may force the industry to change the way it does

* Julian Dibbell, "You Say You Want a Revolution?," *www.mp3.com/news/258.html*.

business. Labels may ultimately *have* to give a few tunes away in order to sell an album. They may at some point *have* to start selling songs one tune at a time rather than as complete albums—the whole concept of what constitutes an "album," which is currently an artifact of the amount of recording time that can be fit onto a vinyl LP or CD, may be permanently affected. They may even be forced to lower the unjustifiably high prices they currently charge for compact discs.

In short, MP3 and related formats may force the record industry to downsize its approach to the whole game, and to come to grips with the fact that they can't fight this phenomenon forever. For its part in the debate, the watchdog organization Electronic Freedom Foundation (*www.eff.org*), has stated that:

> ...architecture is policy. Given the choice, consumers would choose to purchase music in open formats. We believe that artists who allowed their works to be distributed in open formats would gain competitive advantages over artists who locked up their work.

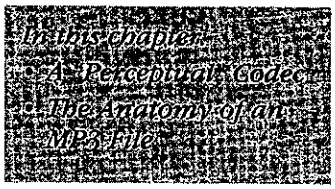
The EFF's fear is that if the Big 5 record companies (Sony Music Entertainment, EMI Recorded Music, Universal Music Group, BMG Entertainment, and Warner Music Group) are allowed to monopolize the trade in online music, independent artists will be stymied just as their big opportunity appears on the horizon.

Can't we all just get along?

The record industry is not going to simply go away. They'll still be around to promote artists, to manage tours, to print and sell t-shirts and bumper stickers, and to run the web sites of their artists. As one representative put it, "Without the help of the industry, the independent artist's experience may be closer to that of a street performer, rather than a stage performer." It is, after all, possible that a marketing machine more powerful than MTV itself may rise like a Phoenix from the chaos of the Internet.

The industry has yet to feel the full impact of MP3, but they will—it's inevitable. The genie is already out of the bottle.

2



How MP3 Works: Inside the Codec

So what's the trick? How does the MP3 format accomplish its radical feats of compression and decompression, while still managing to maintain an acceptable level of fidelity to the original source material? The process may seem like magic, but it isn't. The entire MP3 phenomenon is made possible by the confluence of several distinct but interrelated elements: A few simple insights into the nature of human psychoacoustics, a whole lot of number crunching, and conformance to a tightly specified format for encoding and decoding audio into compact bitstreams. In this chapter, we'll take a look at these elements in detail in order to understand exactly what's going on behind the scenes of MP3 encoding and decoding software, as well as some of the chicanery that takes place between your ears.

Note that this chapter goes fairly deeply behind the scenes of MP3, and is somewhat technical in nature. You can skip this chapter if you're not interested in learning how MP3 works. If you just want to get started creating and playing MP3 audio, you can skip ahead to Chapters 3, 4, and 5.

A "Perceptual" Codec

Well-encoded MP3 files can sound pretty darn good, considering how small they are. As mentioned in Chapter 1, *The Nuts and Bolts of MP3*, your typical MP3 file is around one-tenth the size of the corresponding uncompressed audio source. How is this accomplished? That's a somewhat complex topic, so we've devoted this entire chapter to explaining the process.

MPEG Audio Compression in a Nutshell

Uncompressed audio, such as that found on CDs, stores more data than your brain can actually process. For example, if two notes are very similar and very close

together, your brain may perceive only one of them. If two sounds are very different but one is much louder than the other, your brain may never perceive the quieter signal. And of course your ears are more sensitive to some frequencies than others. The study of these auditory phenomena is called *psychoacoustics*, and quite a lot is known about the process; so much so that it can be quite accurately described in tables and charts, and in mathematical models representing human hearing patterns.

MP3 encoding tools (see Chapter 5, *Ripping and Encoding: Creating MP3 Files*, for examples and usage details) analyze incoming source signal, break it down into mathematical patterns, and compare these patterns to psychoacoustic models stored in the encoder itself. The encoder can then discard most of the data that doesn't match the stored models, keeping that which does. The person doing the encoding can specify how many bits should be allotted to storing each second of music, which in effect sets a "tolerance" level—the lower the data storage allotment, the more data will be discarded, and the worse the resulting music will sound. The process is actually quite a bit more complex than that, and we'll go into more detail later on. This kind of compression is called *lossy*, because data is lost in the process. However, a second compression run is also made, which shrinks the remaining data even more via more traditional means (similar to the familiar "zip" compression process).

MP3 files are composed of a series of very short *frames*, one after another, much like a filmstrip. Each frame of data is preceded by a *header* that contains extra information about the data to come. In some encodings, these frames may interact with one another. For example, if one frame has leftover storage space and the next frame doesn't have enough, they may team up for optimal results.

At the beginning or end of an MP3 file, extra information about the file itself, such as the name of the artist, the track title, the name of the album from which the track came, the recording year, genre, and personal comments may be stored. This is called "ID3" data, and will become increasingly useful as your collection grows. We'll look at the structure of MP3 files and their ID3 tags in this chapter, and the process of creating and using ID3 tags in Chapter 4, *Playlists, Tags, and Skins: MP3 Options*. Let's zoom in for a closer look at the entire process.

Always remember to set your encoder to store ID3 data during the encode process, if possible—doing so will save you a lot of work down the road.

Waveforms and Psychoacoustics

Everything is vibration. The universe is made of waves, and all waves oscillate at different lengths (a wavelength is defined as the distance between the peak of one wave and the peak of the next). Waves vibrating at different frequencies manifest themselves differently, all the way from the astronomically slow pulsations of the universe itself to the inconceivably fast vibration of matter (and beyond). Somewhere in between these extremes are wavelengths that are perceptible to human beings as light and sound. Just beyond the realms of light and sound are sub- and ultrasonic vibration, the infrared and ultraviolet light spectra, and zillions of other frequencies imperceptible to humans (such as radio and microwave). Our sense organs are tuned only to very narrow bandwidths of vibration in the overall picture. In fact, even our own musical instruments create many vibrational frequencies that are imperceptible to our ears. Frequencies are typically described in units called *Hertz* (Hz), which translates simply as "cycles per second." In general, humans cannot hear frequencies below 20Hz (20 cycles per second), nor above 20kHz (20,000 cycles per second), as shown in Figure 2-1.* While hearing capacities vary from one individual to the next, it's generally true that humans perceive midrange frequencies more strongly than high and low frequencies,† and that sensitivity to higher frequencies diminishes with age and prolonged exposure to loud volumes. In fact, by the time we're adults, most of us can't hear much of anything above 16kHz (although women tend to preserve the ability to hear higher frequencies later into life than do men). The most sensitive range of hearing for most people hovers between 2kHz to 4kHz, a level probably evolutionarily related to the normal range of the human voice, which runs roughly from 500Hz to 2kHz.

These are simple and well-established empirical observations on the human hearing mechanism. However, there's a second piece to this puzzle, which involves the mind itself. Some have postulated‡ that the sane mind functions as a sort of "reducing valve," systematically bringing important information to the fore and sublimating or ignoring superfluous or irrelevant data.§ In fact, it's been estimated that we really only process a *billionth* of the data available to our five senses at any given time. Clearly, one of the most important functions of the mind is to function as a sieve, sifting the most important information out of the incoming signal, leaving the conscious self to focus on the stuff that matters.

---

* Figure 2-1 is based on a chart and data produced by Xing Technology Corporation (*www.xingtech.com*).

† However, note that frequency-dependent sensitivity flattens out the louder the sound is. For more information, see the section on Fletcher-Munson curves in Chapter 4.

‡ Aldous Huxlely, *The Doors of Perception*. First published by Chatto and Windus Ltd., 1954. Now available in numerous reprints.

§ Presumably, one of the distinguishing characteristics of insanity is a failure of the mind to perform its function as a reducing valve, allowing "irrelevant" information to take as much precedence as the relevant.

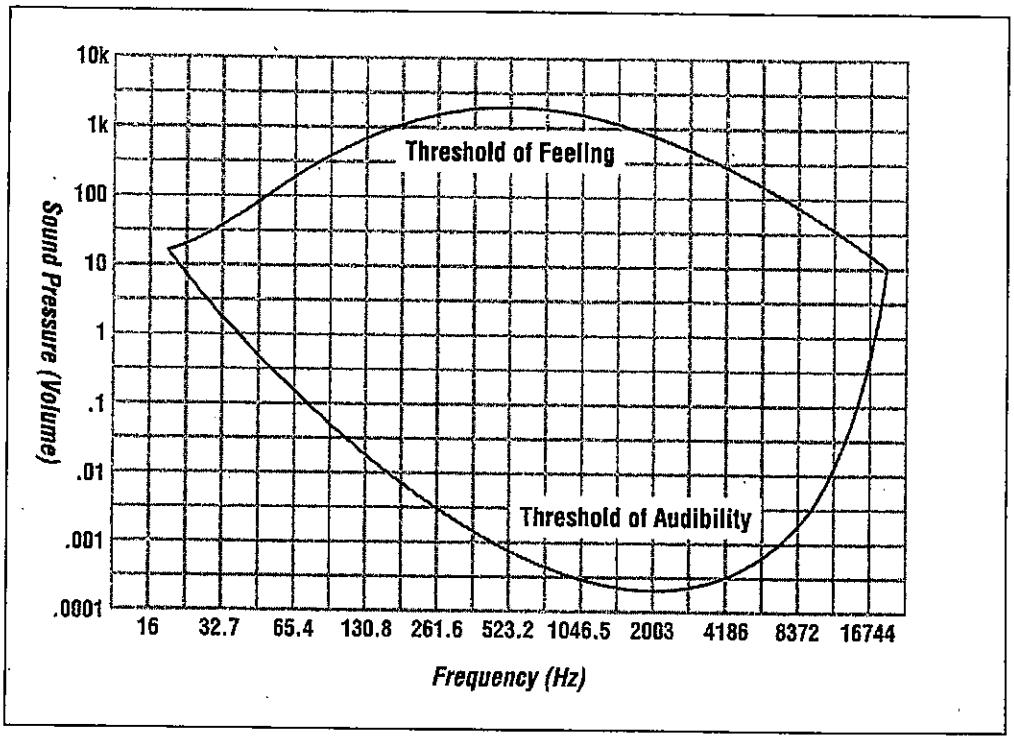


Figure 2-1. While vibratory frequencies extend both above and below, human hearing is pretty much limited to the range between 20Hz and 20kHz

The basic principle of any perceptual codec is that there's little point in storing information that can't be perceived by humans anyway. As obvious as this may sound, you may be surprised to learn that a good recording stores a tremendous amount of audio data that you never even hear, because recording equipment (microphones, guitar pickups, and so on) is sensitive to a broader range of sounds and audio resolutions than is the human ear. After getting an overview of how perceptual codecs work in general, we'll take a closer look at exactly how the MP3 codec does its thing.

The word "codec" is a foreshortening of the words "compress" and "decompress," and refers to any of a class of processes that allow for the systematic compression and decompression of data. While various codecs are fundamental to many file formats and transmission methods (for instance image and video compression formats have their own codecs, some of which are perceptual as well), it's the MP3 codec that concerns us here.

Breaking It Down

MP3 uses two compression techniques to achieve its size reduction ratios over uncompressed audio—one lossy and one lossless. First it throws away what humans can't hear anyway (or at least it makes acceptable compromises), and then it encodes the redundancies to achieve further compression. However, it's the first part of the process that does most of the grunt work, requires most of the complexity, and chiefly concerns us here.

Perceptual codecs are highly complex beasts, and all of them work a little differently. However, the general principles of perceptual coding remain the same from one codec to the next. In brief, the MP3 encoding process can be subdivided into a handful of discrete tasks (not necessarily in this order):

- Break the signal into smaller component pieces called "frames," each typically lasting a fraction of a second. You can think of frames much as you would the frames in a movie film.
- Analyze the signal to determine its "spectral energy distribution." In other words, on the entire spectrum of audible frequencies, find out how the bits will need to be distributed to best account for the audio to be encoded. Because different portions of the frequency spectrum are most efficiently encoded via slight variants of the same algorithm, this step breaks the signal into *sub-bands*, which can be processed independently for optimal results (but note that all sub-bands use the algorithm—they just allocate the number of bits differently, as determined by the encoder).
- The encoding bitrate is taken into account, and the maximum number of bits that can be allocated to each frame is calculated. For instance, if you're encoding at 128 kbps, you have an upper limit on how much data can be stored in each frame (unless you're encoding with variable bitrates, but we'll get to that later). This step determines how much of the available audio data will be stored, and how much will be left on the cutting room floor.
- The frequency spread for each frame is compared to mathematical models of human psychoacoustics, which are stored in the codec as a reference table. From this model, it can be determined which frequencies need to be rendered accurately, since they'll be perceptible to humans, and which ones can be dropped or allocated fewer bits, since we wouldn't be able to hear them anyway. Why store data that can't be heard?
- The bitstream is run through the process of "Huffman coding," which compresses redundant information throughout the sample. The Huffman coding does not work with a psychoacoustic model, but achieves additional compression via more traditional means.* Thus, you can see the entire MP3 encoding

* See the glossary for definition and more information.

process as a two-pass system: First you run all of the psychoacoustic models, discarding data in the process, and then you compress what's left to shrink the storage space required by any redundancies. This second step, the Huffman coding, does not discard any data—it just lets you store what's left in a smaller amount of space.

- The collection of frames is assembled into a serial bitstream, with header information preceding each data frame. The headers contain instructional "metadata" specific to that frame (see "The Anatomy of an MP3 File" in this chapter).

Along the way, many other factors enter into the equation, often as the result of options chosen prior to beginning the encoding (more on those in Chapter 5). In addition, algorithms for the encoding of an individual frame often rely on the results of an encoding for the frames that precede or follow it. The entire process usually includes some degree of simultaneity; the preceding steps are not necessarily run in order. We'll take a deeper look at much of this process in the sections that follow.

Notes on "Lossiness"

Compression formats, whether they operate on audio, video, images, or random collections of files, are either *lossless* or *lossy*. The distinction is simple: Lossless formats are identical to the original(s) after being decompressed, while lossy formats are not. A good example of a lossless compression format is the ubiquitous .zip archiving scheme. When you unpack a zip archive containing a backup of your system from last month, losing even a single byte is unacceptable. However, some types of data can withstand having information thrown away, on the grounds that either you'll never notice what's missing, or you're willing to make a compromise: Smaller files in exchange for missing but unimportant data.

A good example of a lossy compression format is JPEG, which banks on the fact that image files often store more information than necessary to display an image of acceptable quality. By throwing away some of the information, and by encoding redundant information with mathematical algorithms, excellent compression ratios can be achieved for images that don't need to be displayed at high resolutions.

While the JPEG analogy doesn't depict the MP3 compression process accurately, it does illustrate the concept of lossiness, and it's important to understand that all MP3 files, no matter how well-encoded, have discarded some of the information that was stored in the original, uncompressed signal.

Many lossy compression formats work by scanning for redundant data and reducing it to a mathematical depiction which can be "unpacked" later on. Think for a moment of a photograph depicting a clear blue sky, and below it a beach. If you were to scan and store this image on your hard drive, you could end up storing

hundreds of thousands of pixels of perfect blue, all identical to one another, and therefore redundant. The secret of a photographic compression method like GIF is that this redundant information is reduced to a single description. Rather than store all the bits individually, they may be represented as the mathematical equivalent of "repeat blue pixel 273,000 times." When the part of the image depicting the sand is encountered, the sand is analyzed for redundancy and similar reductions can be achieved. This is why simple images can be stored as small files, while complex images don't compress as well—they contain less redundancy. On the other hand, JPEG compression works in accord with user-defined "tolerance thresholds"; determining how similar two adjacent pixels (or, more accurately, frequencies) have to be before they're considered redundant with one another is the key to determining the degree of lossiness. If JPEG compression is set high, light blue and medium blue pixels may be treated as being redundant with one another. If JPEG compression is set low, the codec will be more fussy about determining which pixels are redundant. The end result will be a clearer picture and a larger image file.

Masking Effects

Part of the process of mental filtering, described earlier in this chapter, which occurs unconsciously at every moment for all of us, involves a process called *masking*, and is of much interest to students of psychoacoustics: the study of the interrelation between the ear, the mind, and vibratory audio signal. Two separate masking effects come into play in MP3 encoding: auditory and temporal.

Simultaneous (auditory) masking

The simultaneous masking effect (sometimes referred to as "auditory masking") may be best described by analogy. Think of a bird flying in front of the sun. You see the bird flying in from the left, then it seems to disappear, because the sun's light is so strong in contrast. As it moves past the sun to the right, it becomes visible again. In more concrete audio terms, recall how you can sometimes hear an acoustic guitarist's fingers sliding over the ridged spirals of the guitar strings during quiet passages. Of course, you seldom if ever hear this effect during a full-on rock anthem, because the wall of sound surrounding the guitar all but completely drowns these subtle effects.

The MP3 codec, of course, is unconcerned with guitar stings; all it knows are relative frequencies and volume levels. So, to put simultaneous masking into more concrete terms, let's say you have an audio signal consisting of a perfect sine wave fluctuating at 1,000Hz. Now you introduce a second perfect sine wave, this one fluctuating at a pitch just slightly higher—let's make it 1,100Hz—but also much quieter—say, –10db.* Most humans will not be able to detect the second pitch at

* Decibels are a unit of sound pressure, more commonly known as "volume." The measurement of decibels is always relative, so –10db does not imply something quieter than silence, only that the second tone is quieter than the first.

all. However, the reason the second pitch is inaudible is not just because it's quieter; it's because its frequency is very close (similar) to that of the first. To illustrate this fact, we'll slowly change the frequency (pitch) of the second tone until it's fluctuating at, say, 4,000Hz. However, we'll leave its volume exactly as it was, at –10db. As the second pitch becomes more dissimilar from the first, it becomes more audible, until at a certain point, most humans will hear two distinct tones, one louder than the other, as illustrated in Figure 2-2. At Point A, Tone 2 is barely audible next to Tone 1. At Point B, Tone 2 is quite audible, even though its volume remains unchanged.

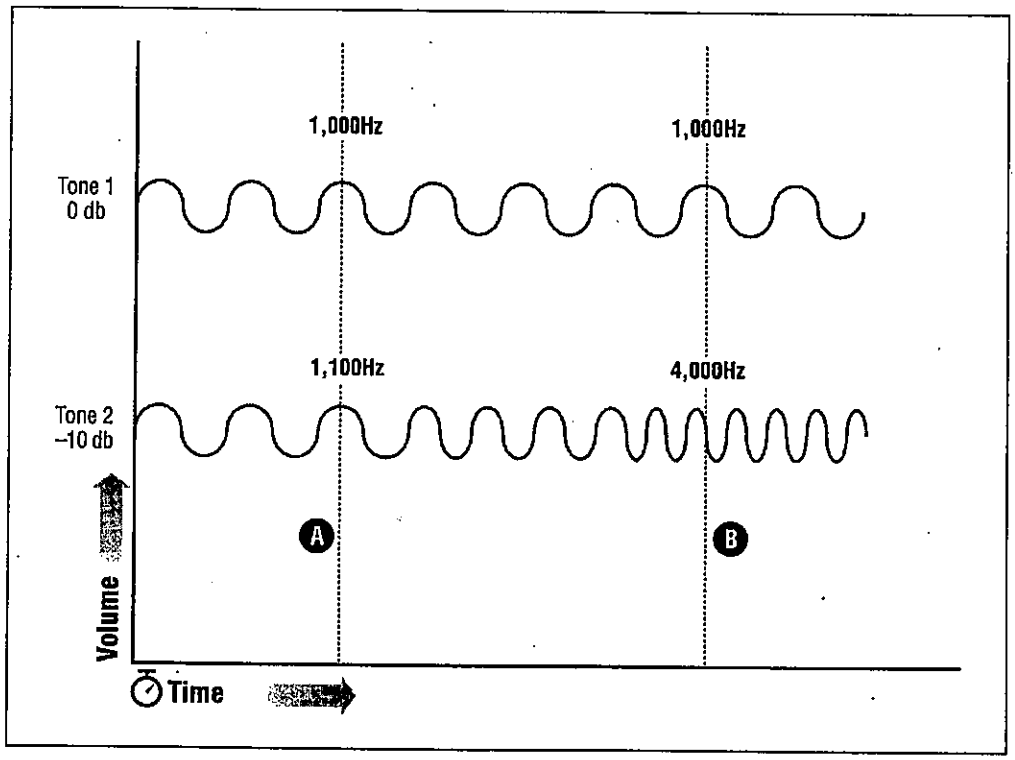


Figure 2-2. As two simultaneous tones become more dissimilar, they become recognizable as separate entities

What's going on here is a psychoacoustic phenomenon called "simultaneous masking," which demonstrates an important aspect of the mind's role in hearing: Any time frequencies are close to one another, we have difficulty perceiving them as unique, much as mountains on the distant horizon may appear to be evenly textured and similarly colored, even while the same mountains might be full of variation and rich flora if one were hiking in them. In effect, we have the aural equivalent of an optical illusion—a trick of our perceptual capacity that contributes to our brain's ability to filter out the less relevant and give focus to stronger elements.

Now consider for a moment the fact that an audio signal consisting of two sine waves—even if one is quieter—contains almost twice as much data as a signal containing a single wave. If you were to try and compress an audio signal containing two sine waves, you would want the ability to devote less disk storage space to the nearly inaudible signal, and more to the dominant signal. And, of course, this is precisely what the algorithms behind most audio compression formats do—they exploit certain aspects of human psychoacoustic phenomena to allocate storage space intelligently. Whereas a raw (waveform or PCM*) audio storage format will use just as much disk space to store a texturally constant passage in a symphonic work as it will for a dynamically textured one, an MP3 file will not. Thus, MP3 and similar audio compression formats are called "perceptual codecs" because they are, in a sense, mathematical descriptions of the limitations of human auditory perception. The MP3 codec is based on perceptual principles but also encapsulates many other factors, such as the number of bits per second allocated to storing the data and the number of channels being stored, i.e., mono, stereo, or in the case of other formats such as AAC or MP3 with MPEG-2 extensions, multichannel audio.

Temporal masking

In addition to auditory masking, which is dependent on the relationship between frequencies and their relative volumes, there's a second sort of masking which also comes into play, based on time rather than on frequency. The idea behind temporal masking is that humans also have trouble hearing distinct sounds that are close to one another in time. For example, if a loud sound and a quiet sound are played simultaneously, you won't be able to hear the quiet sound. If, however, there is sufficient delay between the two sounds, you will hear the second, quieter sound. The key to the success of temporal masking is in determining (quantifying) the length of time between the two tones at which the second tone becomes audible, i.e., significant enough to keep it in the bitstream rather than throwing it away. This distance, or threshold, turns out to be around five milliseconds when working with pure tones, though it varies up and down in accordance with different audio passages.

Of course, this process also works in reverse—you may not hear a quiet tone if it comes directly before a louder one, so premasking and postmasking both occur, and are accounted for in the algorithm.

* Pulse Code Modulation is the standard designator for the digitization of uncompressed audio, such as that found on audio CDs. PCM audio is sampled 8000 times per second at 8 bits, for a total storage consumption of 64 kbps.

For more information on psychoacoustics, read any of the excellent papers on the subject at *www.cpl.umn.edu/auditory.htm.*

Enter Bitrates, Stage Left

While MP3 users cannot control the degree of lossiness specifically, as they might do with a JPEG image, they can control the number of bits per second to be devoted to data storage, which has a similar net result.

In the process of coding, the "irrelevant" portions of the signal are mapped against two factors: a mathematical model of human psychoacoustics (i.e., the masking requirements), and the bitrate, which is established at the time of encoding (see Chapter 5). The bitrate simply refers to the number of bits per second that should be devoted to storing the final product—the higher the bitrate, the greater the audio resolution of the final product, as shown in Figure 2-3. An easy way to visualize the effect of bitrate on audio quality is to think of an old, turn-of-the-century film. Old movies appear herky-jerky to us because fewer frames per second are being displayed,* which means less data is distributed over a given time frame.

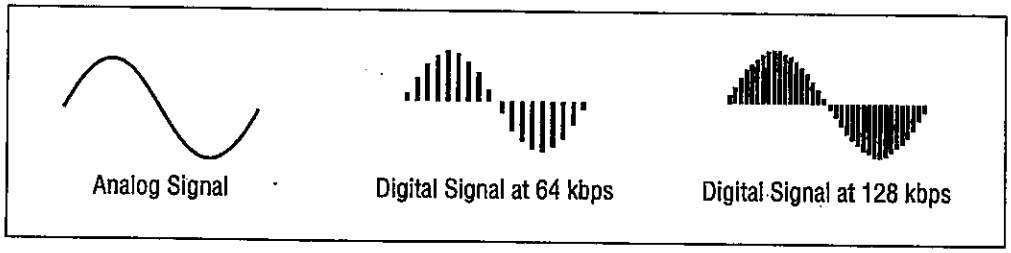


Figure 2-3. More bits per second means more audio resolution, pure and simple

For example, the current de facto standard is to encode MP3 at 128 kbps, or 128,000 bits per second. The codec takes the bitrate into consideration as it writes each frame to the bitstream. If the bitrate is low, the irrelevancy and redundancy criteria will be measured harshly, and more subtlety will be stripped out, resulting in a lower-quality product. If the bitrate is high, the codec will be applied with leniency, and the end result will sound better. Of course, the file size of the end product corresponds directly with the bitrate: If you want small files, you have to settle for less quality. If you don't mind larger files, you can go for higher bitrates.

* And because old movie cameras did not operate at a constant rate, nor was the frame rate accurate with the passage of time in the real world.

Bitrates refer to the total rate for all encoded channels. In other words, a 128 kbps stereo MP3 is equivalent in size and quality to two separate 64 kbps mono files. However, a 128 kbps stereo file will enjoy better quality than two separate 64 kbps mono files, since in a stereo file, bits will be allocated according to the complexity of the channels. In a given time, one channel may utilize 60% of the bits while the other uses only 40%. The cumulative size in bits will, however, remain constant.

CBR vs. VBR

Most of the information you'll read in this book and elsewhere assumes that the bitstream is being encoded at a constant bitrate (CBR). In other words, if you specify a 128 kbps encoding, then that's what you're going to get, start to finish. The drawback to CBR is that most music isn't structured with anything approaching a constant rate. Passages with many instruments or voices are succeeded by passages with few, simplicity follows complexity, and so on. The response to this situation has been the development of variable bitrate (VBR) encoders and decoders, which vary the bitrate in accordance with the dynamics of the signal flowing through each frame. VBR technology was first implemented by Xing, which is now owned by Real Networks, but is now supported by dozens, if not hundreds, of third-party products.

Rather than specifying a bitrate before encoding begins, the user specifies a threshold, or tolerance, when encoding with VBR. All notions of bits per second go right out the window, of course; instead, one selects VBR quality on a variable scale. Confusingly, this scale is represented differently in different encoders. While MusicMatch Jukebox gives you a scale of 1 to 100, the LAME command-line encoder lets you specify a quality of 0 to 9, where the scale represents a distortion ratio. Therefore, you can't just assume that higher numbers mean higher quality—see the documentation for your encoder before proceeding, or run the tests yourself. In any case, the scales are essentially arbitrary; think of them as though you were using a slider to control the overall quality versus file size ratio as you might with a JPEG editor.

While VBR files may achieve smaller file sizes than those encoded in CBR at a roughly equivalent fidelity, they present a number of drawbacks of their own. First, these files may not be playable in older-generation decoders, which had no notion of VBR concepts (although the ISO standard specifies that a player must handle VBR files if it's to be considered ISO-compliant). Second, VBR files may present timing difficulties for decoders. You may expect your MP3 player to display inaccurate timing readouts—or no timing information at all—when playing back VBR files. However, VBR techniques conveniently take some of the guess

work out of trying to find an optimal bitrate for any given track—whereas you might have to encode a file several times with CBR to find the perfect balance, you can just set your encoder to use a relatively high quality level and let the computer figure out an optimal bitrate for each frame automatically.

In general, the header data in most CBR files is same for each frame, while header data is necessarily different for each frame of a VBR file. However, VBR files don't incur more processing power, as all MP3 players read the header data for each frame regardless of whether they're playing a CBR or VBR file.

Bitrates vs. samplerates

Bitrates aren't quite the final arbiter of quality. The resolution of audio signal in general is in large part determined by the number of source samples per second stored in a given format. While bitrates are a measure of the *amount of data stored* for every second of audio, samplerates measure the *frequency with which the signal is stored*, and are measured in kiloHertz, or thousands of samples per second. The standard samplerate of CD audio is 44.1kHz, so this is the default samplerate used by most encoders, and found in most downloadable MP3 files. Audio professionals often work with 48kHz audio (and, more recently, 96kHz*). Digital audio storage of lectures and plain speech is sometimes recorded as low as 8kHz. Streamed MP3 audio is often sent out at half, or even a quarter of the CD rate in order to compensate for slow Internet connection speeds. If you need to minimize storage space, or are planning to run your own Internet radio station, and are willing to sacrifice some quality, you'll want to do some experimenting with various samplerates. More details can be found in Chapter 5.

Note that nothing is ever actually played or heard during the encoding process—you can encode MP3 on a computer with no sound card or speakers, if you need to for some reason. In fact, this is exactly how things are done in some professional organizations, particularly those dedicated to Internet broadcasting (see Chapter 8, *Webcasting and Servers: Internet Distribution*). In such instances, one computer may be used for auditioning and selecting files, a second used for the actual encoding process, and a third dedicated to serving the files to the Internet. Of course, the beefiest machine available will always be used as the encoding machine in such a scenario.

* Generally, stored audio frequencies cannot be higher than half the samplerate, so a 96kHz samplerate allows for storage of frequencies well outside the human 20kHz threshold.

Freedom of Implementation

Interestingly enough, the MP3 specification (ISO 11172-3) does not specify exactly *how* the encoding is to be accomplished. Rather, it outlines techniques and specifies a level of conformance; in other words, it tells developers that their resulting MP3 files must meet certain structural criteria.* This is necessary for the same reason that any standard exists: To allow for the proliferation of MP3 encoders and players by various vendors and developers. The specification only serves to guarantee a baseline consensus in the community regarding how certain things will operate. An encoder developed according to the MP3 specification will be capable of outputting a "compliant bitstream" that can be played successfully with any MP3-compliant decoder, just as you can create a JPEG file in any image editor under any operating system and expect it to display properly in any JPEG-compliant image viewer on any operating system.

It's important to maintain the distinction between the primary developers of the codec itself, The Fraunhofer Institute, and the committee that codified the work of Fraunhofer into the MPEG-I Layer 3 specification, the International Standards Organization (ISO). Standards are often created this way: A company produces a technology, other companies apply to become a part of the standards-creation process, and together they lay down the laws of implementation so that all vendors can compete around that technology. Note, however, that just because MP3 has been standardized by ISO does not mean that Fraunhofer (and their partners Thomson Multimedia) don't still hold the patent on the technology itself. As you'll see in Chapter 7, *The Not-So-Fine-Print: Legal Bits and Pieces*, Fraunhofer's patent is being aggressively exercised, making it difficult for small-time developers to affordably implement the ISO standard.

In any case, while the standard specifies exactly how *de*coding is to be accomplished, it only provides sample implementations (one simple and one complex) for *en*coding. As a result, there's a certain degree of headroom available for developers to make up some of the rules as they go along. In general, encoder developers work toward two goals: speed and quality. While there is some difference in the quality of audio files output by various encoders (as you'll see in Chapter 5), there are vast differences in the speed at which encoders operate. Sometimes, encoding speed comes at a distinct disadvantage to the quality of the resulting bitstream, though this is not necessarily the case.

A good example of the kind of freedom left to developers is the fact that the MP3 standard does not specify exactly how to treat the upper end of the spectrum, above 16kHz. Since human auditory perception begins to diminish greatly (with

* Documentation on the MP3 specification can be ordered for around $150 from *www.iso.ch*.

age and exposure to loud volumes) between 16kHz and 20kHz, some developers have historically chosen to simply chop off frequencies above 16kHz, which can be beneficial at low bitrates, since it leaves more bits available for encoding more audible frequencies. Xing, for example, did this with the first versions of their very fast codec. Later, they rewrote their codec to handle frequencies up to 20kHz (probably at the behest of the audiophile MP3 community).

If you're curious about the upper and lower thresholds of your own hearing, download a sine wave generation program for your platform and run some tests. If you find a graphical program, you can simply turn the dial or drag the slider up the frequency spectrum until you can no longer hear it. If the program works from the command line, you can either generate sweep frequencies or generate a series of files at different frequencies at the upper end of the range and play them in sequence. BeOS and Linux users should check out a utility called sinus, while users of any platform can generate pure tones through any of the many simple synthesizer programs available at your favorite software library. The potential problem with running this kind of test lies in the fact that your playback hardware may itself not be capable of reproducing frequencies above, say, 17kHz. A test like this is best conducted on the highest quality equipment you can find.

Other Considerations

In addition to the general principles outlined in this chapter, the MP3 codec does a lot of additional work maintaining frequency tables, storing and allocating bits optimally, handling user options set at encode time, and the like. While we don't cover everything the encoder is responsible for exhaustively, here are a few of the more important additional chores the encoder must tackle.

Dipping into the reservoir

Because the bitrate is taken into consideration at every time frame, there will inevitably be certain frames of such complexity that they cannot be adequately coded to adhere to the limitations imposed by the chosen bitrate. In such a case, the MP3 spec allows for a "reservoir of bytes," which acts as a sort of overflow buffer when the desired amount of data cannot be stored in the given timeframe. In actual practice, this reservoir is not a separate storage space in the file, but rather the "empty space" left over in frames where the necessary information was encoded into the available space with room to spare. In other words, the byte reservoir is a portion of the algorithm designed to rob Peter and pay Paul.

While the CD and DAT audio formats typically offer 16 bits of resolution, the processing of a very complex musical passage may result in only four or six bits of resolution being encoded into the final bitstream,* since there isn't enough storage space allocated to handle the data needs of each frame. What can't be drawn from the reservoir will simply result in an audible degradation of the signal quality. Thus, the byte reservoir is only a partial solution to the loss of signal quality in complex passages. The only real solution to quality loss is to encode the signal at a higher bitrate.

The joint stereo effect

Most people have had an opportunity at some point to listen to a stereo system with a separate subwoofer attached (in fact, most better-quality computer speaker systems consist of two or four satellite speakers and a separate subwoofer). And as you may have noticed, the placement of satellite speakers is critical to high-quality audio reproduction, whereas the placement of the subwoofer is almost entirely irrelevant—people stuff subwoofers under desks, behind couches, or integrate them with other pieces of living room furniture. The reason it's possible to do this without affecting sound quality is because the human ear is largely insensitive to the location of the source of sounds at the very low and very high ends of the frequency spectrum.

The MP3 spec optionally exploits this aspect of human psychoacoustics as well. A file being encoded in stereo is by definition twice as large as a monophonic file. However, this file size doubling effect can be somewhat mitigated by combining high frequencies across the left and right tracks into a single track. This is done during the encoding phase by selecting the "joint stereo" option in the encoder's preferences, or by passing an appropriate command-line option to the encoder (there are actually several subtle differences between the various joint stereo encoding "modes"—more on that in Chapter 5). Since you might not be able to tell which speaker very high signals are emanating from anyway, there may be no point in storing that data twice.

Some hard-core audiophile tweaks claim that bass sounds are not entirely nondirectional, only that they're less so than mid- and high-frequency sounds. Listeners with ears trained this well are probably not much interested in MP3 to begin with, but those listeners might be able to tell the difference in high-frequency spatialization when comparing MP3 to unencoded audio.

* This isn't necessarily a bad thing; it depends on the complexity of the passage in question.

When the joint stereo option is enabled, a certain amount of "steering information" is added to the file so that these sounds can be placed spatially with some approximation of accuracy during playback. This becomes especially important at the upper edge of the bass spectrum, where the ear becomes more sensitive to the spatial location of bass signals. Joint stereo (in "Intensity" mode) really is a low-fi solution best reserved for situations where you need to keep file size at an absolute minimum.

The joint stereo option can *in some instances* introduce audible compression artifacts which can't be removed by increasing the bitrate. The only way to find out whether this is a problem for you is to experiment. If you don't like the results, re-encode without joint stereo enabled. Remember: Your ears don't lie.

Side Information

If joint stereo is used in M/S (middle/side) mode, the left and right channels aren't encoded separately. Instead, a "middle" channel is encoded as the sum of the left and right channels, while a "side" channel is stored as the difference between the left and the right. During the decoding process, side information is read back out of the frame and applied to the bitstream so that the original signal can be reconstructed as accurately as possible. The side information is essentially a set of instructions on how the whole puzzle should be re-assembled on the other end.

Who Defines "Imperceptible?"

Before moving away from the topic of perceptual codecs, there's an important point to be made about the category as a whole: They all make baseline *assumptions* about the limitations of human perception, and about how closely the end result will be listened to. The fact of the matter is that all that stuff being stripped out adds up to something. While no recording format, whether it be vinyl, reel-to-reel, compact disk, or wax cylinder, can capture all of the overtones and subtle nuances of a live performance, nor can any playback equipment on the face of the earth reproduce the quality of a live performance. All compression formats—especially perceptual codecs—are capable of robbing the signal of subtleties. While certain frequencies may not be distinctly perceptible, their cumulative effect contributes to the overall "presence" and ambience of recorded music. Once a signal has been encoded, some of the "magic" of the original signal has been

stripped away, and cannot be retrieved no matter how hard you listen or how good your playback equipment. As a result, MP3 files are sometimes described as sounding "hollow" in comparison to their uncompressed cousins. Of course, the higher the quality of the encoding, the less magic lost. You have to strike your own compromises.

Many feel that the current digital audio standard offers less resolution than the best analog recording, which is why many audiophile purists still swear by vinyl LPs. Digital audio introduced a host of distortions never before encountered with analog, but hasn't had analog's 50+ years of research and development to eradicate them. Compressing and further modifying "CD quality" audio with a lossy perceptual codec like MP3, some might say, adds insult to injury.

But then there's reality, and the reality right now is that the vast majority of us do not listen to music with the trained ears of a true audiophile, nor do most of us possess magnificent playback equipment. Most of us use middle-ground sound cards and PC speakers, most of us have limits to the amount of data we can store conveniently, and most of us connect to the Internet with relatively low-bandwidth modems. Reality dictates that we make compromises. Fortunately, the reality of our sound cards and speakers, the quality of which lags far behind the quality of decent home audio systems, also means that most of these compromises won't be perceived most of the time.

The bottom line is that the perceptual codec represents a "good enough" opportunity for us to have our cake and eat it too. As things stand now, it all comes down to a matter of file size if we want to store and transfer audio files with anything approaching a level of convenience. In a perfect world, we would all have unlimited storage and unlimited bandwidth. In such a world, the MP3 format may never have come to exist—it would have had no reason to. If necessity is the mother of invention, the invention would never have happened. Compression techniques and the perceptual codec represent a compromise we can live with until storage and bandwidth limitations vanish for good.

The Huffman Coding

At the end of the perceptual coding process, a second compression process is run. However, this second round is not a perceptual coding, but rather a more traditional compression of all the bits in the file, taken together as a whole. To use a loose analogy, you might think of this second run, called the "Huffman coding," as being similar to zip or other standard compression mechanisms (in other words, the Huffman run is completely lossless, unlike the perceptual coding techniques). Huffman coding is extremely fast, as it utilizes a look-up table for spotting possible bit substitutions. In other words, it doesn't have to "figure anything out" in order to do its job.

The chief benefit of the Huffman compression run is that it compensates for those areas where the perceptual masking is less efficient. For example, a passage of music that contains many sounds happening at once (i.e., a "polyphonous" passage) will benefit greatly from the masking filter. However, a musical phrase consisting only of a single, sustained note will not. However, this passage can be compressed very efficiently with more traditional means, due to its high level of redundancy. On average, an additional 20% of the total file size can be shaved during the Huffman coding.

Raw Power

If you've surmised from all of this that encoding and decoding MP3 must require a lot of CPU cycles, you're right. In fact, unless you're into raytracing or encryption cracking, encoding MP3 is one of the few things an average computer user can do on a regular basis that consumes *all* of the horsepower you can throw at it. Note, however, that the encoding process is far more intensive than decoding (playing). Since you're likely to be decoding much more frequently than you will be encoding, this is intentional, and is in fact one of the design precepts of the MP3 system (and even more so of next-generation formats such as AAC and VQF).

Creating an MP3 file, as previously described, is a hugely complex task, taking many disparate factors into consideration. The task is one of pure, intensive mathematics. While the computer industry is notorious for hawking more processing power to consumers than they really need, this is one area where you will definitely benefit from the fastest CPU (or CPUs) you can get your hands on, if you plan to do a lot of encoding.

It's impossible to recommend any particular processor speed, for several reasons:

- People have very different encoding needs and thresholds of what constitutes acceptable speed.
- Encoders, as you'll see in Chapter 5, vary *radically* from one to the next in terms of their overall efficiency.
- Any mention of specific processor speeds would surely be out-of-date by the time you read this.
- There's more to the equation than just the speed of the CPU. While MHz may be a good measure of CPU speed when comparing processors from the same family, it's not a good performance measurement between systems. The difference in size of the chip's on-board cache may have a big impact on encoding speeds, as do floating-point optimizations, so one must be careful to note such differences when benchmarking.

In any case, it's easy enough to set up batch jobs with most encoders, so you can always let 'er rip while you go out to lunch, or even overnight. Unless you're really stuck with an old clunker of a machine (a CPU manufactured prior to 1996, for example) and your needs aren't intensive, don't even think about running out to get a new computer just to pump up your encoding speed. You'll be better off making sure you have an adequate complement of RAM, a fast and accurate DAE-capable* CD-ROM drive, a good sound card, and that you're using the most efficient encoder available for your platform (see Chapter 5).

Notes on Decoding

As noted earlier, the great bulk of the work in the MP3 system as a whole is placed on the encoding process. Since one typically plays files more frequently than one encodes them, this makes sense. Decoders do not need to store or work with a model of human psychoacoustic principles, nor do they require a bit allocation procedure. All the MP3 player has to worry about is examining the bitstream of header and data frames for spectral components and the side information stored alongside them, and then reconstructing this information to create an audio signal. The player is nothing but an (often) fancy interface onto your collection of MP3 files and playlists and your sound card, encapsulating the relatively straightforward rules of decoding the MP3 bitstream format.

While there are measurable differences in the efficiency—and audible differences in the quality—of various MP3 decoders, the differences are largely negligible on computer hardware manufactured in the last few years. That's not to say that decoders just sit in the background consuming no resources. In fact, on some machines and some operating systems you'll notice a slight (or even pronounced) sluggishness in other operations while your player is running. This is particularly true on operating systems that don't feature a finely grained threading model, such as MacOS and most versions of Windows. Linux and, to an even greater extent, BeOS are largely exempt from MP3 skipping problems, given decent hardware. And of course, if you're listening to MP3 audio streamed over the Internet, you'll get skipping problems if you don't have enough bandwidth to handle the bitrate/sampling frequency of the stream.

Some MP3 decoders chew up more CPU time than others, but the differences between them in terms of efficiency are not as great as the differences between their feature sets, or between the efficiency of various encoders. Choosing an MP3 player becomes a question of cost, extensibility, audio quality, and appearance.

* DAE stands for Digital Audio Extraction, and refers to a CD-ROM drive's ability to grab audio data as raw bits from audio CDs, so you don't have to rip tracks via the sound card. More details on DAE can be found in Chapter 5.

That's still a lot to consider, but at least you don't have to worry much about benchmarking the hundreds of players available on the market (unless you've got a really slow machine).

If you're a stickler for audio quality, you've probably got a decent to excellent sound card already. However, if you've got an older sound card (such as a SoundBlaster 16) and a slower CPU (slower than a Pentium 133), be aware that the "look ahead" buffer in the MP3 player can easily become exhausted, which will result in an audible degradation of sound quality. However, sticking a better sound card (such as a SoundBlaster 64) in the same machine may eliminate these artifacts, since better sound cards perform more of the critical math in their own hardware, rather than burdening the computer's CPU with it.

While this situation won't affect many modern geeks, there's an easy way to test your equipment to determine if its lack of speed is affecting audio quality: Just pick a favorite MP3 file and decode it to a noncompressed format such as WAV, then listen to the MP3 and the WAV side-by-side. If the WAV version sounds better, you'll know that your machine isn't up to the MP3 playback task, since the uncompressed version requires very little processing power to play.

The Anatomy of an MP3 File

Aside from being familiar with the basic options available to the MP3 encoder, the typical user doesn't need to know how MP3 files are structured internally any more than she needs to know how JPEG images or Word documents are structured behind the scenes. For the morbidly curious, however, here's an x-ray view of the MP3 file format.

Inside the Header Frame

As mentioned earlier, MP3 files are segmented into zillions of frames, each containing a fraction of a second's worth of audio data, ready to be reconstructed by the decoder. Inserted at the beginning of every data frame is a "header frame," which stores 32 bits of meta-data related to the coming data frame (Figure 2-4). As illustrated in Figure 2-5,* the MP3 header begins with a "sync" block, consisting of 11 bits. The sync block allows players to search for and "lock onto" the first available occurrence of a valid frame, which is useful in MP3 broadcasting, for moving around quickly from one part of a track to another, and for skipping ID3 or other data that may be living at the start of the file. However, note that it's not enough

* Figure 2-5 is based on a diagram produced by ID3.org (*www.id3.org/mp3frame.html*).

for a player to simply find the sync block in any binary file and assume that it's a valid MP3 file, since the same pattern of 11 bits could theoretically be found in any random binary file. Thus, it's also necessary for the decoder to check for the validity of other header data as well, or for multiple valid frames in a row. Table 2-1 lists the total 32 bits of header data that are spread over 13 header positions.

Table 2-1. The Thirteen Header Files' Characteristics

Position	Purpose	Length (in Bits)
A	Frame sync	11
B	MPEG audio version (MPEG-1, 2, etc.)	2
C	MPEG layer (Layer I, II, III, etc.)	2
D	Protection (if on, then checksum follows header)	1
E	Bitrate index (lookup table used to specify bitrate for this MPEG version and layer)	4
F	Sampling rate frequency (44.1kHz, etc., determined by lookup table)	2
G	Padding bit (on or off, compensates for unfilled frames)	1
H	Private bit (on or off, allows for application-specific triggers)	1
I	Channel mode (stereo, joint stereo, dual channel, single channel)	2
J	Mode extension (used only with joint stereo, to conjoin channel data)	2
K	Copyright (on or off)	1
L	Original (off if copy of original, on if original)	1
M	Emphasis (respects emphasis bit in the original recording; now largely obsolete)	2
		32 total header bits

Following the sync block comes an ID bit, which specifies whether the frame has been encoded in MPEG-1 or MPEG-2. Two layer bits follow, determining whether the frame is Layer I, II, III, or not defined. If the protection bit is not set, a 16-bit checksum will be inserted prior to the beginning of the audio data.

The bitrate field, naturally, specifies the bitrate of the current frame (e.g., 128 kbps), which is followed by a specifier for the audio frequency (from 16,000Hz to 44,100Hz, depending on whether MPEG-1 or MPEG-2 is currently in use). The padding bit is used to make sure that each frame satisfies the bitrate requirements exactly. For example, a 128 kbps Layer II bitstream at 44.1kHz may end up with some frames of 417 bytes and some of 418. The 417-byte frames will have the padding bit set to "on" (1) to compensate for the discrepancy.

Locking onto the Data Stream

One of the original design goals of MP3 was that it would be suitable for broadcasting. As a result, it becomes important that MP3 receivers be able to lock onto the signal at any point in the stream. This is one of the big reasons why a header frame is placed prior to each data frame, so that a receiver tuning in at any point in the broadcast can search for sync data and start playing almost immediately. Interestingly, this fact theoretically makes it possible to cut MPEG files into smaller pieces and play the pieces individually. However, this unfortunately is not possible with Layer III files (MP3) due to the fact that frames often depend on data contained in other frames (see "Dipping into the reservoir," earlier). Thus, you can't just open any old MP3 file in your favorite audio editor for editing or tweaking.

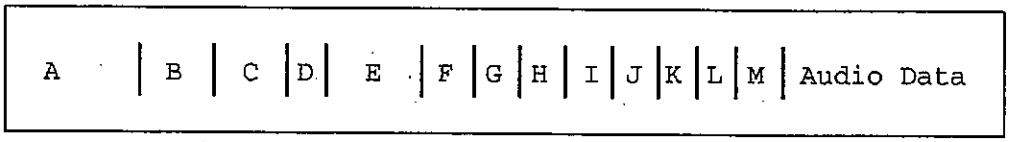


Figure 2-4. Data describing the structural factors of that frame; this data is called the frame's "header"

The mode field refers to the stereo/mono status of the frame, and allows for the setting of stereo, joint stereo, dual channel, and mono encoding options. If joint stereo effects have been enabled, the mode extension field tells the decoder exactly how to handle it, i.e, whether high frequencies have been combined across channels.

The copyright bit does not hold copyright information per se (obviously, since it's only one bit long), but rather mimics a similar copyright bit used on CDs and DATs. If this bit is set, it's officially illegal to copy the track (some ripping programs will report this information back to you if the copyright bit is found to be set). If the data is found on its original media, the home bit will be set. The "private" bit can be used by specific applications to trigger custom events.

The emphasis field is used as a flag, in case a corresponding emphasis bit was set in the original recording. Th emphasis bit is rarely used anymore, though some recordings do still use it.

Finally, the decoder moves on through the checksum (if it exists) and on to the actual audio data frame, and the process begins all over again, with thousands of frames per audio file.

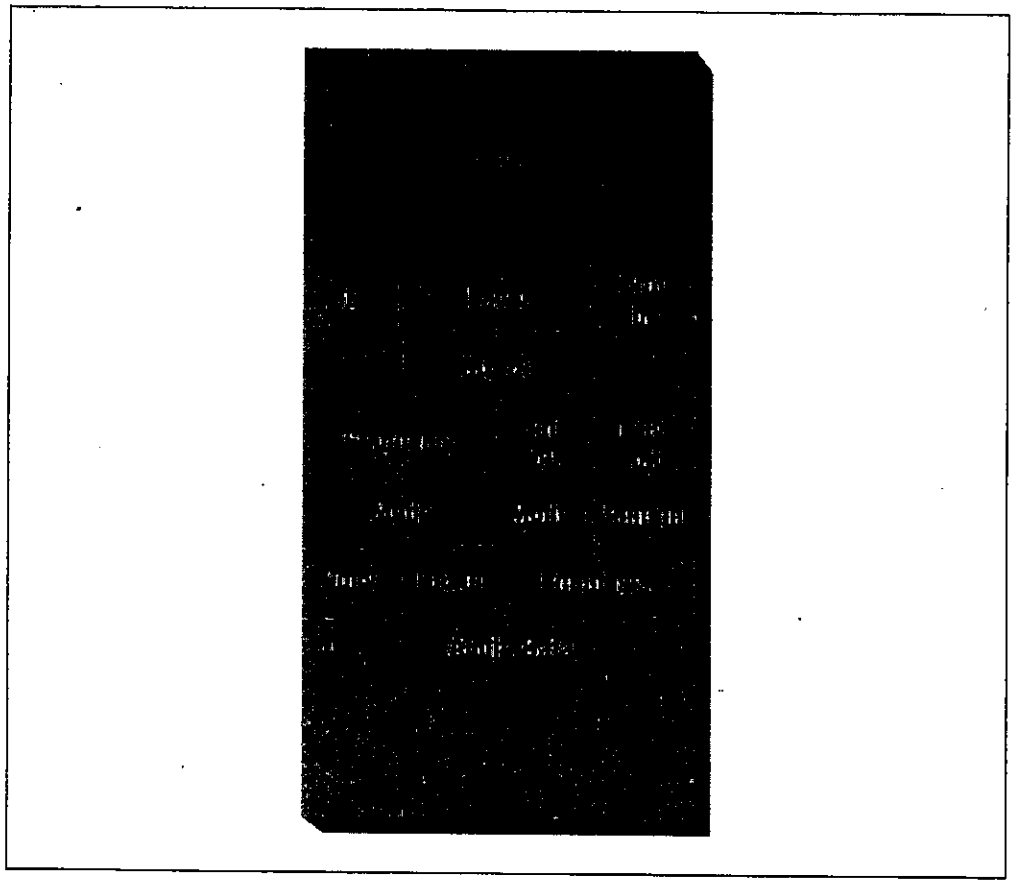


Figure 2-5. The MP3 frame header represented visually

For more details on the structure of MP3 header frames, including the actual lookup tables necessary to derive certain details from the bit settings previously listed, see the Programmer's Corner section at *www.mp3tech.org/*. If you want to go straight to the horse's mouth, start at *www.iso.ch*.

ID3 Space

http://www.id3.org/ Tacked to the beginning or end of an MP3 file, "ID3" tag information may be stored (see Chapter 4), possibly including artist and title, copyright information, terms of use, proof of ownership, an encapsulated thumbnail image, and comments. There are actually two variants of the ID3 specification: ID3v1 and ID3v2, and while the potential differences between them are great, virtually all modern MP3 players can handle files with tags in either format (though a few older players will have problems with ID3v2 tags). Not only are ID3v2 tags

capable of storing a lot more information than ID3v1 tags, but they appear at the beginning of the bitstream, rather than at the end. The reason for this is simple: When an MP3 file is being broadcast or streamed rather than simply downloaded, the player needs to be able to display all of this information throughout the duration of the track, not at the end when it's too late.

It's unfortunate that ID3 tags ever ended up being tagged onto the end of MP3 files to begin with; we'd be much better off if all MP3 files stored their ID3 data at the beginning rather than at the end of the file. As it stands, some MP3 players will simply give up if actual audio data is not encountered within the first few frames. While players developed to the actual ISO MPEG specification will know how to handle either type, the specification itself is unfortunately vague on this point. It simply states that a player should look for a "sync header," without specifying exactly where seeking should start and stop. This laxness in the spec has caused some controversy among developers of ID3-enabled applications, who naturally don't want their applications seeking blindly through 1GB image files, should the user happen to hand one to the application. Fortunately, the ID3v2 spec is more specific on the matter.

One of the more interesting portions of the ID3 specification is the numerical categorization of types of audio, as shown in the Appendix. The numerical identifiers are stored in the ID3 tag, and typically mapped to the actual names via a picklist or another widget in the MP3 player or ID3 tool.

Frames per Second

Just as the movie industry has a standard that specifies the number of frames per second in a film in order to guarantee a constant rate of playback on any projector, the MP3 spec employs a similar standard. Regardless of the bitrate of the file, a frame in an MPEG-1 file lasts for 26ms (26/1000 of a second). This works out to around 38fps. If the bitrate is higher, the frame size is simply larger, and vice versa. In addition, the number of samples stored in an MP3 frame is constant, at 1,152 samples per frame.

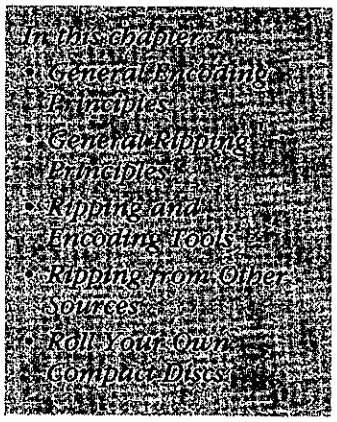
The total size in bytes for any given frame can be calculated with the following formula: FrameSize = 144 * BitRate / (SampleRate + Padding).

Where the bitrate is measured in bits per second (remember to add the relevant number of zeros to convert from kbps to bps), SampleRate refers to the samplerate of the original input data, and padding refers to extra data added to the frame to fill it up completely in the event that the encoding process leaves unfilled space in the frame. For example, if you're encoding a file at 128 kbps, the original samplerate was 44.1kHz, and no padding bit has been set, the total size of each frame will be 417.96 bytes: 144 * 128000 / (44100 + 0) = 417.96 bytes.

Keeping in mind that each frame contains the header information described above, it would be easy to think that header data accounts for a lot of redundant information being stored and read back. However, keep in mind that each frame header is only 32 bits long. At 38fps, that means you get around 1,223 bits per second of header data, total. Since a file encoded at 128 kbps contains 128,000 bits every second; the total amount of header data is miniscule in comparison to the amount of audio data in the frame itself.

5

Ripping and Encoding: Creating MP3 Files



Sure there's tons of free music available on the Internet. But are you able to find the music you're actually looking for? Chances are that the music you seek is out there somewhere, but servers go up and down every day, and you'll quickly discover that the MP3 search engines don't always deliver the goods. When you do find the track you're looking for, you may find that it's poorly encoded. Most of the time, you won't even know what bitrate has been used in the encoding until you've already got the file in question on your hard drive. And, of course, there's the fact that most MP3s available for download are pirated, which means many users end up depriving the artists they respect out of income to which they're entitled.

There's one meta-solution to all of this: Spend your energy on encoding your own CD/LP/DAT/8-track collection, rather than trying to download someone else's. If you don't redistribute the files you encode, you won't run afoul of legal hassles, you'll have complete control over quality issues, and you won't have to wait for lengthy downloads. Of course, if you're an artist, you'll also want to know how to create MP3s from your own original music.

General Encoding Principles

Creating MP3 files is generally a two-step process: Extract your audio from the original source medium into an uncompressed format stored on your hard drive, then run that uncompressed audio through an MP3 encoder. However, many tools exist to help you rip and encode through a single interface, in a single pass. Before taking a look at the tools and processes, you may want to read these notes and suggestions on ripping and encoding principles.

Achieving Optimum Quality

As described in Chapter 2, *How MP3 Works: Inside the Codec*, MP3 is a "lossy" compression format, meaning that some audio information is discarded in exchange for smaller file sizes. The big question you have to face when encoding your own MP3 files is *how much* information do you want to discard. The more you throw away, the worse your files will sound and the smaller your MP3 files will be. The more you keep, the better they'll sound and the larger the resulting files will be. Only you can decide where on this spectrum you want to sit, but again, always remember that your computer may not be the only place where you play your MP3 files. Next year you may purchase an MP3 playback component for your home stereo, a better sound card, or better computer speakers. All of a sudden, you may discover that the MP3 files you once thought sounded just fine don't sound so hot after all. Always aim for a threshold higher than your current tolerances, and remember that disk space and blank CDs just keep on getting cheaper. Unless you'll be listening to MP3s in a noisy environment such as a car, or will be dealing with limited storage space (as you might with a portable MP3 player), you can probably afford the larger file sizes incurred by going with a higher bitrate. I recommend setting 128 kbps or approximately 50% VBR (VBR is explained later in this chapter) as your lower threshold, and going for 160 kbps, 192 kbps, or higher if you're serious about this stuff. Music with a lot of smooth, or synthesized tones (such as techno) will far better at lower bitrates. Note, however, that no matter how high you set the bitrate, Garth Brooks will still suck lemons.

Remember: Quality can always be reduced later from the original source material, but it's extremely difficult to significantly improve on the quality of an existing recording. At this writing, encoded MP3 files cannot be altered at all without decoding to WAV and then back to MP3. However, even though it's a very difficult task, never say never—this may become possible in the future. In any case, "Garbage In, Garbage Out" (GIGO) has been the motto of programmers for decades, and it applies very well to recording and encoding considerations as well. Think of it as if you were cutting your hair—you can always chop more off later, but if you cut too much, you have to suffer the consequences.

Pre-encoding optimizations

So what can you do prior to encoding to optimize the quality of the final results? The answer to this relies on the fact that encoding is always a two-step process. First the source signal is transferred into an uncompressed file format such as WAV or AIFF, and then that uncompressed file is encoded to MP3. Even if you use "all-in-one" software such as MusicMatch Jukebox (covered later in this chapter), that software still has to take this intermediate step, even if it's transparent to the

user. So... while your audio is still in this raw, uncompressed, and highly editable state, you can open it in a sound editing/mastering tool of your choice and perform any necessary equalization, de-hissing, de-popping, and de-scratching. You can cut the silent bits off the beginning and end of your files, add effects, alter the levels, and more. There are many such tools available for various operating systems, and you should be able to find them at your favorite software library.

Technically speaking, you can *only* perform these sorts of functions on uncompressed audio, not on MP3. However, there are increasing numbers of tools that will appear to let you work directly on MP3 audio, such as CoolEdit 2000 (*www.syntrillium.com/cooledit/*) or GoldWave (*www.goldwave.com*). However, these tools are actually decoding MP3 to PCM audio behind the scenes, and then recompressing when finished, so they're not actually editing MP3, even if they make it seem like they are. If you want to edit an existing MP3 file, your only option is to decode it to WAV or AIFF (this chapter), run your voodoo on it, and then re-encode it. However, this process of going back and forth between formats will probably degrade the file's quality enough to outweigh any gains you might achieve. Your best bet is usually to re-encode the original source material.

While the possibility of "re-mastering" your original source material is always there whether it was originally stored on CD, LP, cassette, etc., you'll probably find that this option is really only interesting/useful when trying to clean up old or damaged non-CD material that you simply can't get into MP3 format any other way.

You may also find dedicated audio-editing tools useful for creating "unusual" MP3s. For example, you may want to create an audio collage out of snippets of sound sources, or create psychedelicized versions of your favorite songs by running flange, echo, reverb, and reversal effects on them in an audio editor. Save the results of your auditory mangulations back to WAV format and encode them to MP3 just as you would any other track. Popular sound editing tools include CoolEdit Pro and SoundForge for Windows, Deck or Peak for the Macintosh, XWave for Linux, and Pebbles or 3dsound for BeOS.

A true audiophile will recommend not preprocessing prior to encoding, reason being that encoding directly from the source material will give you the most neutral signal possible—you can always alter the signal somewhat with your MP3 player's built-in equalizer later on. However, this admonishment doesn't take into account situations where you just want to remove blank space from the beginning or end of a file, mix your signal with other sources, or do other things you can't easily do after the fact with an MP3 file.

Samplerates

The quality of uncompressed audio is in large part determined by its *samplerate*. This is a measure of how many times per second the audio signal is digitally represented in the final stream (whether it be "live" or stored on disk), and is typically described in Kilohertz. A samplerate of 22kHz means that "slices" of the audio stream are represented 22,000 times per second. As with bitrates, the higher the samplerate, the better the quality and the larger the resulting file.

All consumer compact disks are sampled at a rate of 44.1kHz, 16 bit (16 bit meaning there are 65,536 possible values—i.e., 2^16—for each sample), and you're well-familiar with compact disk audio quality. However, the next generation of digital audio hardware, particularly in the high-end and professional audio recording space, will be built to handle 24-bit audio at 96kHz, more than doubling the number of samples per second and greatly increasing the number of possible values for each bit. The result will be much higher fidelity and much less of the "cold," "brittle," "mechanical," or "grainy" effect that audiophiles typically associate with CD audio. Utilizing this higher samplerate on your home computer will require both a 96kHz-capable sound card and upgraded software that can handle the higher data throughput. Note, however, that MP3 can't handle anything higher than 48kHz; if that's your bag, you'll have to turn to a codec like AAC or MPEG-4.

For detailed comparative analyses between MP3, RealAudio, and MS Audio encodings, see *david.weekly.org/audio/*.

For most MP3 users, that's neither here nor there. More likely is the possibility that you'll want to *decrease* samplerates prior to encoding. You might want much smaller file sizes at the expense of fidelity when working with the spoken word—for encoding class lectures, for example. Even more likely is that you'll want to start running a SHOUTcast or icecast server, streaming MP3 audio from your home computer to the Internet at large. In streaming situations, decreasing the samplerate of your outgoing signal is the easiest way to decrease the bandwidth necessary for listeners to hear your broadcast without skipping and without huge buffers.

Lower samplerates are, by the way, the main reason why streamed MP3 typically sounds worse than standard MP3 downloads, though broadcasters will often decrease both the samplerate *and* the bitrate. Lowering the samplerate results in less resolution, while lowering the bitrate results in poorer dynamics.

Samplerates can be controlled in two ways. You can open your uncompressed audio files in any audio editing software prior to encoding, downsample it to a lower rate (see your audio editor's documentation), and then encode. If you're broadcasting MP3 streams from your machine, you can tell SHOUTcast or icecast to downsample your existing MP3 files "on-the-fly." More on that in Chapter 8, *Webcasting and Servers: Internet Distribution.*

Speed vs. Quality

It's hard to imagine another area in the software industry where similar applications differ so radically in terms of performance. While encoders may all share the same basic functionality, the degree to which the underlying source code has been optimized for speed differs *radically* from one encoder to the next. To throw yourself for a loop, rip a WAV file from your favorite CD, then download as many encoders as you can find for your platform. Use each encoder in turn to compress the WAV file, using identical encoding options if possible, and carefully time the process in each case. You'll be amazed to find that some encoders are as much as ten times faster or slower than others.

The reason for the disparity is simple: Some developers and vendors feel that encoding speed is of the utmost importance when it comes to pleasing the general public and in making the MP3 phenomenon viable. Thus, you'll find that some of the big commercial MP3 encoders, such as MusicMatch Jukebox and RealJukeBox are able to encode faster than real time (e.g., a 3-minute song may be encoded in 90 seconds). Meanwhile, some freeware encoders, such as BladeEnc, may require 25 minutes to encode the same song.

Again, speed and quality may or may not be directly related. For example, early tests of the LAME patches against the ISO sources showed that it was capable of producing bit-by-bit identical output to that of BladeEnc, but 2.5x faster.

As you might guess, not all of the speed delta is achieved through simple optimization of source code and by taking advantage of floating-point CPU extensions—some of it comes from *cheating*—discarding frequencies that the developers felt just weren't as important (although this only shaves one-twentieth of a second from encoding time, and doesn't affect the quality of encodings at 128 KB or lower). Purists may prefer to stick to brass tacks and wring every possible drop of quality from their encodings, but in most cases, you can great results with fast encoders—don't automatically assume that fast means bad. If you do decide to work with a slower encoder for whatever reason, you can always let your encoding process chug along merrily in the background as you work, or to set up batch encoding jobs to crunch as you sleep.

While not 100% applicable, the general rule of thumb is that the encoders you pay for, especially those from large companies, are based on either Xing or the later versions of the Fraunhofer coder MP3ENC, which at this writing were the fastest in the biz. While some freeware encoders may be faster than others, few can begin to approach the speed of these two, with the Fraunhofer coder edging out Xing in most tests, both on speed and on quality. In fact, after a long period of dominance, some major vendors (such as MusicMatch) were beginning to shift away from Xing in favor of Fraunhofer. Note also that many speed advantages come from MMX processor optimizations.

Because MP3 encoding is so math-intensive, PowerMac users may enjoy faster encoding speeds than x86 users. In fact, a PowerMac G4 with a fast, DAE-accurate CD-ROM drive (see the "Notes on manual track separation" section later in this chapter), is the fastest ripping/encoding solution available to consumers at this writing. G4 users have reported being able to rip and encode tracks up to ten times faster than the playback speed of the track in question. Current advances in x86 technology—such as the AMD Athlon processor and Intel's Merced architecture—should even the score here. But remember: it's not all about raw processor speed; the real question is how well-optimized the CPU and the encoding software are at handling floating point and FFT functions. x86 encoders such as GOGO (covered later in this chapter) do an exceptional job at taking full advantage of special processor instructions such MMX, SSE, and 3DNow!, and at working with multiple processors if present.

Normalization

One bugaboo that often crops up when creating mixed song collections is the fact that the original source materials are all recorded at slightly different levels, leaving you with MP3 files of varying volumes. This isn't much of an issue when playing entire albums, but can get annoying with mixed collections and random playlists. The solution is to use a *normalizer*, which will boost the overall signal of weakly recorded tracks and diminish levels for loud ones. See the "Plug-ins: Extending Your Reach" section in Chapter 4, *Playlists, Tags, and Skins: MP3 Options*, to learn about the AudioStocker normalizing plug-in for WinAmp. Many encoding applications also have normalization functions built in, as do some CD burning applications; see your encoder's or burner's documentation for details.

Most normalizers let you control the "threshold" of normalization. If you normalize at 100%, all tracks will have the same average volume, whereas 95% normalization will allow for some distinction to be made between loud and soft tracks, which is probably what you want in most cases.

Normalizing downward (decreasing the volume) can degrade the sound quality of both PCM and MP3 audio because of the linear distribution of sample levels. Not a ton—but enough to be noticeable to some people. You may want to do your normalization in your MP3 player rather than at encode time, so you judge for yourself whether quality has been audibly degraded, and not have to re-encode your files if it turns out you don't like the results. If you normalize during the encoding process, those files will be normalized forever. If you normalize at playback time, you can always make adjustments in the equalizer or volume controls to accommodate different listening environments. However, some people feel there are more advantages to normalizing at encode time, because you only have to do it once, and you won't be consuming extra processor cycles with every playback. Go with the solution that works best for you.

The Venerable CDDB

Surprisingly enough, compact disks do not include a simple table of contents listing the names of the tracks they contain.* As a result, ripping and encoding tools have no easy way to give your extracted files reasonable file names. By default, ripped tracks will end up with names like *Track01.wav*, *Track02.wav*, and so on.

However, it *is* possible to extract a unique identifying string (called a CDDB-ID) from any audio CD by summing the lengths of the tracks on a disc and running a quick mathematical algorithm on it to generate a unique identifying number. The chances of this ID number for any two audio CDs being identical are very small.

Conversely, it is not always the case that there is only one unique CDDB-ID for a given *album*. Because pressing plants may press different batches of the same album with different techniques in different months, and because of variants in promotional CDs and remastered versions of original albums, the same artist/album can potentially generate multiple valid CDDB-IDs. Some encoding tools will even show you the results of an ambiguous lookup if they encounter a disc they can't be sure about.

Thanks to this capability, one of the Internet's great collaborative efforts has arisen: The *Compact Disk Database*, or CDDB (*www.cddb.com*), is populated by normal

* Although they do contain a "reference table" that allows CD players to discover the byte offsets where tracks begin. In addition, the CD spec actually does allow for a text listing of audio tracks, though it has seldom been used by the industry. Sony now takes advantage of this capability, under the name "cd-text." Others may follow.

users with the track listings of just about every CD ever made (only the most obscure recordings have not been logged—see if you can stump it!). While you can search for track listings directly at the CDDB web site, the primary use of the database is as a back-end to supporting CD players and ripper/encoder tools, as shown in Figure 5-1.

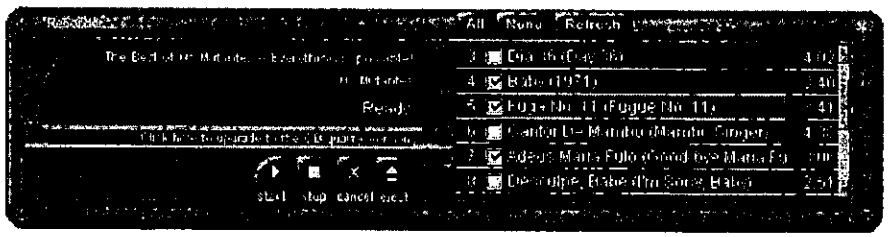


Figure 5-1. MusicMatch JukeBox uses the CDDB to extract CD-specific information

Any ripping/encoding tool can extract a CD's unique identifier string, look it up in the online CDDB, and return a plain-text table of contents, including artist name, album name, and track listing. It can then use this table of contents to give your ripped or encoded tracks reasonable file names, or even output files to directories and subdirectories named for the artist and album. In addition, this data can be used to embed meaningful ID3 tags in your files as they're being encoded, so you don't have to do it manually later on. Note that ID3 tag creation isn't always enabled by default. Look around in your encoder's settings panels for an ID3 options checkbox.

Of course, taking advantage of the CDDB requires online access, at least as the ripping session begins. But even if you don't enjoy a permanent Internet connection, rippers and encoders that know how to speak CDDB will save you tons of file-renaming time, and it's worth the effort to make the connection. Most operating systems also maintain a database of audio CDs that have already been looked up, so the next time you insert the same CD you won't have to run the lookup procedure again. Most of the tools covered in this chapter support CDDB connections, or interface with other tools that do.

If you're using ID3v2 tags and are looking for lyrics to associate with your tracks, there's no better place to search than the International Lyrics Server, at *www.lyrics.ch*. However, legal actions taken in 1999 forced the site to put all lyrics into Java applets, which prevents users from simply copying and pasting them into other applications. You'll have to manually type the lyrics into your ID3v2 tagger.

Alternatives to the CDDB

Escient, who owns and maintains the CDDB, imposes certain requirements on developers of applications who make use of their database. For example, their license requires that such applications display the CDDB logo, and that those applications do not make use of any competing database. Some developers—especially those in the Open Source community—feel that these requirements are unfair and overly restrictive. What, for example, would this mean for the developer of a command-line-only CDDB lookup tool which can't display the logo by definition, or for a CDDB-savvy filesystem, such as the one built into BeOS?

In response, a couple of alternatives to CDDB have arisen, without these restrictions. You may find some tools configured to point to freedb (*www.freedb.org*), rather than CDDB by default, and you can of course edit your encoder's preferences to point to freedb instead. Another alternative, cdindex (*www.cdindex.org*) intends to offer a superior technique for identifying CDs, fuzzy lookups, and instant web-based submission of new albums. The only disadvantage to these alternatives is that their databases are smaller, so you may not get successful lookups on some more obscure CDs. Since most encoders will let you select multiple databases to check, you can tell your tool to look at freedb or cdindex first, and then to look at CDDB only if the CD isn't found.

Other Encoding Options

While all encoders include some tool, widget, picklist, or command-line flag to let you select the bitrate at which your files should be encoded, bitrate selection is where the similarity ends in terms of available options. While it would be impossible to catalog all of the encoding options made available by all available encoders, here are descriptions of the most commonly found encoding options. If you don't find these options in your encoder's interface or described in the documentation, it may be time to look for another encoder (assuming these options are important to you).

As a general rule, you'll probably find that some or most of these options are absent from the majority of encoders aimed at the general public, while the more arcane options are often found in command-line and less-popular encoders. More options means more complexity and possibly a more intimidating interface. To be perfectly frank, however, most of the more unusual encoding options are simply not useful to most of us most of the time. If all you want to do is encode your collection with reasonable quality and a minimum of fuss, you'll probably never need to think about most of these possibilities. If your encoder's interface does have these options, the chances are they'll already be set to reasonable defaults, and you can leave them alone.

MPEG-1 vs. MPEG-2

Some encoders may let you choose between the various MPEG classes, as described in Chapter 1, *The Nuts and Bolts of MP3*. When creating MP3 files, you always want to make sure this is set to MPEG-1.

Layer

Some encoders are capable of creating Layer I, II, or III files. Unless you're encoding audio that's destined to be embedded in an MPEG movie or want to create MP2 files for any other reason, you want to make sure this is always set to Layer III. Since Layer III is the most complex, this setting will always take the longest, but you can't make MP3s without it.

Stereo/Joint Stereo/Dual-channel/Mono

You'll probably encounter few occasions where you'll actually want to create monophonic MP3s, unless you're encoding the spoken word or preparing your files for Internet broadcast and want to save on file size. For most purposes, leave this set to stereo. Remember that a 128 kbps MP3 file actually stores left and right channels separately, allocating proportions of the data dynamically to account for the needs of the respective channels (one may be dynamic while the other grows quiet, resulting in a 65%/35% split, for example). You can enable joint-stereo if you want to save a little space by storing high frequencies monophonically (since very high frequencies are difficult to locate in space), but this capability only goes into effect when the "intensity stereo" option is enabled... which, by the way, destroys phase information and makes the file unsuitable for some applications. The M/S (middle/side) option is not subject to the same limitations as intensity stereo.

Dual-channel was designed to allow for transmission of independent content in either channel—for example, you might want to put an English track in one channel and a Japanese track in the other. Of course, this means that each channel will be restricted to half the bitrate it would otherwise get, so quality will be degraded.

In general, you'll find that joint stereo encoding options work better at lower bitrates.

Psycho 1/Psycho 2

If this option is present (and you'll only find this in ISO-based encoders), you may be able to toggle between two different psychoacoustic models in order to optimize your encoding for various types of music. Psycho 1 is available only when encoding Layers I and II, while Psycho 2 works with Layer III, and is a more complex algorithm, delivering better quality at the expense of slower encoding.

Emphasis/De-emphasis

This option is only useful for files that have been run through a noise-reduction processes in an audio-editing application prior to encoding. If enabled, you can choose between two modes: *50/15 microseconds* or *ccit j.17*. Enabling these options does not trigger a change in the encoding, but merely sets a flag in the MP3 file which can be read back by the MP3 decoder so that corresponding noise reduction can be undertaken by the decoder.

Private/Protection/Copyright/Original

These options set corresponding bits in the MP3 header frame, and do not affect audio signal. They may be used by specific MP3 players to display ownership attributes of the file. Very few MP3 players read or display these bits, and the MP3 world basically ignores them. This kind of data becomes much more important in "secure" formats such as those proposed by SDMI, but not in straight MP3. Artists encoding their own music who want some measure of ownership stated in the file will probably want to set these bits, but note that simply enabling these options will not automatically make your file secure—you'll still need to adopt a complete security system for that, and straight MP3 probably won't be it—you'll probably want to look instead towards Liquid Audio, Windows Media, or any of the other secure formats discussed in Chapter 9, *Competing Codecs and Other File Formats.*

It is possible that the additional meta-data storage area offered by ID3v2 may allow for true security mechanisms to be built into straight MP3 files. While no such implementations were available at this writing, this book strongly recommends investigating the possibility. Avoiding proprietary formats such as Windows Media will help to keep digital music distribution a fair and open game, accessible to users of any player and any operating system. Supporting open formats helps keep any one company from becoming "landlord" over digital music distribution. Fighting MP3 piracy may also help artists and labels from being cautious of utilizing the format.

VBR

As described in Chapter 2, increasing numbers of encoders and decoders support "variable bit rate," or VBR encoding techniques. These allow denser musical passages to use a higher bitrate (and hence more disk space) and sparser passages to use a lower bitrate. This allows for a maximum storage space/quality ratio. VBR thresholds are adjustable not in specific bitrates, but are specified as percentages. The downsides to VBR are that not all MP3 players support VBR-encoded files (though most do), and that you really need to do a lot of careful listening tests to the get the most out of the option. Different types of music respond differently to different VBR thresholds.

Finding the Best Encoder

At the rate the MP3 industry moves, any recommendations on specific encoders would surely become outdated in six months. Nevertheless, almost all encoders are originally based either on the Fraunhofer or the ISO source code bases. Because Fraunhofer has always looked to MP3 as an "Internet radio" solution, their codec is optimized for lower bitrates (at or below 128 kbps). When encoding at 160 kbps or higher, look instead to encoders based on the ISO sources (such as BladeEnc or LAME). Encoders based on the Xing or later Fraunhofer codecs will generally be five to seven times faster than either of these, but Xing suffers some when it comes to the nuances and details because it works in long "blocks" and therefore has trouble with some transient signals.

Didn't Fraunhofer put together the ISO reference sources? If so, why are we discussing them here as separate entities? The reason they are treated separately is because Fraunhofer publishes an encoder under its own umbrella that is separate from the original reference sources. Later versions of the Fraunhofer encoder are better optimized for both quality and speed. Technically speaking, all encoders available are descendants of the original ISO source code, though many (if not most) have evolved along divergent paths as their developers worked on independent and often proprietary optimizations.

You'll see a lot of talk out there about the supposedly poor quality of Xing-based encoders, because Xing established an early reputation for getting high encoding speeds by cutting corners at higher frequencies. You should know, however, that more recent versions of the Xing encoder do not cut as many corners, and most people find their quality quite good. Only the most extreme audiophiles still shy away from Xing encoders, while many former critics of Xing have seen the light and been converted by later editions of the Xing encoder, which works all the way up to 20kHz. Many real sticklers and purists still swear by BladeEnc as offering the absolute highest quality, but its reputation is fading as the later Fraunhofer encoder soaks up more attention, and the development pace of the LAME encoder continues to improve both its quality and its speed.

If you're using a command-line version of the Xing encoder, you can force it to use the old 16kHz cut-off by adding the –N switch to your command.

You'll see the "What's the best encoder?" question asked frequently on MP3 mailing lists and Usenet groups, and will probably be asking yourself the same question before long. Unfortunately, there is no simple answer to this question, and every time the question comes up, you'll see a dozen different people offering a

dozen different answers. You need to evaluate your own needs and priorities, try a bunch of them, determine whether you can hear the differences in the quality of their output, and decide whether speed, quality, or convenience is the most important factor.

Improving the Quality of Existing MP3s

Once you start playing with an audio editing tool, you may get inspired to start tweaking on some of your existing MP3 files. Before you get too excited, be prepared for disappointment; no audio editor under the sun will let you change the sound of an existing MP3 file. It just doesn't work that way. The complex algorithms involved in compressing and decompressing audio to MP3 and back make live MP3 editing impossible. In addition, the "bit reservoir" aspect of the MP3 spec means that each frame may depend on knowing something about the frame that comes before and after itself; if you start cutting up MP3 files, you're going to end up with audible glitches at every edit point (this limitation does not apply to MPEG Layer II, since it doesn't use the bit reservoir). Your only option is to decode your MP3 tracks to WAV or AIFF, edit those, and then re-encode them. Again, going back and forth like this is going to result in an overall degradation of quality, and you'll probably find it isn't worth the bother.

Your best bet when dealing with MP3s of poor quality is to do what you can with your MP3 player's built-in equalizer. In other words, make adjustments in the way your computer handles the output stream, rather than trying to change the MP3 file itself. Of course, that will only get you so far, and won't help to eliminate that "hollow" or "swishy" sound you get with some badly encoded files. If at all possible, just re-encode the file again from the original source.

While you can't edit the actual audio signal stored in an MP3 file, you may be able to cut a few seconds from the beginning or end of a file that starts or ends with "dead air" or other unwanted audio cruft. At this writing, only a couple of tools were known to be capable of this. MP3Trim and MP3 Cutter are both capable of very simple edits to MP3 files (Windows only). Linux and BeOS users may be able to use a tool called mp3asm to clean up "broken" MP3 files, finding and discarding illegal header frames, and even assembling multiple MP3 files into a single new one.

Again, the best solution is usually to just go back and re-encode the file. Use a better ripper, clean it up with aftermarket audio manipulation tools while it's still in WAV, RAW, or AIFF format, and re-encode it. Really and truly, this is the only way to guarantee high quality.

General Ripping Principles

Before you can encode a single bit, you've got to get the music you intend to compress into your computer. This may be accomplished by piping the signal directly into your machine through your sound card's Line In jack (either from a mixing board or from home stereo components) or by extracting tracks from compact discs by way of your machine's CD-ROM drive in a process known as "ripping."

Ripping from Compact Discs

When you insert an audio CD into your machine's CD tray, then try to view its contents through your file manager (such as Explorer), you'll notice that all you see is a collection of 1 KB tracks labeled "Track1.CDA," "Track2.CDA," and so on. Drag one of these tracks to another location, and you'll find that you haven't copied that track's audio data to your system at all. This is because CDA (Compact Disc Audio) tracks aren't actually audio tracks—they're just "handles" that tell the operating system where on the CD to find the actual audio bits. Technically speaking, what you see when you view a folder full of CDA files is the CD's "table of contents," which tells the CD player at which byte offsets to find corresponding PCM data on the disk. By default, most operating system's file managers will only give you access to CDA handles and not to actual audio data, so you need to find a way around the situation if you want to encode those tracks.

You could take the long, cumbersome route and play the audio CDs through your machine's CD player application, then record the sound stream to disc as it plays, but that would grow very tiresome very fast and degrade quality, since the audio stream would be passing through a superfluous set of digital-to-analog and analog-to-digital conversion routines.* The real solution is to use "ripping" software, which is specially designed to read CDA files, find their associated audio data on the CD, and transfer that data to your hard drive as an uncompressed WAV, AIFF, or RAWS file. In other words, rippers let you get around the barriers the operating system throws up in your face, and gives you direct access to the actual audio data living on the CD. There are exceptions to this, as you'll see later on in the chapter.

Don't, however, let this mislead you into thinking that CDs store audio data in WAV format—they don't. They actually store it in a format called *pulse-code modulation*, or PCM. However, PCM, WAV, and AIFF are all slight variants of the same thing: raw, uncompressed, unfiltered audio data.† All of these formats are essentially the same, and none sound any better or worse than the others; they just

* This is how most people ripped audio from CD before DAE-compatible CD-ROM drives became popular.

† Technically, the WAV format can support compressed audio—even MP3 audio—but that's an entirely different story. For all practical intents and purposes, WAV audio is uncompressed audio.

wear slightly different hats (technically, they sport different file headers) and are preferred on different operating systems for historical reasons.

There are literally hundreds of ripping tools and solutions available, covering every operating system under the sun, and they take on dozens of different guises. Some work from the command line, some offer nice graphical interfaces. Some serve multiple purposes, and also let you copy entire data (as well as audio) CDs. Some are capable of encoding to MP3 *as* they're ripping. Some are freeware, others will cost you. Nevertheless, the principles behind all of them are the same, and their usage is generally pretty intuitive. We'll look at some of the most popular ripping solutions for various platforms in the next section. Note, as always, that these are just *examples*—not necessarily recommendations. Check your favorite software library or search engine for other rippers, experiment, and stick with the one that serves your needs.

Notes on digital audio extraction

Ripping audio tracks from CDs is not just a matter of having the right software—your CD-ROM drive must also be capable of *digital audio extraction*, or DAE.* Fortunately, the vast majority of modern CD players are DAE-compatible. Very few users find that their CD-ROM drives are not capable of digital extraction. Even if you do have a DAE-compatible drive, note that some drives are simply better at the process than others, and some ripping software is more "paranoid" (careful) about ripping quality than others.

Ideally, you want to find and use ripping software that does its own error correction, so that if bits aren't sucked off the disc perfectly the first time, redundant passes will be made until the bits being output exactly match the bits on the CD. The process is not as simple as copying normal files through your file manager—the hardware and the software must all cooperate to ensure perfect rips. Look around on the web sites or in the documentation of hardware and software you intend to use to find out whether they perform any kind of CRC checks, redundancy, or error-correction. You may also want to ask online to find out which brands and products produce good results for other users. If in doubt, just rip the same track from a CD a few times and check their exact byte sizes. If any of them differ by even one byte, then you'll know that you're either not using an error-correcting ripper or your CD-ROM unit doesn't do perfect DAE.

* It is possible to rip from audio CDs without a DAE-compatible drive, but this requires going through the analog stage of the sound card, with a concomitant reduction in sound quality. You'll lose a minimum of 20dB to noise in the process, and it's just not worth it. If you're shopping for a new CD-ROM drive, make sure it's DAE-compatible (most are).

When purchasing hardware, keep the following notes about CD-ROM devices in mind, but remember that quality is not always a function of cost; an inexpensive IDE drive *may* outperform a more expensive SCSI drive in the final analysis.

If you have problems creating error-free rips, try slowing down the rip speed. Take it all the way down to 1x if necessary—whatever it takes to eliminate *all* pops and glitches. You may also want to increase the buffer size.

SCSI

The SCSI-1 specification had no standards for audio operations. SCSI-2 introduced audio playback operations, but in a very limited way. SCSI-3 is much more complete and defines a complete set of audio related operations, including playback, audio seeking (a.k.a. fast-forward and rewind), and digital audio extraction. Good SCSI drives include Pioneer, later Yamaha CD-RW drives, and especially some of the newer Plextors, which set the bar for DAE quality. Plextor drives do offer a Windows utility to let you view your CDA tracks directly as WAV files, though files are "blocked" when in use, so you can't play multiple CD tracks simultaneously or cross-fade your CD tracks, as you can in BeOS with the cdda-fs driver. Plextors do excellent error correction.

ATAPI

ATAPI employs a variant of SCSI-3 over the IDE bus to achieve similar capabilities. ATAPI devices are basically inexpensive clones of their SCSI progenitors; their firmware is not as reliable, though some are quite acceptable. Many Toshiba drives produce excellent results, while those from Acer, Cyberdrive, and TAEIL have developed less-than-sterling reputations as audio performers.

If you have trouble getting a ripper to work at all with your CD-ROM drive and you're sure it's DAE-compatible, look around in the options or preferences for ASPI and MSCDEX ripping methods. If one doesn't work, try the other.

Ripping and Encoding Tools

Depending on the tools you use, ripping and encoding may either be a one-step or two-step process. The traditional path is to rip first and then encode. However, increasing numbers of tools that perform both steps in one pass are beginning to appear. We'll cover rippers, encoders, and combination tools together in this section, which is broken down by operating system. Many of these tools have been ported to various platforms, with operation virtually identical for each port.

Windows

As is usually the case, there are far more ripping and encoding options available for Windows users than for any other platform. The following are just a few.

HyCD's HyCD

www.hycd.com HyCD is a well-rounded, multi-purpose tool designed to handle every facet of your CD ripping, encoding, and CD copying needs. Users looking for a way to clean up pops and scratches from LP and other noisy recordings will also find an excellent Effects processor included in the HyCD package. While most of the HyCD tools include an interface dedicated to the task at hand, its built-in ripper/encoder is somewhat oddly designed. Rather than launching a separate application, just navigate through Windows Explorer to the location of an audio CD track or tracks you want to rip. Right-click on those tracks and choose *HyCD Copy* from the context menu. A dialog will appear, asking whether you want to "paste" the files into the selected folder in WAV or MP3 format. In other words, do you just want to rip the selected tracks, or do you want to rip and encode them all in one step? If you think you'll need to pre-process or clean up your WAV files prior to encoding, then you'll just want to rip. Otherwise, choose Encode to go straight from PCM to MP3.

The biggest disadvantage to HyCD's ripper/encoder combination is that it offers no connectivity with the CDDB, so you'll have to manually rename and add ID3 tags to your tracks later. Despite this design limitation, HyCD offers a great collection of additional tools that anyone serious about building a quality MP3 collection will find useful, including a "Sampler" that will turn MP3 files back into WAVs in preparation for burning your own CDs (covered later in this chapter).

Xing's AudioCatalyst

www.audiocatalyst.com One of the most popular all-in-one tools available for Windows is Xing's AudioCatalyst, a very fast combination ripper/encoder with CDDB support, a zillion encoding options, ID3 support, and excellent normalization features, as shown in Figure 5-2. AudioCatalyst also comes bundled with an MP3 player of its own, though power users will probably want to use one of the more sophisticated players outlined in Chapter 3, *Getting and Playing MP3 Files.*

Usage is intuitive and documentation is detailed. Just be sure you check the settings before you begin to make sure ID3 support is enabled, and remember to click the CDDB button before starting so your tracks end up with reasonable filenames.

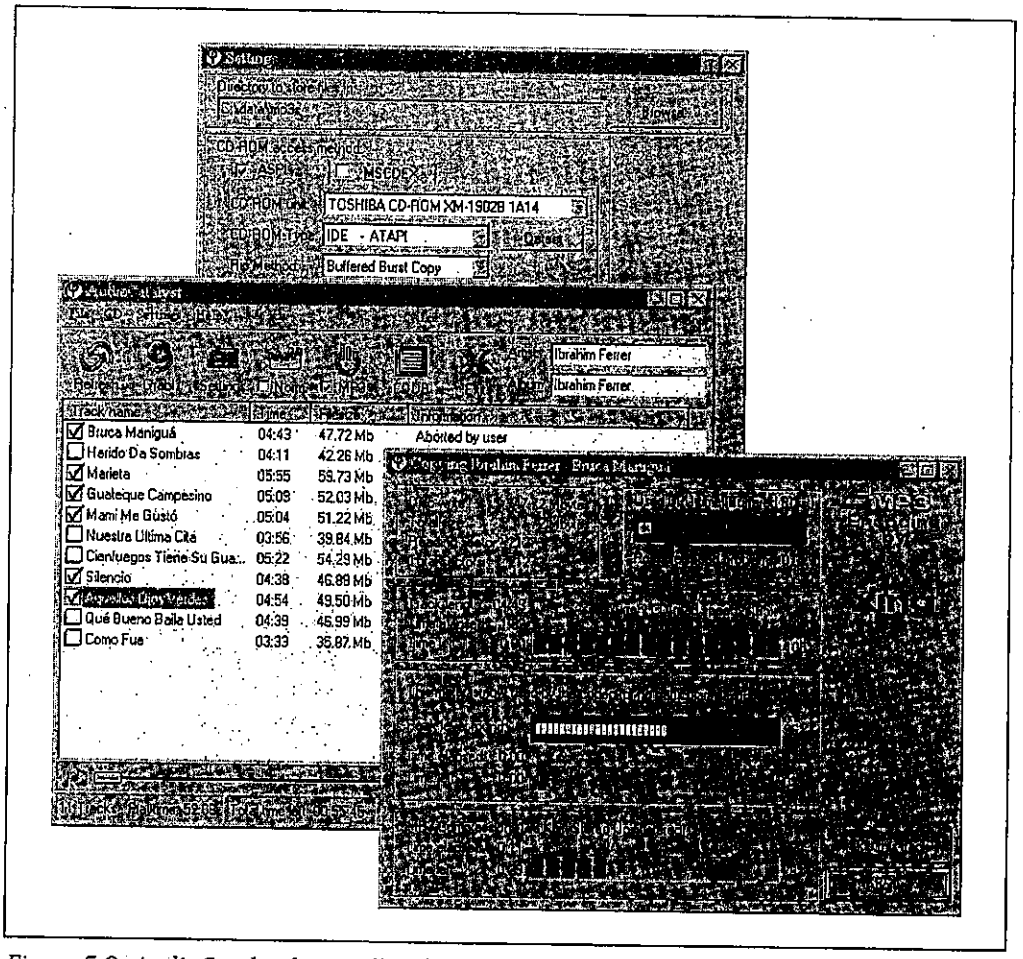


Figure 5-2. AudioCatalyst has it all: a fast encoding engine, a no-nonsense interface, and tons of features and options

AudioCatalyst has trouble detecting some CDs accurately, especially those in laptop computers. If you have a non-SCSI CD-ROM drive and get mysterious error messages as you start to encode, enter the Settings panel and change the drive to "ATAPI – IDE," which should work with the vast majority of drives out there.

AudioGrabber

www.audiograbber.com-us.net A popular cousin of AudioCatalyst is AudioGrabber, which works similarly. If the two seem to have eerie similarities, it's because Xing licensed some of the AudioGrabber code for use in AudioCatalyst. One of the great things about AudioGrabber, though, is the fact that it can work with

alternative encoders that have been compiled into Windows .DLLs, essentially treating encoders like plug-ins. At this writing, the LAME encoder (covered elsewhere in this chapter) had just become available as an option for AudioGrabber users, alongside Xing, BladeEnc, Plugger, and Fraunhofer encoders. The nice thing about this system is that it makes it easy for users to test the comparative speed and quality of various encoders without having to switch to an entirely different system.

MusicMatch Jukebox and Real Jukebox

Two of the easiest-to-use encoding tools on the market are MusicMatch's and RealNetworks' "Jukebox" products, both of which combine ripping/encoding and playback tools into a single interface. Both products are covered in Chapter 3.

CDFS.VXD

Ah, the wonders of the Internet. Somewhere out there, a pretty smart programmer figured out a way to do what Windows—for whatever reason—doesn't offer automatically: Display the tracks on your audio CDs as WAV files directly in Explorer, allowing you to completely bypass the need for third-party ripping tools. This file, which simply replaces your existing Windows CD-ROM driver file, is distributed anonymously, and has no central homepage. No one seems to know where the file originated. Just search the Net for "cdfs.vxd" and unzip the package.

In Explorer, navigate to *C:\WINDOWS\SYSTEM\IOSUBSYS* and rename *CDFS.VXD* to *CDFS.VXD.OLD* (not *CDFS.OLD.VXD*). Then drop the downloaded version of *CDFS.VXD* into this folder, restart your machine, and insert an audio CD. You'll see all of the usual CDA files in the root of the drive as usual, but you'll now also find subdirectories named for stereo and mono variants, 8- and 16-bit bit-depths, and a variety of samplerates. For most purposes, you'll want to navigate to stereo, 16bit, 44,100Hz, where you'll find the highest quality WAV files. However, if you're doing any kind of Internet streaming or need low-size, lower quality WAVs for any other purpose, they're all there waiting for you. Note that *CDFS.VXD* works only under Windows 95 and 98, not Windows NT.

CDFS.VXD does not, unfortunately, do CDDB lookups.

BladeEnc

home.swipnet.se/~w-82625/ Hard-core command-line users may want to look to the legendary BladeEnc. BladeEnc was developed by a programmer who was dissatisfied with the quality of the typical 128 kbps MP3 file, and wanted to ensure the best quality he could for his own encodings. BladeEnc was once hailed as being one of the best quality encoders available, though this claim is now widely disputed. While BladeEnc was originally based on ISO sources, its encoding engine was later optimized for speed as well, eventually making it 50–100% faster

than the original ISO source (although BladeEnc is still no speed demon—in our tests, it was slower than virtually every other encoder available). BladeEnc's author feels that his encoder surpasses its Fraunhofer competitors in quality at around 160 kbps.

Usage of BladeEnc is pretty straightforward. A simple:

```
bladenc *.wav
```

Will encode every WAV file in the current directory at the default rate of 128 kbps, while:

```
bladenc -160 *.wav
```

will do the same, but at 160 kbps. If you want to specify the output directory, just use the -output flag:

```
bladenc -192 -output c:\data\sounds\mp3 track*.wav
```

If you're not comfortable with the command line, the Windows version of BladeEnc will also let you drop your uncompressed files onto *bladenc.exe* (or a shortcut to it) and have 128 kbps files output automatically. Note that even though the program opens a DOS shell to do its business, BladeEnc is not in fact a DOS-based program—it's a true 32-bit Windows application that just happens to run in a DOS shell.

There are also many graphical "frontend" programs that will let you operate BladeEnc through a GUI. Consult your favorite MP3 software site for details.

MP3ENC

www.iis.fhg.de/amm/download/mp3enc What may constitute the definitive encoder—the one coming directly from Fraunhofer themselves—is also paradoxically one of the more seldom-used (at least by individuals and casual MP3 collectors). The biggest reason for this is that MP3ENC (sometimes referred to as FhG, after its creator) is expensive. At this writing, it cost $199 to purchase Fraunhofer's very high quality encoder (a demo version limited to outputting 30 seconds of MP3 is freely downloadable from the URL above). And for that price, all you get is a command-line tool with no graphical user interface (which, many feel, is a blessing, not a curse).

While this tool may be both more expensive and more difficult to use than the popular encoders available from large vendors, it will give you some of the best quality you're liable to find anywhere, and offers all possible encoding options. Even if you don't think you'll be able to afford a registered version of MP3ENC, download the demo and run some listening tests against other encoders—you'll probably be able to hear the difference.

While the command-line switches are simple to use, you'll need to know at least a few of them before beginning—be sure to read MP3ENC's documentation thoroughly. Assuming you've already ripped or created a WAV file, basic usage is:

```
mp3enc -if filename.wav -of filename.mp3
```

where `-if` represents the input file and `-of` represents the output file. The default bitrate is an assumed 128 kbps. If you want to output a different bitrate, you'll need to specify it in bits per second, not kbps. In other words, use:

```
mp3enc -br 160000 -if filename.wav -of filename.mp3
```

to generate a 160 kbps MP3 file. Because MP3ENC does not spit out a progress report to the command line by default, you'll probably want to use the `-v` flag (for verbose) most of the time. MP3ENC also takes a `-qual` flag with a value of 0 to 9, which lets you control the balance between encoding speed and quality by toggling various aspects of the algorithm on and off. To generate the best possible MP3 files without worrying about speed or filesize, use something like:

```
mp3enc -qual 9 -br 320000 -if filename.wav -of filename.mp3
```

Yes, it will be slow, but the quality will be unparalleled. Conversely, set the quality to 0 and watch how fast it whips along.

MP3ENC is available for Windows 95/98/NT, Linux, Solaris, SUN OS, IRIX, and Alpha/OSF1.

Bypassing Protection Schemes

If you want to save audio streams moving through the system to which you don't have direct access (such as Internet broadcasts that your playback application won't let you save), search the Internet for the Wave To Disk, or WTD driver. This driver masquerades as an audio driver, so that any audio signal normally headed for your sound card is stored in a WAV file rather than being played. Some music pirates use the WTD to bypass copyright protection mechanisms, though it doesn't work in all versions of Windows. If you just want to encode RealAudio or G2 RealMedia files into MP3 format, search the Internet for the $30 Ra2Wav utility, which supports file format conversion either singly or in batches.

MacOS

It's a toss-up: For a long time there, the most popular ripper/encoder solution for MacOS was Xing's AudioCatalyst, usage of which is virtually identical to the Windows version of the product—just set your encoding options in the preferences and drag audio files into the encoding window (see the previous section for details on using AudioCatalyst).

SoundJam

www.soundjam.com The advent of SoundJam MP for the Mac (see Chapter 3) has presented a serious threat to the dominance of AudioCatalyst, thanks to its extreme ease of use and neat integration with the SoundJam playback system. The SoundJam "Converter" (Figure 5-3) is capable of converting WAV or AIFF files, tracks from CDs, or even QuickTime movie soundtracks to MP3. To convert from CD, insert a disc and choose Audio CD from the Window menu. Select the track numbers you want converted, and click the "Add to Converter" button. Alternatively, you can drag tracks directly from the Finder—or even the whole CD icon on the Desktop—into the Converter window. Click the Configure button and establish preferences for the usual range of MP3 output options: bitrate, mono/stereo, samplerate, and (very important) whether or not to use ID3 tags. Click the Start Converting button and let 'er rip.

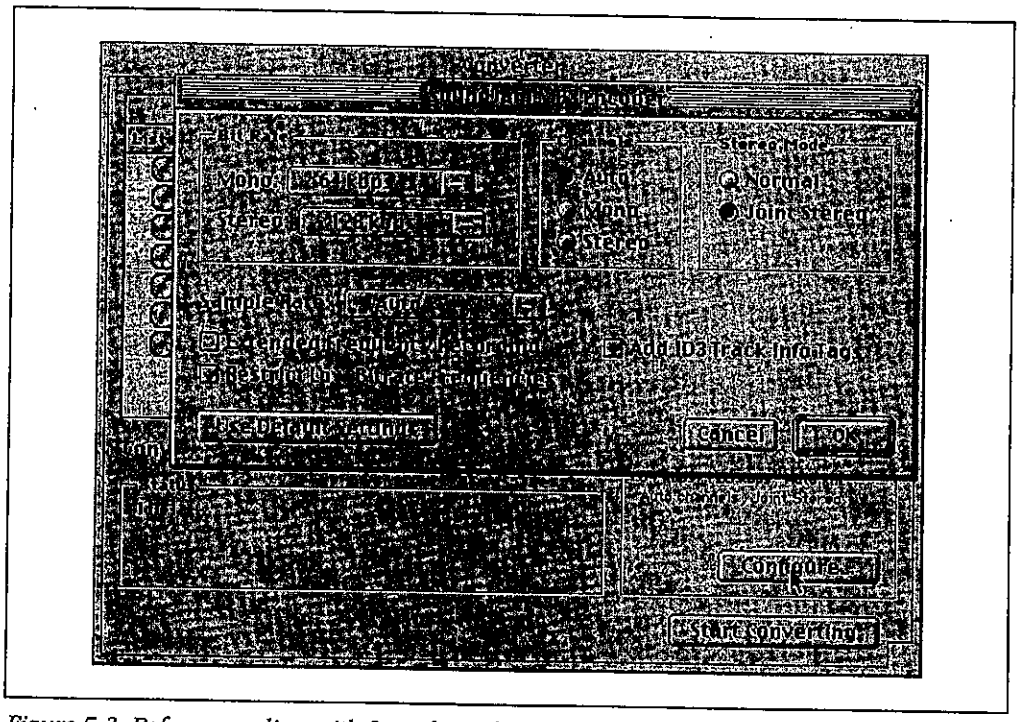


Figure 5-3. Before encoding with SoundJam, be sure the "Add ID3 Track Info Tags" option is enabled

Audiophiles take note: By default, the SoundJam encoder discards frequencies above 16kHz, for faster encoding. While this is fine for most purposes, people who intend to listen to their MP3 collection on systems other than their computer speakers should select the "Extended Frequency Recording" checkbox, which will cause the SoundJam encoder to reproduce frequencies all the way up to 22kHz. Note, however, that doing so will subtract from the amount of data space per frame available to storing information in the more audible lower frequencies. Therefore, you should only use this option when encoding at higher bitrates, or you'll end up with worse-sounding MP3 files, not better. On the other end of the spectrum, you can save additional disk space by selecting the "Restrict Low Bitrate Frequencies" checkbox, which throws out additional data at the high end *only* when you're encoding at 96 kbps or lower.

Oddly enough, the Converter's preferences panel does not include an option for setting the output folder into which encoded MP3 files will be placed. By default, all MP3 files will be dumped into the Music folder inside the SoundJam application folder. However, you can change the default location for all SoundJam files from the main preferences panel.

SoundJam is also capable of converting source files to AIFF, which may be useful if you're preparing to burn an audio CD and have an older version of Toast that doesn't convert from MP3 to PCM automatically, or if you want to be able to open your MP3 files in an audio editing application for tweaking. MacOS users may also want to check out Fraunhofer/Macromedia's SWA Plugin for SoundEdit 16, as well as the independently developed Mpecker (*www.anime.net/~go/mpeckers.html*) encoder/decoder combination and a shareware utility called TrackThief.

Developers of the once-popular, free Mpecker encoder were contacted by Fraunhofer/Thomson Multimedia in August of 1999, in search of their royalty fees. As a result, Mpecker was taken out of public distribution until an agreement could be reached. By November of the same year, the situation was unchanged and it was unclear whether Mpecker would ever surface again.

N2MP3

www.n2mp3.com Mac audio pros gravitate toward another all-in-one ripping/encoding solution for the Mac called N2MP3, which many feel generates superior quality encodings to those generated by SoundJam. The encoder built into N2MP3 is based originally on the LAME encoder, with additional proprietary optimizations.

The company calls their version, "Proteron." N2MP3 sports a very clean, logical interface, all of the power options you would expect, full integration with the Finder, plus VBR, CDDB, and ID3 support.

Linux

There are two rippers for Linux that bear mention: cdparanoia and cdda2wav. This, however, does not mean there are only two ripping *solutions* available for Linux. Since anyone can build scripts around these two core tools, or GUI interfaces on top of them, there are zillions of ways to turn CDA tracks into WAV or MP3 files from within Linux. And of course, these aren't the only rippers available for Linux—just the most popular. As far as encoders go, users generally consider LAME to render the highest quality, although again, there are a multitude of choices out there.

cdparanoia

www.xiph.org/paranoia/ Of the two, cdparanoia is the more popular (though cdparanoia began life as a series of patches to cdda2wav before branching off in its own direction). The tool is so-named because of its author's unflagging attention to detail. Just read cdparanoia's FAQs to see what I mean; cdparanoia corrects for every possible glitch that can crop up during the digital extraction process. To preserve optimal sound quality, cdparanoia is insistent about always retrieving data in digital form, never ever letting audio data near the sound card where it might be converted into analog signal during processing. Given a clean CD, your rips are guaranteed to be 100% accurate, with no skips or pops due to data transfer errors.

To compile cdparanoia, grab the sources from *www.xiph.org/paranoia/* and type:

```
./configure
make all
```

Once installed, you can test the tool by inserting an audio CD and typing:

```
cdparanoia -Q
```

This will query your CD-ROM device and return a listing of all tracks. To rip a single track, use:

```
cdparanoia 4
```

to rip track 4, for example. An ASCII progress indicator will give you a readout of the ripping process, while a series of symbols will inform you if errors are encountered (see cdparanoia's documentation for details on these symbols). When complete, a file called *cdda.wav* will be dumped in the current directory. Of course,

there are zillions of command-line options you can use for more precise control. To specify the output filename, use:

```
cdparanoia 4 ~/yourname/testoutput.wav
```

To do batch encodes, use the -B flag, followed by a directory name (remember to append the directory name with a trailing slash, or you'll get unexpected results). The -B flag, in turn can be handed arguments to specify a series of tracks. For example:

```
cdparanoia -B 3- ~/waxwing/music/mp3/
```

will output every file from track 3 to the end of the CD to *~/waxwing/music/mp3/*. cdparanoia is also capable of extracting specific portions of tracks. To grab just the audio between time points 1:32 and 3:47 in track 8, use:

```
cdparanoia "8[1:32]-8[3:47]" output_file
```

Here's a command sequence you can use to simultaneously rip and encode tracks 1–10 on the current CD at 160 kbps, using cdparanoia in combination with the l3enc encoder (not covered here, but similar to other tools described in this section):

```
for i in 1 2 3 4 5 6 7 8 9 10
    do cdparanoia $i $i.wav
    l3enc -br 160000 $i.wav $i.mp3
# Comment out the following line if you don't want
# the wav source files deleted after encoding.
rm -f $i.wav
done
```

Because of all the error checking and extreme "paranoia," cdparanoia will always run at a maximum of 1x speed. If you're wondering why your 36x CD-ROM drive is ripping so slowly, add the -Z flag to disable paranoid checking and get the full speed out of your drive.

cdparanoia was available for Linux and BeOS at this writing, with xBSD, IRIX, and Solaris next on the "to-do" list.

cdda2wav

Available at nearly any Linux software site. If you prefer speed over accuracy, you might want to try the somewhat faster cdda2wav, instead. Don't leap to the conclusion that your rips are going to be full of errors if you use cdda2wav; rips made from the vast majority of drives with scratch-free audio CDs are going to come out fine. You just have to ask yourself how paranoid you are about these things. cdda2wav functions very similarly to cdparanoia, with a few exceptions in its command-line syntax. See the included file *HOWTOUSE* for details.

When compiling cdda2wav, note that the default Makefile generated by *./configure* will point to a SCSCI CD-ROM device. If you have a SCSI CD-ROM, make sure the appropriate line in the Makefile is pointing to the correct device before compiling. If you have an ATAPI device, comment out the SCSI line and uncomment the line beneath it, pointing to */dev/cdrom*. Save the Makefile, then run `make` and `make install`.

LAME

www.sulaco.org/mp3/ Widely considered to render the most consistently high quality of the available encoders, LAME (Lame Ain't an MP3 Encoder), is not an encoder in and of itself. To get around issues surrounding encoder licensing issues, LAME is, as its name says, not an encoder. By itself, it is incapable of producing an MP3 stream. Instead, it's a set of patches against the ISO reference sources. In other words, it gets by on a technicality. Its authors have put a great deal of work into making LAME faster and of a higher quality than the ISO sources, and the work shows. The sources compile on GNU/Linux, Unix, Windows, MacOS, BeOS, AmigaOS, and OS/2.

Once installed, you can discover all of LAME's command-line options by typing `lame` with no arguments. The following are some basics.

Encode a WAV file to a 192 kbps MP3 file called `funky_popsicle.mp3`:

```
lame -b192 funky-popsicle.wav ~/funky_popsicle.mp3
```

Do the same, but use a VBR setting of 7 instead (when using VBR with LAME, remember that the `-V` parameter specifies an allowed threshold of distortion, so `-V1` is better than `-V7`). Note that here we're using both a VBR setting and specifying a bitrate—if you specify a bitrate, the bitrate will never drop below that bitrate, even in totally silent passages:

```
lame -V7 -b128 funky-popsicle.wav ~/funky_popsicle.mp3
```

Do the same, but specify that the input source is 22kHz and was ripped on a Mac and therefore needs its bytes swapped from big endian to little endian (the `x` flag takes care of that):

```
lame -x -s22050 -V7 funky-popsicle.wav ~/funky_popsicle.mp3
```

Later versions of LAME even include some ID3 tagging features, so you don't have to use id3ren separately:

```
lame -b192 --tt "Funky Popsicle" --ta "The Jaw Droppers" \
--tl "A Day in the Factory" -ty 1964 funky-popsicle.wav ~/funky_popsicle.mp3
```

LAME also has a "fast mode" option which lets the encoder run at up to 2.5x its normal speed, with some quality degradation (though not as much as you might expect). Still, I recommend reserving this `-f` option for use only with spoken word, broadcasts, and very badly recorded music. To learn how the `-f` option works, see "GOGO-no-Coda" section, next.

GOGO-no-Coda

www.kurims.kyoto-u.ac.jp/~shigeo/gogo_e.html A high-performance variant of LAME called GOGO (available for Windows, Unix/Linux, BeOS, and other x86 platforms) had just appeared as this book was going to press, and was impressing the heck out of users with its blazing speed. GOGO is written to take explicit advantage of MMX (Enhanced), 3D Now!, and SSE instruction sets if your processor supports them. Many portions of the LAME code are also rewritten in assembly language. In addition, GOGO is the only encoder available (at this writing) that takes advantage of multiple processors. If you use a dual-proc machine, you'll definitely want to check this out. By default, GOGO is about twice as fast as LAME. With two processors, you'll get virtually identical encodings four times faster than LAME. The important thing to note about GOGO is that it achieves its speed not by cutting corners in the encoder, but by taking better advantage of modern processors, especially in the most math-intensive areas of the codec.

In addition, GOGO supports the command-line flag `-nopsy`, which disables the psychoacoustic model and speeds up encoding even more (way more). But how can you have a valid MP3 file that doesn't use the psychoacoustic model? The `-nopsy` option is the same as the `-f` (fast mode) in LAME. Recall from Chapter 2 that the MP3 codec computes masking thresholds between frequency sub-bands in order to decide how many bits should be allocated to each band. The `-nopsy` option causes the codec to stop computing the masks. Instead, it assumes a constant, predefined masking threshold for each sub-band, regardless of the content being encoded. So in essence, `-nopsy` is a hack. The psychoacoustic model isn't really being thrown out entirely as it is being cheated of its intelligence. This enables GOGO to encode at blazing speeds, with some loss of quality. Depending on your expectations, you may be surprised to learn just how good `-nopsy` encodings are, though I recommend avoiding this option for most purposes—quality should almost always be your paramount concern.

Get a Grip

www.nostatic.org/grip/ If you prefer to do things through a friendly, graphical interface, Grip is an excellent GUI interface onto either cdparanoia or cdda2wav (your choice), written by MP3.com employee Mike Oliphant. Grip doesn't stop with ripping, however. It also functions as an interface onto encoding tools such as BladeEnc (see the Windows section for BladeEnc coverage) and ID3 tagging

Grabbing Streams

Linux users may find this trick useful for stuffing any audio moving through the system into a raw audio file. The link `/dev/audio` normally sends all audio signal straight to your sound card. By replacing this link with an actual file, all audio will end up in that file. Try this:

```
mv /dev/audio /dev/audio-bak; touch /dev/audio
```

Now start your favorite audio application, and anything it plays will be dumped into the file at `/dev/audio`. You can later copy this to a new filename, replace the original link, clean up or edit the audio file in an audio editor, and encode it to MP3. Some users circumvent copyright-protected formats in this way, although the technique does not always work, especially if the audio stream is pointed at `/dev/dsp` rather than `/dev/audio`.

If you have a copy of the text-mode lynx browser on your system and want to capture an MP3 broadcast stream to disk, try this:

```
lynx -source http://ip_address:port/ > somefile.mp3
```

tools such as id3ren (Chapter 4). In addition to handling virtually every command-line option supported by these tools, Grip also supports CDDB connections, so your tracks will end up with reasonable filenames and ID3 tags right off the bat.

Grip requires GTK 1.2 or later. See *www.gtk.org* for details, as well as the GTK installation notes in Chapter 4.

Before running Grip for the first time, su to root, since you'll need to have appropriate permissions to establish Grip's configuration details. Click on the Config tab to establish the path to your ripper executable, specify any command-line parameters you'd like to pass to it, the ripping file format, the CPU-intensiveness at which Grip should run (i.e., its "niceness"), the preferred output directory, and the paths to your MP3 encoder and ID3 tools. You also have complete control over the file naming convention and output directory. By default, Grip will create a new directory for every new artist and within that, a new directory for every new album you rip. You may want to begin by specifying some location within your home directory; otherwise, output will be placed in the current directory.

Once you've saved your configuration, return to the Tracks panel. If you're connected to the Internet and have an audio CD inserted, you'll find that the current CD has been properly detected and that all of its tracks are properly listed, as

shown in Figure 5-4. You may want to click the pencil icon in the lower toolbar to display the Disc Editor, in case you want to change parameters such as Genre or Year. When finished, move to the Rip tab, click the Rip button, and cdparanoia or cdda2wav will kick into action. If you've also got an MP3 encoder installed and properly specified in the Config panel, your tracks will be simultaneously ripped and encoded.

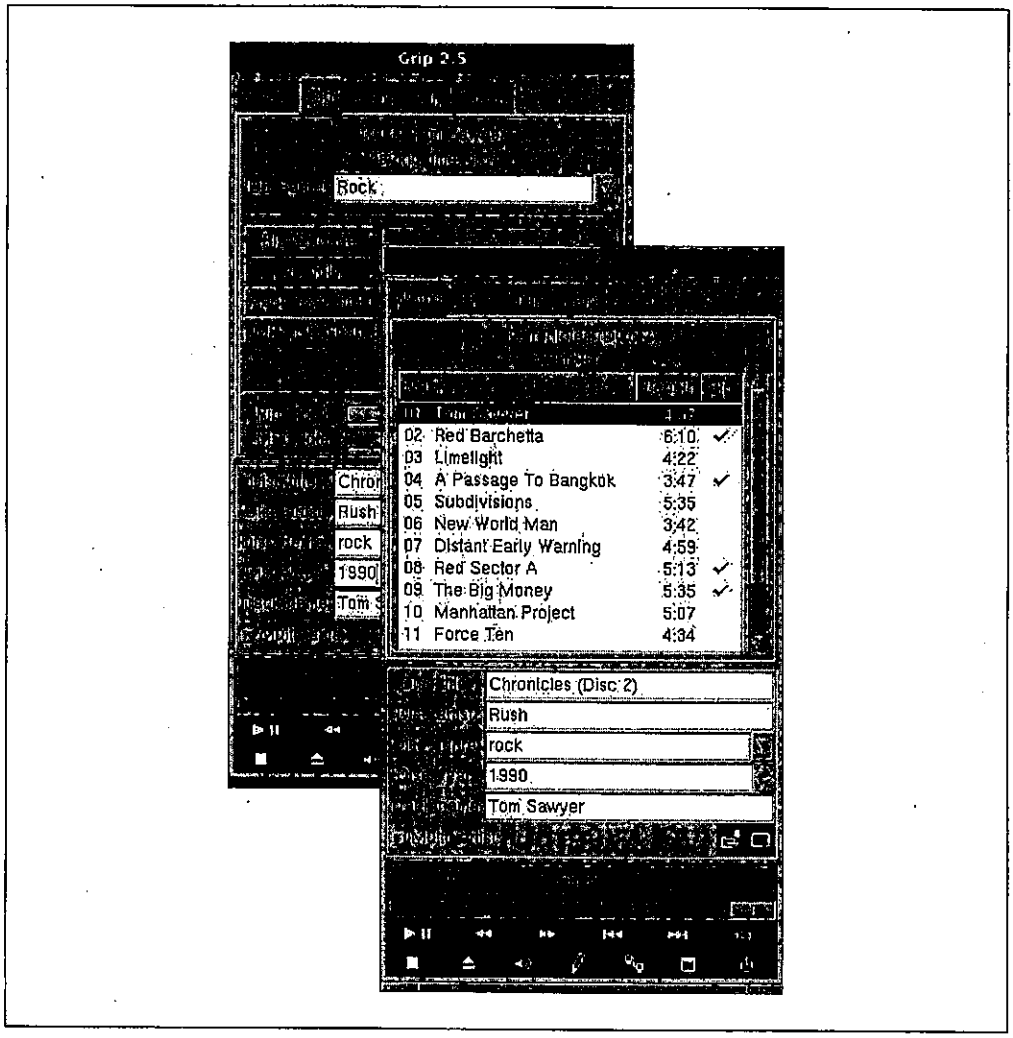


Figure 5-4. Grip for Linux serves as a GUI frontend to command-line tools

RipEnc

www.asde.com/~mjparme Following the same philosophy as Grip and other graphical front-ends, users who don't run X (or just prefer the command line) will appreciate some of the shell scripts out there that do a similar "all-in-one" job. One

of the most popular and complete shell scripts for MP3 ripping and encoding on Linux is RipEnc—a monster of a script which interfaces with your choice of cdparanoia or cdda2wav for ripping, BladeEnc, 8hz-mp3, or l3enc for encoding, and the command-line CDDB lookup tool "cda." This makes for a complete, all-in-one tool comparable in power to Windows applications like AudioCatalyst, but consisting only of "glue" that holds the classic Unix "atoms" together. Oh, and like most Linux tools, RipEnc is free.

Getting RipEnc set up for the first time can be a bit tricky—you've got to make sure all of your ducks are in a row. First of all you'll need to make sure that at least one of the rippers and one of encoders discussed in this section are installed and in your path (actually, RipEnc can run without an encoder if you just want extract uncompressed audio).

If you want to save yourself the trouble of having to manually rename and add ID3 tags to your tracks, make sure you've got full CDDB support: Head over to *metalab.unc.edu/tkan/xmcd/xmcd.html/* and download the package appropriate for your platform, make the `install.sh` script executable, and run it. You'll be led through a series of questions regarding the CDDB servers you want to connect to and the make and model of your CD-ROM drive.

In RipEnc's launch menu, tell the program which ripper and which encoder you want to use. The first time you run RipEnc, be sure to check the path to the `xmcd` library on your system, which may not match the default path provided (use the `find` or `locate` commands to track this down). Once you start ripping, you'll be prompted to enter any ID3 fields that could not be determined from the CDDB lookup. If you opted not to encode the entire disc, you'll be asked which tracks to encode. The rest is gravy, baby... yeah!

RipEnc has also been adapted to work under BeOS, though it functions somewhat differently. See the BeOS section below for more on that..

BeOS

Users won't find many ripping tools available for BeOS, because it doesn't need any. Since the operating system supports alien filesystems with the addition of a single device driver module, Be provides a "cdda-fs" driver with every copy of the operating system. The cdda-fs driver basically creates a "front" for CDA tracks, making them appear to the operating system as WAV files. Insert an audio CD into your CD-ROM drive and it will appear on the Desktop just like any other mounted volume. Inside, you'll find two folders: One called *CDA* and another called *WAV*. The CDA folder contains the standard CDA handles described earlier, while the WAV folder contains actual WAV files, which can be played immediately without ripping them first. These WAV files can be used directly by the user or accessed

from the command line or from within third-party applications, and can of course be dragged off the CD and into the Tracker. Performing this mapping of CDA to WAV incurs no delay; nothing is being translated. The filesystem driver is simply mapping the CDA pointers to actual audio data, granting you direct access. BeOS encoders can therefore encode to MP3 without having to rip tracks first.

This capability is useful for more than just MP3 encoding—you can also play these WAV files in any BeOS audio player. In fact, if you use a tool like SoundPlay (Chapter 3), you can even play three or four tracks simultaneously, backwards or forwards, at variable speeds, *directly off the CD.*

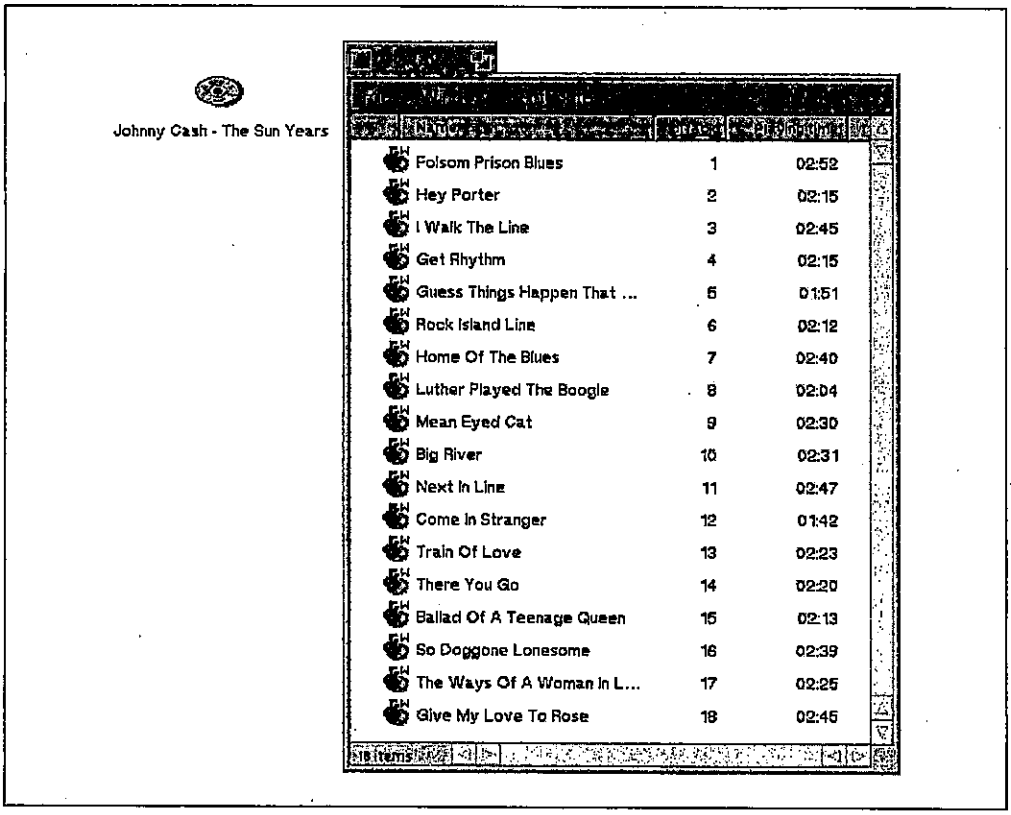


Figure 5-5. Tracks on audio CDs in BeOS appear to the Tracker as WAV files

As you can see in Figure 5-5, the cdda-fs driver doesn't just make the files masquerade as WAV files—it also works in conjunction with a small daemon called *cddblinked*, which runs in the background and automatically makes contact with the CDDB (or other database) whenever you insert an audio CD (if you have an

Internet connection). The CD volume is then mounted with the artist and album name, and all tracks on the disc appear with their actual song names in the Tracker. This means that BeOS ripping/encoding tools don't necessarily have to do CDDB lookups of their own—they can just extract the artist, album, and track names directly from the filesystem.

If you want to be really hardcore about it, you can rip tracks from audio CDs without the help of cdda-fs, by using the system's built-in "play" command (which offers a "save track to file" option), or use the BeOS port of cdparanoia, described earlier in this chapter.

Be has licensed the MPEG codec from Fraunhofer and Thomson, and includes MP3 encoders and decoders in the system. This means that, in addition to ease of ripping from CD, you can save directly to MP3 from any BeOS audio mixer/editor/sequencer application that works with the media kit.

CDPlayer and SoundRecorder

If you prefer to do this graphically, or just want to rip portions of tracks, the graphical CDPlayer application included with BeOS also has a built-in ripper and encoder. Just click the floppy disk icon in CDPlayer's interface, select a track number, and use the triangular sliders to select the portion of the track you want to rip (the default setting rips the entire track). Select an output format to save your selection as WAV, AIFF, RAW, or directly to MP3. Similarly, the system's built-in SoundRecorder application will let you crop arbitrary segments of audio from any track and right-drag them to the Desktop. Choose an output format from the context menu and you'll be cropping and encoding audio to MP3 simultaneously.

MediaPede's UltraEncode

www.mediapede.com The easiest to use of the BeOS ripper/encoder solutions is an all-in-one package from MediaPede called UltraEncode, which sports an intuitive interface and fast encoding thanks to Be's inclusion of an MP3 encoder in the system. Usage is straightforward: Establish your CDDB server, bitrate, and destination directory in the preferences, insert a CD, and its track names will appear in the Extractor Queue. Establish a genre and year for the CD from the Settings menu, and double-click any track name for which you want to edit details.

Check off the tracks you want to encode, or tap Alt+A to select all or Alt+D to select none. Click the Start button and ripping/encoding will take place simultaneously (UltraEncode is heavily multithreaded). You can show/hide any of the interface panels by clicking the "switch" widgets at the right of the interface (see Figure 5-6).

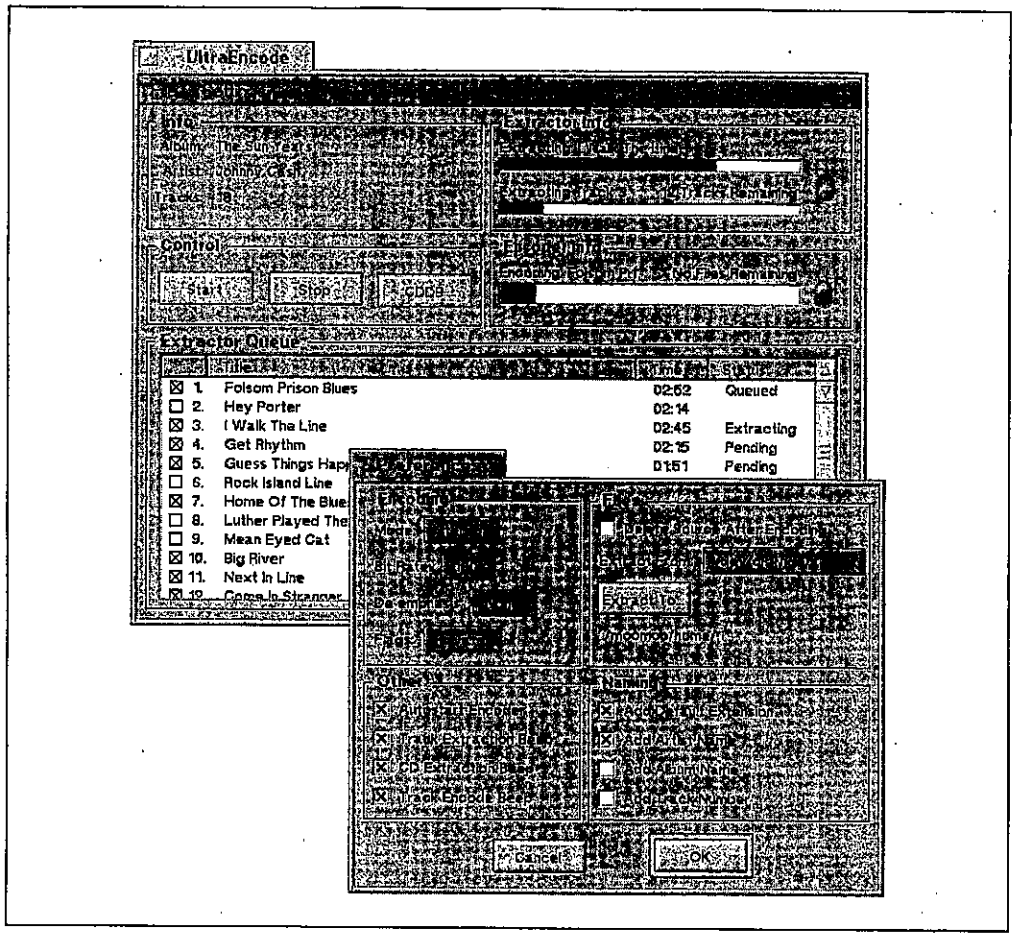


Figure 5-6. MediaPede offers an all-in-one ripping and encoding package for BeOS users

RipEnc

www.betips.net/software/ RipEnc for BeOS is a distant cousin of the Linux version mentioned above. The BeOS version (by this book's author) started life as a straight port of the Linux version, but was later gutted and rewritten from scratch to take advantage of cdda-fs. The BeOS version is therefore "ripless"—tracks are encoded directly as they appear in the Tracker, without being extracted from the CD first. Like the Linux version, the BeOS version is a "frontend" to a small collection of command-line tools, including id3ren (covered in Chapter 4), the GOGO encoder (covered ealier in this chapter), and a tool called id3attr, which copies ID3 tag information into BeOS filesystem attributes which can be queried on to create custom playlists quickly. RipEnc for BeOS also makes use of "folder templates" so that generated folders full of MP3 files are pre-organized in a customizable fashion, showing ID3 tags as attributes in the Tracker (as shown in Figure 5-6). RipEnc also supports optional manual track naming, a "play when done" option, variable bit rate mode, CD automounting, and other goodies.

The nice thing about using frontends such as RipEnc for Linux or BeOS is the fact that the user can choose their favorite tools to be used with the system. For example, you can choose to use a different encoder while still retaining the same overall encoding framework. And because shell scripts are easy to edit, users can dig in and make modifications if they wish.

Other popular ripping and encoding tools for BeOS include RipChord, KizunoMP, GOGO Gadget, 8hz, and KonaKoder. You'll find these tools and more at *www.be.com/beware/* or at *www.bebits.com.*

Ripping from Other Sources

Of course, the audio you want to encode isn't always going to originate on audio CDs. Many of us still have huge music collections stored mostly on LPs, cassettes, DAT, 8-tracks, reel-to-reels, and possibly other media. MP3 represents a great opportunity to finally create a permanent digital archive of these collections... or at least the best of them (most people would have to be horribly masochistic to dedicate themselves to the digitization of every track they own). The good news is that encoding your existing collection in this way is do-able. The bad news is that it's going to require quite a bit more manual labor than encoding from CD. Since you can't stick an LP into your computer and let it twirl, you've got to go to the extra step of patching your home stereo into your sound card and manually cueing everything up properly. And of course, there are no batch-ripping/encoding solutions available for digitizing 8-track tape collections. Finally, you may have to separate songs manually when encoding from analog sources—LPs don't have any real awareness of when one track ends and the next begins. Without special software (see the following section), you'll have to do all of that yourself in an audio application prior to encoding.

The term "ripping" doesn't exactly apply when talking about sources other than compact disc—nothing needs to be ripped since the source isn't hidden, as it is with .CDA files. We use the term "ripping" in a somewhat looser sense in this chapter, though it might be more accurate to refer to "extraction."

If you're a recording artist, you'll probably already have your repertoire stored on tape, DAT, or another medium, so this section applies to you as well. Of course, it's possible to connect a microphone or other instrument directly into your sound

card and save the signal direct to disk, and you may even have a fancy-schmancy sound card that handles multiple inputs and lets you do all kinds of preprocessing with dedicated DSPs, etc. But even if you are a digital-friendly musician, you're still going to want to do all of your mixing and mastering in dedicated multi-track audio software, creating a final stereo mix before going to MP3. In other words, while recording live to MP3 is a possibility, it's not often done unless you're capturing a live performance—and even then, you'll probably want to store that unencoded first.

When recording from LP, cassette, or other analog sources that don't have the same crispness at the high end, try using an equalizer to boost the higher frequencies by a couple of dB before encoding. This can help to alleviate some of the "swishy" effects for which MP3 is known. However, increasing high frequencies on LPs and tapes will also result in more noise, so there's a trade-off here. As always, you'll get better results from analog sources by using higher bitrates.

Direct to Hard Disk

As described in Chapter 4, a lot of audio hardware, including most sound cards, deal with two possible input specifications: Line level and Mic level. Mic input is used for amplifying low level input, which mostly consists of that coming from microphones. Line input handles signal from audio equipment with line-level outputs, such as CD players, tape decks, tuners, and so on. Most sound cards offer two or more input jacks, one Mic and one Line. If you're not sure which input jack on your sound card is which, consult its documentation. Note that phono cartridges need a unique amplification stage that converts their output to line level.

There are two ways to connect audio hardware to a sound card. You can either connect an instrument, microphone, or a stereo component's output directly to the appropriate input jack on your sound card, or you can run a line from your stereo amplifier/receiver's output terminal to your sound card (if your amplifier has an output jack—not all do). With the first method, you'll have the very slight advantage of eliminating your stereo's circuitry from the recording chain for slightly better input quality, but have to deal with the physical awkwardness of yanking stereo equipment and placing it near your computer. The second method incurs a little extra circuitry, but may be more convenient. In all cases, you'll get better results whenever you can minimize the length of cable runs.

Record What You Hear

Note that most of the same techniques outlined above also apply if you want to record other audio streams moving through your system, such as those coming from Internet broadcasts (any format). The only difference is that rather than fiddling with cables and jacks, you'll need to set your system's audio mixer to trap all signals. Look for a setting labeled something like "Speaker Out" or "Record what you hear." This is different in every operating system, and some sound card utility installations override your system's default settings, so you may need to use your sound card's software rather than the operating system's mixer to establish this setting. In any case, it should be possible to save any incoming signal to WAV or AIFF, clean it up in an audio editing application, and encode the resulting audio file to MP3 normally. Some audio pirates use this technique to circumvent protection schemes built into Internet broadcast formats.

Alternatively, you can use the tape output jacks on your amplifier, which will let you use the amp's switching facilities (such as the Tape Monitor button). If you do this, the volume control on the amplifier won't affect the level of the signal being sent out, so you can change the volume of the music you're hearing without affecting the level of the recording being made.

Once you've got your equipment hooked up, you need to be able to save the signal coming into your sound card to hard drive. While most operating systems come with basic audio recording software, you'll probably find that this software isn't quite up to the task. Since you may need to edit out silent spaces at the beginning and end of various tracks, or apply noise-masking, equalization, or other effects to the tracks you record prior to encoding, I strongly recommend looking into the third-party audio software options available for your platform. Usage of general audio recording software is outside the scope of a book on MP3, but popular applications include CoolEdit Pro or SoundForge for Windows, Pebbles or 3dsound for BeOS, Deck or Peak for the Macintosh, and fOX Mixer for Linux. See the section "Encoding from Analog Sources" later for more information.

If you do want to encode directly to MP3 from audio input, some of the more complete tools—such as MusicMatch Jukebox—are capable of doing so. In MusicMatch, pull down Options → Recorder → Source and choose from Line In, Mic, or System Mixer. Set your encoding preferences as usual, hit Record, and you'll be encoding from that input.

About S/PDIF

If you want to make sure your transfers stay 100% digital, or if you need to get audio data out of digital devices such as DAT players, look for a sound card with a Sony/Philips Digital Interface Format (S/PDIF) port, which usually looks like a standard RCA jack and can be connected directly to the S/PDIF port on the device. You'll also need special software (generally supplied with the digital device) to manage the transfer. All transfers will be 100% perfect; none of the issues that make CD ripping sometimes problematic affect S/PDIF transfer.

Ripping from DVD

The audio tracks on most DVD disks are stored in the copy-protected VOB format to prevent ripping. If you should be in possession of a DVD disk to which you have legal copyright access, you can bypass this copy protection by going through the sound card—just plug the DVD player's output directly into the sound card and record from that channel with an audio-editing application. If you want to go for higher quality, run the S/PDIF output of the sound card into a digital amplifier, then from that amplifier's output to an external CD recorder. You can take the analog output from a DVD player, run it through analog-to-digital conversion to output a WAV file, and encode that. If you have a digital sound card, you can take the digital output from the DVD player, run it through the sound card's digital input, and generate a WAV from that.

The fact that a group of Linux hackers had succesfully unlocked the DVD security mechanism in late 1999 may mean that tools for getting signal from DVD to hard disk may become available by the time you read this; it was too early to tell at this writing

Encoding from Analog Sources

Many thanks to technical editor Bruno Prior for his contribution of this section.

There are two basic ways to encode from analog sources such as LP or tape: Either use an "all-in-one" solution to encode directly from the audio source to MP3, or record the source to an uncompressed audio file (WAV, AIFF, etc.), preprocess in a dedicated audio editor, and compress it with an MP3 encoder. While it may seem at first like all-in-one tools would be the easiest route, this often turns out not to be the case, for a variety of reasons. For the purposes of illustration, we'll look at one example of each type.

MusicMatch Jukebox

www.musicmatch.com We've already covered the basics of MusicMatch Jukebox in Chapter 3, but here we're going to take advantage of one of Jukebox's more advanced features—its capability of encoding from your sound card's line-in jack. To be more consumer-friendly, MusicMatch refers to encoding processes as "recording," so we'll use that term here, even though it technically isn't quite correct.

To enable the Recorder, make sure the "Show Recorder" option is selected under Options → View. To configure the Recorder for line input, navigate to the Recorder configuration page via Options → Recorder → Settings and select Line-in from the Recorder Source picklist. Select your Recording quality (bitrate) from the options on the left. Click the Songs Directory button and specify the folder and filename formats under which your recordings should be stored.

You may also want to click the Advanced button to specify whether tracks should fade in or out, which can be useful if the actual fade in or out on the source material is corrupted with vinyl crackle or tape hiss. You can also choose to offset the start of each track, (which is useful to account for gaps between tracks), and to normalize your recordings.

The MP3 encoding selection box will let you choose between Joint Stereo (the default), Stereo, and Mono. As described earlier, you probably want to select Stereo here, unless you want to keep file sizes down. This can be particularly important when recording from tape, since left and right stereo channels on some tapes can be slightly out-of-phase, especially with older music. Combining the lower frequencies of out-of-phase channels, as Joint Stereo does, can produce significant distortion—the only way to avoid this is to use simple Stereo encoding.

The most important section in Advanced dialog is the Auto Song Detect option, which is intended to let MusicMatch Jukebox detect gaps between songs so you don't have to stop and start recording for each track on an LP or tape. If you know the approximate or minimum length of gaps between songs you're recording, you can help MusicMatch make a better educated guess as to where songs begin and end.

You'll notice that the Jukebox recording interface now looks slightly different than it does when ripping from a CD. Sample album title and artist names will have been added, and only one track will be displayed, awaiting the name to be entered. Enter real album and artist names and give a meaningful name to the first track. With your turntable or tape deck attached to your machine as described earlier, start the music source playing and hit Start on the Recorder. If you chose

the Auto Song Detect option, it will attempt to break the input into tracks wherever it finds a gap in the music. If it doesn't find gaps, Jukebox will output the entire audio stream until you press Stop.

There are a number of problems you may encounter when recording from an analog source. For example, Jukebox does not start recording immediately when you press Start. Instead, it listens for the start of the song before recording begins. This is fine if the song starts assertively, but if it fades in or starts with a quiet intro, you'll probably lose the start of the song. You can get around this by selecting the Delayed Record option in Recorder Settings and then specifying Start Immediately. The side effect of this technique is that you'll have to stop and start recording for each track, whether or not you have selected Auto Song Detect.

Regardless whether you have Start Immediately enabled, Auto Song Detect may introduce other problems. For instance, ASD simply looks for gaps longer than the specified Gap Length which do not contain sound levels above the specified Gap Level. Unfortunately, most analog sources contain quite a bit of noise, so you'll have to specify a fairly generous Gap Level—probably at least 10%. But if your music encompasses a significant dynamic range, you may end up with split tracks, as Jukebox may mistake quiet passages for song gaps (try recording Queen's "Bohemian Rhapsody" from vinyl for a splendid example of this). Fade-outs may also be prematurely truncated.

Even if you can get past these gotchas, you'll still have some work to do when recording is complete. You now have a series of MP3 tracks, but only the first one will have a meaningful name, while the remaining tracks will have names like "Line-in track 2," etc. You're going to have to rename the files and edit the ID3 tags by hand for them to be of any use. With any luck, a future version of Jukebox will offer a more intelligent interface for handling this in advance of recording.

While Jukebox does a pretty good job as far as it goes, you may end up deciding that you need a more sophisticated tool for reasons like this. LineRipper, for example, may be better suited to meet your needs.

LineRipper

leo.worldonline.es/mpicarth/ LineRipper exists for the sole purpose of encoding from Line-in (although it is also capable of conventional encoding from WAV files, and supports a variety of encoders, including LAME and GOGO. Conveniently, LineRipper lets you specify track names for all the tracks to be recorded beforehand. More importantly, LineRipper lets you specify different artists for each track, which gets around problems commonly encountered by encoders trying to deal with compilation albums. For the real power user, LineRipper can separate tracks either by listening for gaps or by reading a table of track lengths input by the user before encoding begins. While this takes more effort up front, it offers a better chance of accurate track separation.

LineRipper is clearly superior to Jukebox for the purpose of encoding from Line-in, but it has its own weaknesses. Even if you do type in track lengths beforehand, some slippage is still likely to occur due to differing gap lengths. If you use the auto-detect option, there's still no way to avoid the problem of truncating tracks that fade in or out when there's significant noise in the signal, and there's no way to avoid noise from sources like vinyl or tape unless you preprocess the output. This is no fault of LineRipper, but a fundamental weakness in the process of encoding from analog sources in general.

Notes on manual track separation

It may seem like a lot of effort, but in the end, the traditional route of recording to one giant WAV file, processing and splitting the WAV, and then encoding the processed tracks can be the most effective technique for creating MP3s from your tape and vinyl collection. Not that this process is not without its problems either.

The first difficulty you're likely to encounter is the fact that the audio-editing facilities which come standard with your system may not be adequate for the job. Sound Recorder (the bundled Windows recorder), for instance, can only handle around two minutes of music, which won't get you very far (the audio recorder bundled with BeOS, however, is extremely well suited for tasks like this, and can output directly to MP3; there are a number of third-party utilities for MacOS and Linux that will do the job, though I wasn't aware of any that could save directly to MP3). For most purposes, you'll need a program to record audio from Line-in. If you have a Creative Labs sound card, you should have a program on your system called Wave Studio, which goes far beyond the capabilities of Windows' Sound Recorder. Other sound cards often come bundled with similar audio editors.

To really get all the power features, you'll have to lay out the cash for a pro or pseudo-pro sound editing program. Unfortunately, the leading contenders, such as Sound Forge and CoolEdit Pro are not cheap, at several hundred dollars each. Fortunately, cheaper contenders are starting to emerge. One attractive option is CoolEdit 2000 (*www.syntrillium.com/cooledit/*) which currently costs a more reasonable $69. An optional Audio Cleanup plug-in for CoolEdit 2000 is also available, which includes the features you need for reducing clicks, crackles, and hums from analog audio. However, the plug-in is extremely slow, and doesn't offer a visual preview. A popular and even cheaper alternative is GoldWave (*www.goldwave.com*), a $40 shareware audio editor. GoldWave includes noise reduction, but click/pop/crackle removal still has to be performed by an external program.

Remember that these giant WAV files are going to consume large amounts of disk space. One 30-minute album side will take up around 300 MB at the standard 44.1kHz, 16 bit (and you probably don't want to go lower than that for your quality MP3 collection). A reasonably fast hard drive can also be useful when dealing with lots of uncompressed audio.

Notes on noise reduction

Assuming the recording process goes well, you're left with a 300 MB, crackly, hissy uncompressed audio file. The next step is to lose the crackles, hisses, etc. In CoolEdit 2000 (or other, more expensive editors like CoolEdit Pro or Sound Forge), you can probably pass the file through the included Audio Cleanup process. If your editor doesn't include such a feature, a wide variety of programs are available to do this. Some of the better ones are:

Sound Laundry (www.algorithmix.com)
: The Compact Edition offers limited functionality but has the easiest interface of any of these programs, while the full version has a confusing interface but is very flexible and powerful.

Groove Mechanic (www3.bc.sympatico.ca/badgerbytes/groove/)
: Very good output with the default values, but not as flexible as most.

Dart Pro 32 (www.tracertek.com)
: Highly configurable but slightly confusing user interface and somewhat sensitive to source file format.

Regardless of the program you choose, play around with the options for reducing click/crackle and noise/hiss until you're getting the cleanest sound possible without removing any of the actual audio. Keep in mind that it's very easy to throw the baby out with the bathwater here—remove too much signal and you'll affect the ambience of the whole piece. Then pass the file through the filter or filters and save the clean output as a new file.

A good way to make sure you're removing only the noise and not the signal (if your program supports it) is to listen to the sound being removed, rather than to the audio that remains.

Splitting large files

The next step is to chop up the clean file into its constituent tracks by selecting the in-point and out-point of a track, copying the selected area to the clipboard, and pasting the results into a new track. Most editors offer a visual image of the file, so it's easy to see the song gaps as "troughs" on the display. Jump to the gap points, zoom in, and listen carefully to make sure you're selecting exactly the right in- and out-points for each track.

Once you've got a set of clean, uncompressed tracks, all that's left is to encode them to MP3 by passing the files through your favorite encoder. Most encoders will not offer the option to tag the resulting MP3s, as they can't interface raw audio with the CDDB and therefore do not have the information they need. You'll have to insert your ID3 tags manually with a separate ID3 tagger or through your favorite MP3 player, if it includes its own tag editor.

If you named the files sensibly, try a tool such as id3ren (see Chapter 4), which can extract track title, artist, album, and other data from the filename and use that to create the tags automatically.

If splitting and encoding files manually sounds like a lot of work, a tool that will help to do this in one step is LP Ripper (*www.cfbsoftware.com.au*). Tell the program how many tracks are on the recording and it will split the file for you by checking for perceived gaps. You can adjust the start and end times of each track to a sensitivity of 0.1 second, and you can enter track titles beforehand, which LP Ripper will use to name the files. Some basic manual editing (such as removing leading and trailing noise) is advisable before passing the file to LP Ripper, but this tool can save a good deal of time and effort when dealing with giant WAV files. LP Ripper's greatest weakness is that it won't let you jump to the end of a track to adjust the end time. If you want to edit the end time of each track precisely, it's probably quicker to do it manually.

Roll Your Own Compact Discs

When compact discs first appeared, they were "black boxes"; unlike LPs or cassettes, few people understood how they worked. In the mid-'90s, writeable CDs appeared for the first time, enabling anyone to create their own audio or data CDs at home... as long as they were sufficiently wealthy. By the end of the '90s, however, the cost of CD-R (the "R" is for "recordable") devices had fallen to well within the casual user's reach, and the cost of blank CDs dropped to the point where CD-R had a lower price-per-megabyte than any other storage medium.

MP3 collections typify the kind of problem CD-R was meant to solve: What happens when the amount of data you want to keep around exceeds the amount of storage space available on your system's hard drives? True, the price of hard drives has plummeted as well, but no matter how big a hard drive you install, it won't be long before your MP3 collection is even bigger.

However, relying on compact discs as a primary storage medium probably isn't the way you want to go either. With the price of hard drives continuing to go down, some people are surprised to discover that storing large MP3 collections on hard drives is actually *cheaper* than storing them on compact disc. Let's do the math. At this writing, 28 GB hard drives could be had for around $8/GB. Assuming you encode at 128 kbps, you'll need around one megabyte per minute of music. Now let's assume a baseline price of a CD-R drive at $200 and $1 for each disc. One 28 GB disk will cost around $224 and hold around 470 hours of music. Storing the same number of MP3s on CD-R will require around 44 discs, at a cost (including the drive) of $244. Storing the same number of tracks in CD audio format will require around 380 discs, at a cost (including the drive) of $580. So to store approximately 500 hours of music, hard disk is the best cost option. On the other hand, if you double this (i.e., 1,000 hours of music), then storing MP3s on CD-R looks like the best option. But by then, the inconvenience of having to search through 90 CDs to find the track you want will probably outweigh the marginal cost benefits.

If cost is all that counts, then the balance point (where CD-R overtakes hard-disks in terms of value) is currently at just over 500 hours of music. But unless you absolutely require portability, having all your tracks on CD-R as opposed to hard disk is going to be extremely inconvenient. And remember that CD-R is not editable. Keep these considerations in mind before devoting a ton of energy and money into building a giant collection of hand-rolled compact discs.

CDA vs. Data

There are two ways to approach MP3 storage on compact disc:

Keep your files in MP3 format

Advantages

Assuming 128 kbps, you can store 650 MB of MP3s, or about 10 hours of music on a single CD.

Disadvantages

You can only listen to the stored music through an MP3 player, which usually means on your computer. However, increasing numbers of dedicated MP3 hardware devices are appearing on the market, and some of them will let you play data CD-ROMs full of MP3s.

Transform your MP3s into CD audio format

Advantages

Can be played on any CD player (and most DVD players) in the known universe (and possibly in alternate universes as well, excepting those comprised of antimatter).

Disadvantages

Doing so means uncompressing your MP3s, so they take up 10 times more space. You're back to the 74-minute storage length of regular audio CDs. And doing that often means more manual labor to burn each CD.

How you approach this dilemma depends completely on your needs, and which end of the convenience spectrum you want to land on. If you do most of your MP3 listening at the helm of your machine and just want to free up some disk space, you'll want to burn normal data CDs. If you want to create standard audio CDs you can play in the car or on your home stereo, you may want to go the distance and start burning audio CDs.

Hardware/Software Requirements

In order to create CDs of any type, you'll need a CD-Recording device of some kind. Inexpensive models, as of late 1999, could be had for as little as $200, while professional models (and CD duplicators) could set you back more than $1,000. Most people should find the base models more than adequate for their needs, however. If you want to do some detailed research on CD recorders, take a look at CD Media World (*www.cdmediaworld.com*).

You'll also need to decide whether to get a straight CD-R device or go for CD-RW. The latter type is capable of writing to a blank CD multiple times, just like you would with a hard drive. However, blank media for CD-RW is considerably more expensive than blank CD-Rs, and may not provide as much added convenience as you might imagine, primarily because CD-RW discs have a lower reflectance than standard CDs and thus can't be played back in many standard audio CD players. This makes them more suitable for data storage than for audio storage (although many late-model audio CD players are CD-RW-compatible). At a buck or two a pop, CD-Rs are cheap enough to make occasional mistakes with, even if you can't go back and correct them later on.

Burning Notes

The actual process of burning CD-ROMs is completely dependent on the burning software used. While almost all CD-R and CD-RW burners come with some kind of burning software, it's often possible to use software other than what comes with your unit, in case you don't find it up to snuff. Some users with exacting demands,

for instance, find that the Adaptec software that comes with many CD-R drives is not configurable enough, and has trouble with some scratched CDs. Many "pro" users turn to Goldenhawk's CD-recording software instead (*www.goldenhawk.com*). In any case, the process is usually pretty intuitive, and most CD burner applications come with complete documentation and wizards to guide you through, should you need assistance. In general, the process consists of nothing more than telling the software whether you intend to burn an audio or data CD, then dragging your tracks or files into a window.

If you want the audio CDs you burn to be playable both in computers and in stereos, you must remember to "close" the disc when the burn is complete. You should find an option for this somewhere in your burning software.

When shopping for a CD burner, don't get overly excited about speed without asking around online about burning quality first. While not universally true, some people find that the faster the burner, the higher the chance of bad burns, or of pops and glitches in the final CDs. With a high-quality drive such as a Plextor or Yamaha, however, fast burns can still be error-free (the author hasn't toasted a CD yet with his 8x SCSI Plextor burner). The better drives from Panasonic and Pioneer also get good marks among users for bit-perfect DAE. Also, make sure you get a drive capable of "Disk at Once," or DAO technology. This lets you burn disks without a two-second delay between tracks (not generally an issue for mixed collections, but possibly annoying for live shows and for creating segues).

When shopping for blank disks, don't lunge immediately for the 100-pack at the super-mongo discount price. The chances of bad burns (which wastes both time and money) is much higher with cruddy CDs than with good ones. Unfortunately, there are no blanket rules for determining which of the dozens of available brands is going to work best for you. Buy your CDs in small quantities until you find a brand with near-perfect, reliable results. Again, check around online for recommendations, and see *www.cdmediaworld.com* to learn all about the various dye colors and manufacturing plants.

If your CD-ROM drive is working but you're experiencing pops, glitches, or drop-outs regardless of which burning software you use, look for an option in the software that will let you decrease the burn speed or increase the buffer size. You may get more accurate transfers at slower speeds.

Making space

Depending on the CD burning software you're using and the kind of disk you wish to burn, you may need to set aside a big chunk of disk space for the process. If you're burning exact duplicates of existing CDs, this won't be an issue as you'll be copying data directly from one CD to another. If you're burning custom CDs consisting of various tracks from multiple sources, however, you'll need to make sure you've got enough temporary storage space to hold everything that's going to go on the CD. Since the capacity of a CD is 650 MB,* one of the best solutions is to create a 650 MB disk partition. Fill it to capacity with WAV or AIFF files (or MP3s, if you're burning a data CD) and you'll know exactly how much space you've got to work with. You can then just drag the entire folder into the burn window of your CD creation software, or create a "disk image" of the entire partition and burn the image to CD (images or "Disk At Once" options are especially useful if you have albums with songs that flow into each other smoothly, or anytime you want to eliminate gaps between the tracks). If you plan on burning images, however, note that you'll need twice the space, or 1,280 MB, since the image file will be exactly the same size as the total amount of data you intend to burn.

Since creating a new partition isn't a convenient option for all users, you may just have to keep an eye on the size of the folder you're storing source material in. You can learn this information in Windows Explorer by right-clicking a folder name and choosing Properties from the context menu, in the MacOS Finder or BeOS Tracker by selecting a folder and tapping Opt+I to Get Info, and in Unix/Linux by typing `du <pathname>`.

In addition, most graphical burning software will provide some kind of interface widget or status area reporting the total amount of space occupied and remaining on the pending CD by examining the contents of the burn list. Thanks to this feature, you can drag in files from anywhere on your system without needing to set aside a specific folder or partition. Your burning software shouldn't let you attempt to burn a CD with a burn list larger than the capacity of a CD.

Format Conversion

If you only want to burn data CDs, you can skip this section and just use the documentation that came with your CD-R unit. If you're burning audio CDs, read up. While some CD burning software, such as MP3 Maker for Windows, Toast for MacOS, and CDBurner for BeOS will let you drag MP3 files directly into the burner application and have everything decoded to PCM audio automatically, most CD

* Audio CDs have actually pushed this limit up to 750 MB.

Custom CDs vs. Portable MP3 Players

Some people wonder whether it's worthwhile buying a portable MP3 player after they start burning their own audio CDs. Why not just buy a portable CD player and be done with it? After all, most first- and second-generation portable MP3 players won't give you 72 minutes of music encoded at a reasonable bitrate without additional storage cards, and audio CDs don't suffer quality loss due to compression. A portable CD player can play your existing CD collection along with your custom CDs. CDs are universally compatible, and can be played anywhere you find a CD player. Offloading MP3s to CDs will save you tons of disk space. Finally, investing in CD-Rs gives you more flexibility for your money because you can use the same drive as a normal CD-ROM in your computer, and can burn CDs containing any type of computer data, not just audio.

If you're wondering by now why the recording industry isn't more freaked out about recordable CDs than it is about the Rio and similar players, remember that there's a critical difference here: You have to make a physical copy of the disc for every person you want to share your music with, and that's a serious barrier to piracy in comparison to the popular-but-usually-illegal "Upload once, distribute to the entire web population" model. In addition, portable MP3 players are much smaller than portable CD players, and have no moving parts. Portable MP3 players never skip, no matter how hard you exercise. And you don't have to burn a new CD just to get a new playlist!

burner applications will only accept uncompressed audio as input. That means you'll need to find a way to convert your MP3 files to WAV or AIFF before you can make an audio CD. There are many ways to do this, depending on your operating system and the tools you use. Here are a few of the most common methods; see your application's documentation if you use other tools.

Windows: WinAmp

Right-click on WinAmp's interface and select Options → Preferences from the context menu (or just tap Ctrl+P). Navigate to Plugins → Output and select Nullsoft DiskWriter. When you click the Close button, you'll be asked to specify an output directory. Now just play your MP3s normally. You won't hear anything, as all output will be dumped to WAV files rather than sent to the sound card. To decode batches of files, just set up a playlist and let it go to town. Don't forget to return to the Preferences panel and reset the output to WaveOut or DirectSound when you're done!

MacOS

Most MacOS users don't do format conversion directly. Instead, they use CD burning applications like Adaptec's Toast that convert MP3 to raw audio as files are dragged into the burning application itself. Creating audio CDs from MP3s with Toast is no different from creating audio CDs from any other source; the process is elegant and totally transparent.

If you do want to convert from MP3 to AIFF, use the Converter built into SoundJam MP. Open the Converter window, drag in your tracks, and change the MP3 option in the "Convert using..." picklist to AIFF. This may be especially useful if you have an older version of Toast that doesn't convert from MP3 to PCM audio automatically.

Linux: amp

Many command-line MP3 players for Linux include the ability to decode MP3 files to PCM audio; you'll need to check your player's documentation for this ability. Here's an example using the encoder mpg123—a simple shell script that creates a WAV file for each MP3 file in a given directory; you may need to change the initial command and flags on the second line to work with a different tool:

```
for i in *.mp3 ; do
    mpg123 -w $i ${i%.mp3}.wav
```

To burn CDs, most Linux users use cdrecord (*www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html/*), a command-line burner that works with the majority of CD recorders out there, and which has a ton of optional features. cdrecord's documentation is somewhat dense; be sure to look through some of the FAQs and HOW-TOs available on the cdrecord site. For a nice graphical wrapper to cdrecord, download a copy of xcdroast from your favorite Linux software library.

BeOS: SoundPlay

BeOS comes with a drag-and-drop CD burner as a standard component of the operating system—just drag MP3 files from the Tracker into the CDBurner window and click burn—an audio CD will be generated automatically. Third-party burning software is available as well, as is a BeOS port of cdrecord. If you just want to decode MP3 files to WAV without burning them first, drop the files you want to decode onto SoundPlay's interface, then click on the title of the playing file. From the Special menu, select "Decode to file." Tell SoundPlay whether you want to decode to RAW or WAV, single or all files in the current playlist, and specify the destination folder. Click OK, sit back, and you'll end up with a folder full of RAW or WAV files.

CDs vs. MiniDiscs

Some users prefer to store their music on the Sony MiniDisc format rather than recordable audio CDs, for a variety of reasons. First, and most obviously, MiniDiscs are much smaller than CDs, while storing the same amount of music (ten MiniDiscs require less physical volume than four conventional CDs). This also means MiniDiscs are great for joggers and other people who use a portable player heavily. By default, all recording on the MiniDisc is done with a proprietary, non-MP3 Sony format called ATRAC, which compresses audio at a ratio of 5:1.

It's possible to record to MiniDiscs 100% digitally, just by connecting a TOSLINK digital signal converter to your sound card's digital output (actually, this is a technique which should work with any kind of consumer-oriented digital audio device that comes with TOSLINK or S/PDIF output jacks). This means you can just play MP3 tracks through your favorite decoder and dump them straight to MiniDisc. Finally, MiniDiscs are re-writeable, like a CD-RW device. If you're thinking of buying a RAM-based portable MP3 player, you might want to consider a MiniDisc instead—you'll get a lot more playing time and flexibility.

On the other hand, CD-R blanks are quite a bit cheaper than MiniDisc blanks, so you can afford to just throw out the ones you don't like and not worry about it. Perhaps more troubling, you lose compatibility with the zillions of standard CD audio players out there (this problem is somewhat less substantial in Europe and Japan, where MiniDiscs are more popular than they are in the U.S.). Finally, recording to MiniDisc is quite slow—a 74-minute disc will take 74 minutes to record, which is far longer than the speeds you'll get with 2x, 4x, 8x, and even faster CD-R drives.

Internet/Multimedia

O'REILLY®

MP3: The Definitive Guide

MP3 is the most popular compressed file format for easy digital distribution of sound and music over the Internet. *MP3: The Definitive Guide* is the bible of this phenomenon. It's a fast-moving, well-written guide to the juicy bits for MP3 power users. Ripping and encoding are covered in depth, from a true audiophile's perspective. You'll learn exactly what it means to listen to a perceptual codec and how it affects the way you should record, make tests, and listen to your MP3 files. You'll find a detailed examination of the codec itself and its development, as well as how MP3 files compare with other common compression formats.

Like MP3, *The Definitive Guide* doesn't limit itself to one operating system or even one player. It contains advanced information for using MP3 files on all the most popular operating systems, including Windows, Linux, Macintosh, and BeOS. You'll find plenty of information about WinAmp, but players on other platforms are also covered in depth. You'll also find an expert's guide to MP3 files online, web sites to look for, ways to work with newsreaders and ftp, and the latest word on the rapidly evolving technologies of the MP3 audio revolution.

This book also covers:

- MP3 storage, including databases, file systems, and shared playlists. You'll learn about tagged MP3 files and working with global Internet databases to automatically download track and artist listings.
- Security and legal issues, so you can create your own MP3s and stay within the law. Or if you're just interested in the social and legal issues surrounding a file format that could make the record industry irrelevant, you'll learn about the protocol.
- Hardware options. You'll find a buyer's guide to the myriad hardware players including portables and kits. You can listen to MP3s on your PDA, car stereo, or a rack-mounted audio system.

This book will succinctly answer every question, frustration, or problem you might have while creating, using, and listening to music in MP3 format. No matter what your level of involvement with the technology, *MP3: The Definitive Guide* is your blueprint to building a technologically advanced musical library.

ISBN 1-56592-661-7 US $29.95 CAN $43.95

Visit O'Reilly on the Web at *www.oreilly.com*