

# MP3

*The Definitive Guide*



**VANCOUVER ISLAND  
REGIONAL LIBRARY**

**O'REILLY®**

*Scot Hacker*



# MP3

*The Definitive Guide*

Scot HAC

ANF  
780.285  
HAC

Microsoft

**DISCARD**

---

**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

Copyright © 2000 O'Reilly & Associates, Inc. All rights reserved.  
Printed in the United States of America.

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.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps. The association between the image of a hermit crab and MP3 is a trademark of O'Reilly & Associates, Inc.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Library of Congress Cataloging-in-Publication Data

Hacker, Scot

MP3: the definitive guide/Scot Hacker.—1st ed. p. cm.

ISBN 1-56592-661-7 (alk. paper)

1. MP3. 2. MP3 players. 3. Music—Computer programs. 4. Internet (Computer network)—Computer programs. I. Title.

ML74.4.M6 H33 2000

780'.285'65—dc21

00-025403

ISBN: 1-56592-661-7

[6/00]

[M]

---

## Table of Contents

<i>Preface</i> .....	<i>vii</i>
<b>1. <i>The Nuts and Bolts of MP3</i></b> .....	<b>1</b>
MP3 Basics .....	1
Rights, Piracy, and Politics .....	9
<b>2. <i>How MP3 Works: Inside the Codec</i></b> .....	<b>22</b>
A "Perceptual" Codec .....	22
The Anatomy of an MP3 File .....	41
<b>3. <i>Getting and Playing MP3 Files</i></b> .....	<b>47</b>
Choosing and Using an MP3 Player .....	47
Players by Platform .....	52
Obtaining MP3 Files .....	86
Organizing Your Collection .....	94
Equipment Considerations .....	99
<b>4. <i>Playlists, Tags, and Skins: MP3 Options</i></b> .....	<b>104</b>
MP3 Options and Considerations .....	105
Equalization and Sound Quality .....	106
ID3 Tags and Playlists: The Virtual Database .....	116
Skins: Dressing Up MP3 Players .....	133
Plug-ins: Extending Your Reach .....	139
Listening to MP3 Streams .....	148
Performance Considerations .....	153

5. <i>Ripping and Encoding: Creating MP3 Files</i> .....	160
General Encoding Principles .....	160
General Ripping Principles .....	173
Ripping and Encoding Tools .....	175
Ripping from Other Sources .....	193
Roll Your Own Compact Discs .....	201
6. <i>Hardware, Portables, Home Stereos, and Kits</i> .....	209
Playing MP3 Through Your Home Stereo .....	210
Portable Players .....	214
Hand-Held Computers and Other Devices .....	230
Home Stereo MP3 Players .....	233
Car Players .....	241
Kit Players .....	250
7. <i>The Not-So-Fine-Print: Legal Bits and Pieces</i> .....	261
The Rules of Engagement .....	261
The Players .....	268
8. <i>Webcasting and Servers: Internet Distribution</i> .....	284
The Fundamentals of Internet Distribution .....	284
Offering Files for Download .....	285
Webcasting: Real-Time MP3 Broadcasting .....	301
An Interview with MP3.com's "High Geek" .....	322
9. <i>Competing Codecs and Other File Formats</i> .....	329
The Architectures .....	330
The Codecs .....	345
<i>Appendix: ID3v1 Genres</i> .....	363
<i>Glossary</i> .....	365
<i>Index</i> .....	375

---

## *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](mailto:info@oreilly.com)

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

[bookquestions@oreilly.com](mailto: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 Nelsen, 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](http://mp3.lycos.com), [www.listen.com](http://www.listen.com), [www.scour.net](http://www.scour.net), or [www.rioport.com](http://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](http://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](http://www.shoutcast.com)) and icecast ([www.icecast.org](http://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](http://www.mpeg.org). To learn more about Fraunhofer and MP3 licensing issues, see [www.iis.fhg.de](http://www.iis.fhg.de). The official web site of the MPEG Consortium is [drogo.csel.stet.it/mpeg/](http://drogo.csel.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](http://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/](http://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).<sup>\*</sup> 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.

<sup>\*</sup> 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](http://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

*In this chapter:*

- A “Perceptual” Codec
- The Anatomy of an MP3 File

## *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,<sup>†</sup> 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<sup>‡</sup> 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.<sup>§</sup> 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](http://www.xingtech.com)).

<sup>†</sup> 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.

<sup>‡</sup> Aldous Huxley, *The Doors of Perception*. First published by Chatto and Windus Ltd., 1954. Now available in numerous reprints.

<sup>§</sup> 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.

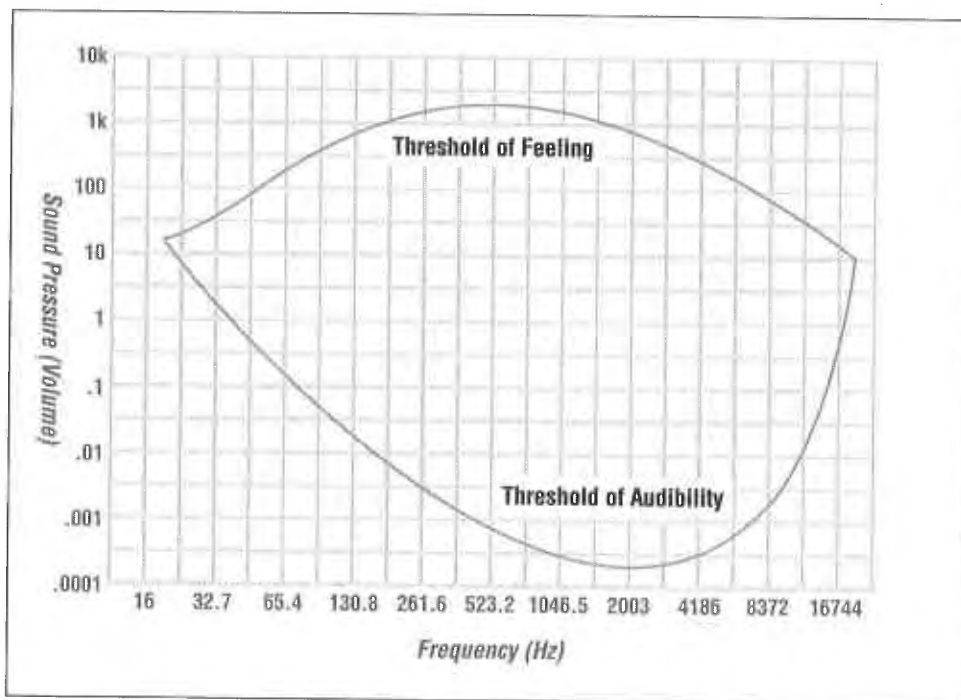


Figure 2-1. While vibratory frequencies extend both above and below, human hearing is pretty much limited to the range between 20Hz and 20kHz

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 "meta-data" 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.

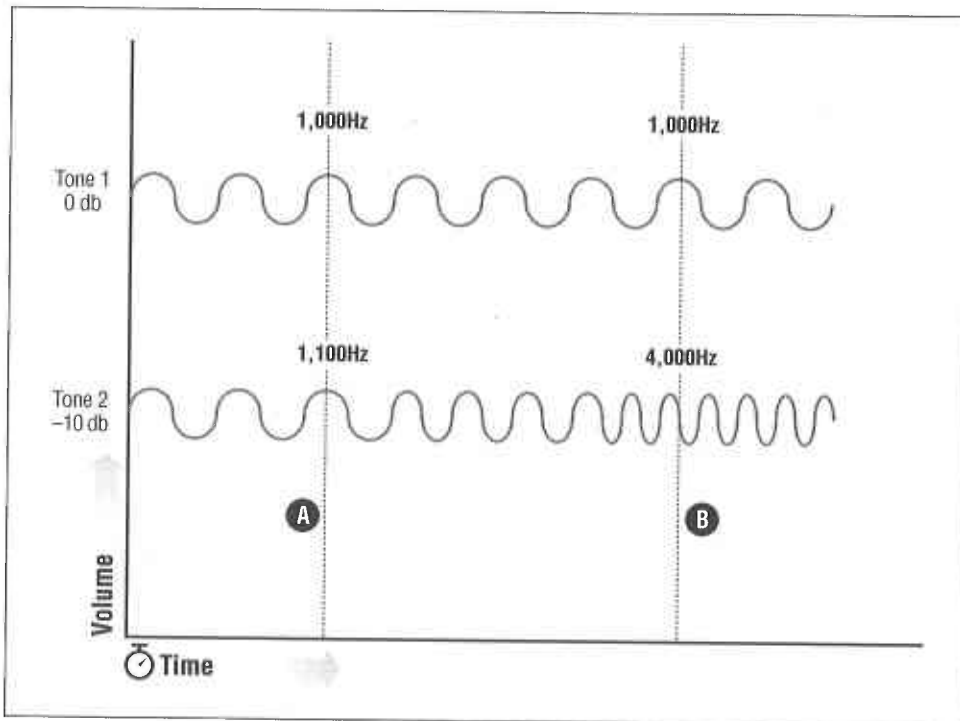


Figure 2-2. As two simultaneous tones become more dissimilar, they become recognizable as separate entities

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, multi-channel 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.

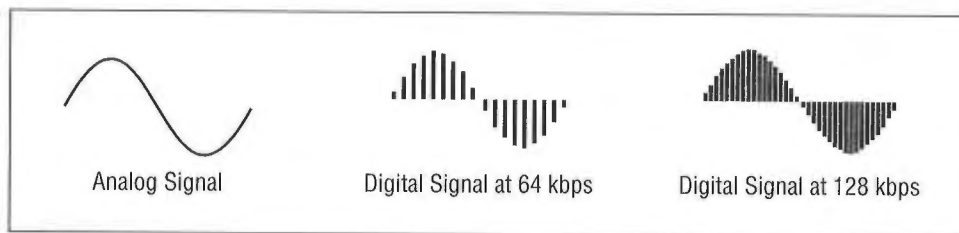


For more information on psychoacoustics, read any of the excellent papers on the subject at [www.cpl.umn.edu/auditory.htm](http://www.cpl.umn.edu/auditory.htm).

### *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 *decoding* is to be accomplished, it only provides sample implementations (one simple and one complex) for *encoding*. 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](http://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 saved 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](http://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 off, 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. The 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.

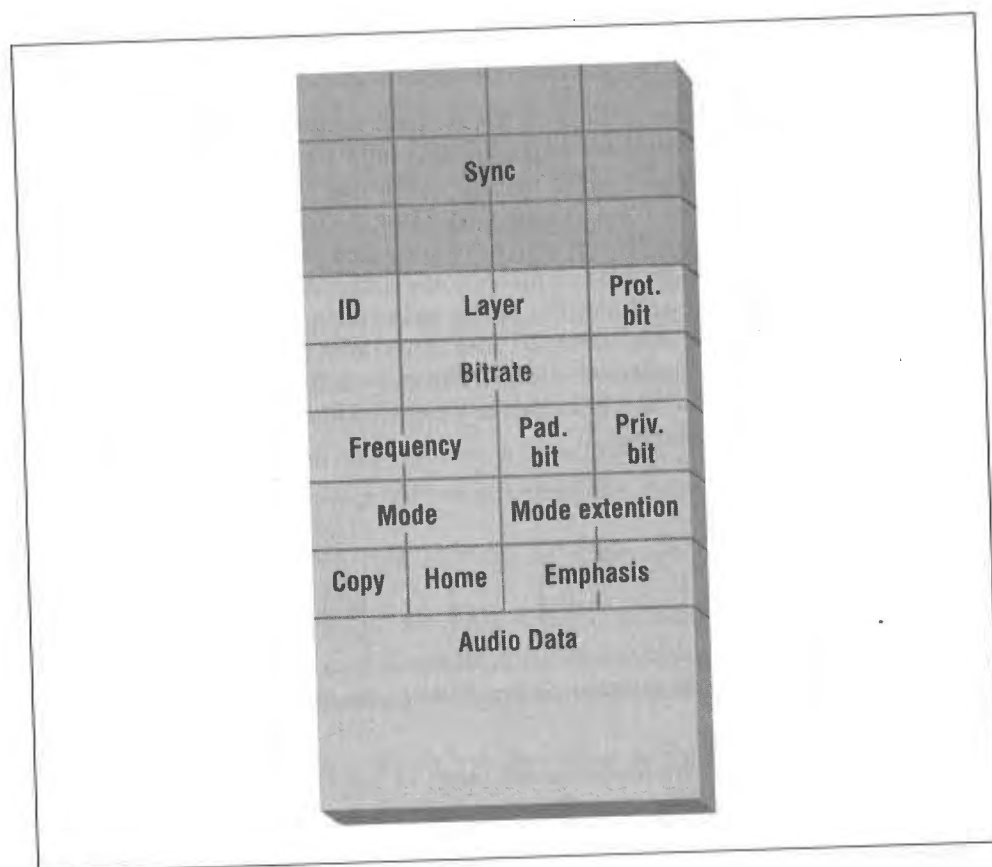


Figure 2-5. The MP3 frame header represented visually



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/](http://www.mp3tech.org/). If you want to go straight to the horse's mouth, start at [www.iso.ch](http://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:  $\text{FrameSize} = 144 * \text{BitRate} / (\text{SampleRate} + \text{Padding})$ .

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 sample-rate 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.

*In this chapter:*

- *Choosing and Using an MP3 Player*
- *Players by Platform*
- *Obtaining MP3 Files*
- *Organizing Your Collection*
- *Equipment Considerations*

# 3

## *Getting and Playing MP3 Files*

At bottom, playing MP3 files is no more difficult than opening any file in any application. In fact, it's usually little more than a simple matter of drag and drop. There are, however, literally hundreds of different MP3 players out there, all of them with specific differences, features, and quirks. There are also a few considerations to keep in mind when building an MP3-optimized computer/sound system, and of course there's the question of how to get MP3 files to begin with, not to mention how to organize them once you've got them. In this chapter, we'll get an overview of the best/most popular MP3 players available for Windows, Mac OS, Linux, and BeOS, and discuss some of their more important options and features. We'll save the more advanced functions and features for Chapter 4, *Playlists, Tags, and Skins: MP3 Options*.

Of course, before you hear the first note, you'll have to get your hands on a starter MP3 file or two. You'll find a basic introduction to MP3 download concepts in Chapter 1, *The Nuts and Bolts of MP3*, and we'll treat the subject in much more detail in "Obtaining MP3 Files," later in this chapter.

### *Choosing and Using an MP3 Player*

At this writing, there are several hundred different MP3 players out there, with new ones appearing practically on a weekly basis. While the lion's share of players exist only for the Microsoft Windows platform, there are great players for Mac OS, Unix/Linux, and BeOS as well (of course there are players for just about every audio-capable operating system in existence, but we cover just these four platforms in this book). MP3 players vary radically from one to the next. Some are more efficient than others, some have groovy psychedelic interfaces, some work only from the command line, some can be extended to do things their authors never dreamed of by way of plug-ins, and so on. You have some choices to make.



While this book gives URLs for most of the tools covered, you'll find plenty more by surfing around the "players," "encoders," or "software" sections of sites such as [www.mp3.com](http://www.mp3.com), [www.emusic.com](http://www.emusic.com), [www.mp3now.com](http://www.mp3now.com), [www.mp3tech.org](http://www.mp3tech.org), and hundreds of other dedicated MP3 web sites.

If you're serious about MP3, there's no reason to stick with the first decoder you download. Try out a bunch of them, run some careful listening and performance tests, and think about exactly what you want to do with MP3 before making your decision. Note that most MP3 decoders play many more audio file formats than just MP3, including WAV, AIFF, AAC, VQF, RealAudio, ASF/WMA, MOD, XM, IT, S3M, and many more. Many players are also equipped to play audio CDs, and most are easy to set up as your system's default sound player if you like. Fortunately, it's not going to cost you an arm and a leg to experiment with multiple MP3 players to find the one that suits your needs, as most of them are low-cost, and some of them are even free. Even the players that aren't free generally let you test them out in some form before making any commitments.



If you're the user of a Unix-based operating system other than Linux, keep in mind throughout this book that most POSIX command-line tools will compile properly on nearly any other POSIX-compliant platform, including the BSD family, Sun's Solaris, QNX, IRIX, and many others. GUI applications may or may not be cross-platform compatible. We don't mean to leave you out here—we're just attempting to address the widest possible audience.

## Basic Features

In the next section, we'll cover MP3 players for various operating systems. Note that we don't even attempt to cover all of the players for any given platform in this chapter—these are simply the best and/or most popular players available as of early 2000. While MP3 player options may increase and change over time, the basic features and functions of the players covered here will apply in some form to most players on most platforms. The decoders chosen for coverage here are represented as examples of how decoders work on their respective platforms, and don't represent endorsements of these players over and above other players that may or may not be available by the time you read this.

### *The Skin I'm In*

As you'll see in this chapter and the next, many MP3 players live on the edge when it comes to user interface concepts. Not only do many of them employ a look and feel completely unrelated to that of the host operating system, but many also take advantage of an increasingly popular concept in UI customization. Rather than locking the user into an appearance programmed in by the developer or dictated by the operating system, more and more modern applications ship naked, like a paper doll you get to dress up yourself. Users are then free to download and apply "skins," which are collections of bitmap images custom-cut to specified dimensions.

Since skins are just collections of bitmapped images, they're completely operating-system agnostic, which means a single skin will work just fine in any operating system, and typically with any MP3 player that supports the WinAmp skin layout. Thus, you may see pictures throughout this book that *look* like WinAmp, but in fact are not—once a player is wearing a skin, it becomes impossible to tell which player it really is or which platform it's running on. Today, there are literally thousands of skins you can download and apply to the currently running player, without even having to stop and restart. Note that skins are nothing but window dressing—they don't affect the quality or capabilities of the underlying decoder in the least, and all developers seem to have found ways to tack additional functionality onto their players without mucking with the skin-imposed user interface (typically via pop-up or context menus).

It's worth noting that the WinAmp skin format isn't the only one out there. Notably, MusicMatch Jukebox, Sonique, K-Jöfol, SoundJam, and FreeAmp all have their own skin templates, which are incompatible with the WinAmp's. Smaller (but growing) collections of downloadable skins are available for these players as well. Take a closer look at creating and using skins in Chapter 4.

If you've ever used a cassette recorder or VCR, you're already familiar with the universal symbols for play, stop, pause, back, and forward, and most graphical MP3 players sport some semblance of these buttons (note, though, that back and forward buttons often skip to the previous or next track in the current playlist, rather than letting you arbitrarily fast-forward or rewind—similar to some less-full-featured home and car stereos). While MP3 decoders can vary radically from one to the next, the same basic functions are typically found in all of them. NullSoft's WinAmp is pictured in Figure 3-1, but the following basic usage instructions should apply to most players out there. Near the play controls, you'll often find what looks like an eject button. However, the "eject" function on MP3 players usually presents a file panel from which you can select a new track. A progress slider typically serves double-duty, letting you "scrub" back and forth to different points in the current file, while also showing you how much of the file has been played.



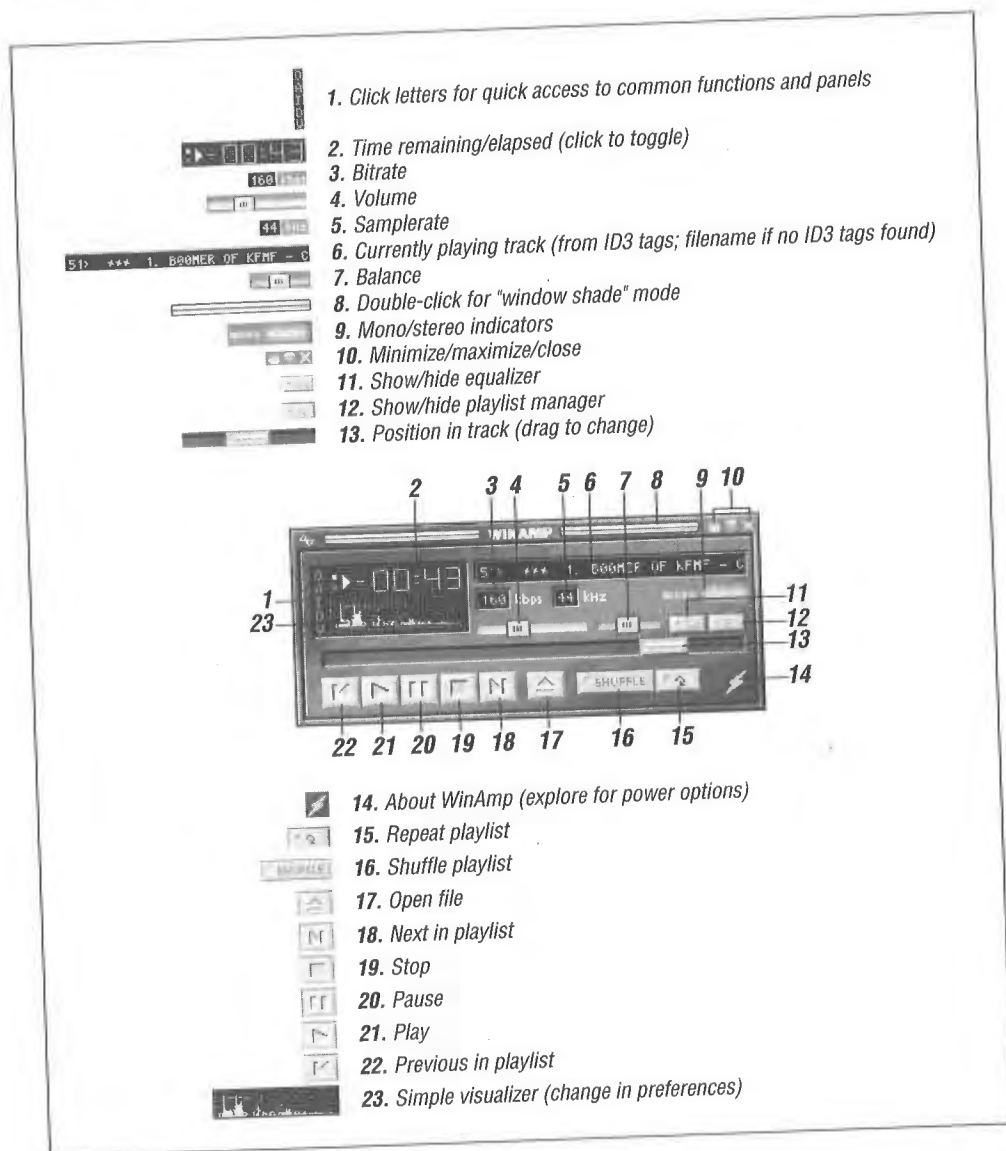


Figure 3-1. The basic functions of an MP3 decoder

Additionally, most MP3 players will include a pair of shuffle and repeat buttons. When the shuffle function is activated, the order of any currently running playlist will be randomized, which can be a gas once you've built up playlists containing many hours of music. The repeat function is often a three-way toggle: repeat current song, repeat entire playlist, and don't repeat.



One of the most enjoyable aspects to working with MP3 players is the fact that they let you create custom “playlists,” which are simply lists of audio files living anywhere on your system and grouped together under a single name. This lets you store your files in logical directory structures as described later in this chapter, but still create listening sessions based on mood, tempo, or other arbitrary criteria. For example, you could create a playlist called “Tranquilizer” that references just the slow songs from dozens of different artists. Behind the scenes, a playlist is just a plain text file naming the full paths to the selected songs. When viewed through the MP3 player’s interface, playlists are often attractive and highly customizable entities that let you rearrange, sort, and skip around through vast collections with ease. Playlists for players that can handle streaming or broadcast MP3 can also store URLs to your favorite broadcast sites (actually, there are two separate types of streaming playlists; we’ll cover those in Chapter 4).

Near the progress slider, you may find a pitch bender of some kind, which lets you speed up or slow down the playrate of the current file. Somewhere on the display, you’ll usually find a digital readout reporting the number of elapsed seconds in the current track, the track’s title or filename (depending on whether ID3 tag data is detected in the file, and often animated or scrolling), the file’s rank order in the current playlist, and its encoding bitrate and sampling frequency. Many MP3 players include graphical frequency spectrum analyzers, which let you visualize the “soundscape” of the current file as it plays. Many players also offer a full graphic equalizer, which lets you control the relative emphasis of various subbands across the frequency spectrum (see Chapter 4). The equalizer may be cleverly concealed in a roll-out drawer or other hidden window; to bring it forward, look for a button labeled “EQ” somewhere in the interface. Similarly, the player’s playlist function will likely be initially hidden, but can be brought forward by clicking a button marked “PL” or with an icon resembling a list.



Some of the skins available out there are *very* bizarre, and may disguise these elements behind cryptic or incomprehensible interface elements. However, note that the basic skin layout is usually fixed, so the position of elements relative to one another should remain constant for MP3 decoders that wear WinAmp skins. Thus, it’s generally easy to find hidden controls once you’ve become familiar with the generic WinAmp layout.

Some MP3 decoder players can be resized into two or more formats, which lets you hide your player without minimizing it completely; many controls are still visible and usable. This feature, known as *window shade* mode, is generally accessed

by double-clicking the player's title bar (Mac OS users will find this behavior familiar), as shown in Figure 3-2. Minimize, window shade, and close buttons will typically appear somewhere on or near the title tab as well, depending on the player. To access additional functions not obvious in the default interface, try right-clicking any unused space (Mac users: Ctrl+click or Opt+click) to access a context menu.



Figure 3-2. WinAmp running in "window shade" mode

Note that many (if not most) MP3 players aren't limited to playing MP3, but can handle a wide variety of additional audio file formats as well. Once you've settled on a player, you may want to dig through its options for a mechanism that will let you easily establish it as the default handler (favored application) for any or all audio file formats you might encounter, if your operating system permits this.

## Players by Platform

With the basics out of the way, we'll take a look at some of the more idiosyncratic or less intuitive features of the most popular players available for various operating systems. The players are listed by operating system, and alphabetically within each operating system section.

### Windows 95/98/NT

Needless to say, there are more applications written for Windows than for any platform on the planet, and that includes MP3 encoders and players. Search your favorite MP3 software archive and you'll find a bottomless pit of options. And because of the sheer critical mass behind Windows, its MP3 apps are also without question the most evolved and mature available.

Unfortunately, Windows users may also expect a few hitches along the way: Glitches in the sound stream, hung applications, and the dreaded blue screen of death are all part of the Windows experience that many of us have come to simultaneously respect and hate. In all fairness, these behaviors may be the exception and not the rule, and most Windows MP3 applications are very well behaved most of the time. If all you want to do is run one audio stream at a time and you have a relatively stable Windows installation, you don't have anything to worry about, and will likely enjoy trouble-free audio. However, you may also experience a perceptible impact on overall system responsiveness when working in other applications while MP3 players do their thing in the background, depending on your

available horsepower and the configuration of your hardware. The primary reason for this behavior is that threading in Windows is rather coarse-grained, meaning that when multiple applications are trying to do work simultaneously, they take long turns with the processor in comparison to, say, Linux or BeOS.

You may also experience specific limitations in the architecture of Windows' audio subsystem. For example, MP3 players like WinAmp and Sonique allow you to specify "Allow multiple instances" in their preferences. Unless you have Microsoft's DirectSound installed, however, trying to open any two audio applications at once in Windows with a SoundBlaster card will result in a "Could not open audio device" message. In other words, Microsoft has added the capacity for handling multiple simultaneous audio streams to the system late in the game, and many millions of Windows users are not able to do this by default. For general-purpose MP3 playback, though, issues like this one are of little concern to the vast majority of users. Meanwhile, many Windows MP3 players are very efficient, with feature sets unmatched on any platform.

### *MusicMatch Jukebox*

*www.musicmatch.com* While in most cases MP3 players and encoders are different beasts, there are a few products out there that combine CD ripping, encoding, playing, playlist management, and the rest of the kitchen sink all in a single, compact interface. MusicMatch Jukebox is a great example of a well-designed (and very fast) all-in-one package, as shown in Figure 3-3.

In addition to a standard playlist feature, MusicMatch offers a "Music Library," which optionally catalogs every supported audio file on your machine, essentially providing a database of available audio from which you can build your custom collections as playlists. As with Windows Explorer, you can sort and organize your music library by clicking the column headers, or resize columns by dragging the header separators.

Since MusicMatch includes excellent documentation of its own, there's no point in rehashing the basics here. In a nutshell, use the Music Library to database the music on your hard drive, then drag files out of the Music Library and onto the player or into the playlist window. Alternatively, drag files directly from Explorer and into the player or playlist window.

MusicMatch includes partial support for ID3v2 tagging (see Chapter 4), so you can store and display additional data related to a given title such as a bitmap image of the album cover, song lyrics, and comments from the artist. If a particular track doesn't already have this information embedded, you can add it by activating the Track Info window, then clicking the Tag microbutton. Enter or change any information related to the track from the Tag panel, then click the Tag Art button, navigate to any 100×100 .BMP or .JPG image on your system and click Done. From

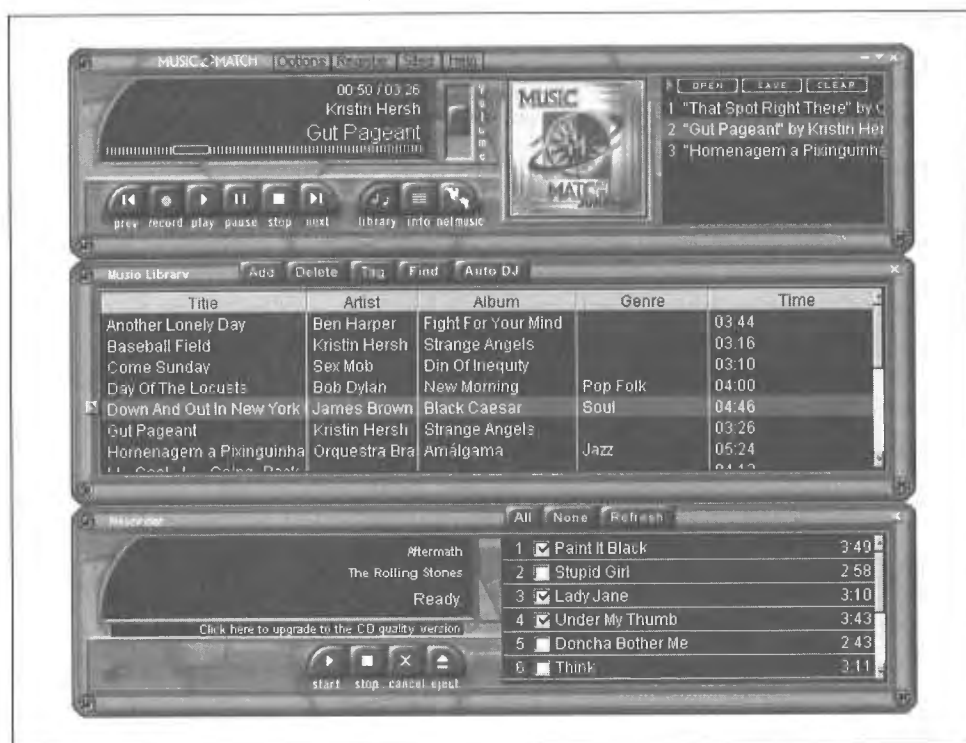


Figure 3-3. MusicMatch offers all-in-one ripping, encoding, and MP3 playback environment in a handsome interface

now on, the image associated with that track will appear in place of the usual MusicMatch logo whenever you're playing that track.

If you find that you like MusicMatch's database functionality but would prefer to use WinAmp as your default MP3 player, pull down Options → Player Controls → Use WinAmp player. While you're in there, dig through the Options menu thoroughly—you'll find tons of configuration possibilities that may not be apparent at first glance.



Interestingly, MusicMatch Jukebox includes a "Line-in" encoding option, which lets you encode MP3s directly from analog sources such as LP or tape. We'll cover this functionality in depth in Chapter 5, *Ripping and Encoding: Creating MP3 Files*.

### *Real's Real Jukebox*

*www.real.com* Perhaps the most consumer-friendly combination player/ripper/encoder for Windows users is Real's Real Jukebox, which includes some nifty database-like functions to help users organize and sift through their ever-growing collections.

#### *Installing Real Jukebox*

Most Real Jukebox configuration options can be established during installation. Pay particular attention to its "security" feature, which is enabled by default. Leaving this option enabled will mark your files with ownership data, making it difficult to play them with other hardware or software MP3 players. You may want to use the Change button on the final installation setup screen to disable security checking.



By turning security off, you are essentially taking it upon yourself to follow your own judgment and use only legal MP3s. By leaving security on, JukeBox can help prevent you from accidentally playing illegal MP3s. This is a nice option for those who choose to respect the wishes of the RIAA and steer clear of pirated music.

#### *Playing music with Real Jukebox*

Before you can play anything with Real Jukebox, you'll need to get some MP3 or other audio files into its database, called the Music Library. If you've already downloaded or encoded a collection of files using other software and just want to start playing your tracks without importing them permanently, pull down File → Open Files and Playlists. If you want to have your existing collection imported into the Music Library so you can further organize, sort and group your tracks later on, pull down File → Import Files and Playlists. The files or playlists you select will begin to play automatically, using JukeBox's built-in decoding engine. Skip to another track by double-clicking its name in the list of tracks. Pause, skip backward/forward, stop, shuffle, and repeat buttons are located in a row near the top of the JukeBox window.

Because the Music Library includes the ability to sift and sort through your collection, you'll find an Explorer-like hierarchy to one side of the main window, with expandable icons for Artist, Album, and Genre. Expand one of these trees and your Music Library will be instantly parsed according to those criteria. JukeBox also makes it very easy to create custom "playlists." Click the New Playlist button at the bottom of the screen and a blank window will appear. Now click Add Tracks and a music navigation window will let you select tracks from any hierarchy in your Music Library. Select a track (or use

Ctrl-click to select multiple tracks) and click the Add button to include that track in your new playlist. Pretty slick.

#### *Ripping/encoding with Real Jukebox*

By default, JukeBox assumes that you'll want to encode all of your files to RealAudio format, rather than MP3. Since, if you're reading this book, you probably want to build an MP3 collection, you'll also want to make this change in the preferences.

Real Jukebox is constructed as a tightly integrated system, designed to be as simple to use as possible. While it does indeed take care of some of the nasty details on your behalf, its simplicity may make it unsatisfactory for some power users, since it offers less control over sophisticated encoding parameters. Before you begin, you'll want to make sure you've set the encoding bitrate to your liking (note that the free version is limited to 96 kbps encodings; you'll need to pay to get the full range of encoding options). If you accepted the defaults during installation, pull down Options → Preferences → Encoding Options and select a quality level (allowable bitrates are listed at the right, though they're not clearly labeled as such). Real Jukebox is also configured by default to start recording audio CDs automatically upon insertion. Blech. Disable this option from the General tab of the Preferences panel.



Although frustrating for sophisticated users, Real Jukebox insists on placing all encoded tracks into a single directory, then keeping track of them with its own internal databases and playlists. Note that Real Jukebox playlists are in a completely proprietary format, and are not hand-editable like normal playlists. You'll find them stored in the *db* subdirectory of the Real Jukebox installation directory. Unlike Music-Match Jukebox, you can't tell Real Jukebox to create a new directory for every new artist, with subdirectories for each album. If you're willing to let Real Jukebox manage all of your music, this won't present a problem since its internal database does a great job of cataloging albums, artists, and genres for you. If, on the other hand, you want to keep working with a pre-existing system of carefully constructed artist/album directory structures, Real Jukebox won't let you. If you move your MP3 files around later, or manually organize them into subdirectories, Real Jukebox will complain the next time it encounters that file in one of its databases or playlists. You'll have the option to tell it where to look for the file instead, but this process quickly grows tiresome. In other words, if you want to use the Real Jukebox system, you have to use all of it—warts and all.

After inserting a CD and launching Real Jukebox, select the record/play CD icon in Real Jukebox's lefthand navigation panel. A connection will be



established with the CDDB,\* and the Album and Artist names will appear along with the Genre, with all track names and lengths listed in the main screen. If you want to record all tracks, just click the red Record button on the main toolbar and sit back. If you want to record only a few tracks, use the empty checkbox at the bottom of the screen to deselect all tracks, then manually select the tracks you want to record. If you'd like to start downloading a collection of MP3 files, the Real Jukebox interface includes built-in links to sites where you can find legitimate (non-pirated) MP3 files.

### *Sonique*

*www.sonique.com* One of the most eagerly awaited MP3 players to be unleashed on the scene in a long time is Night 55's Sonique, one of the first new contenders to give long-reigning champ WinAmp a serious run for its money. In fact, within months of its initial release, Sonique had become the second most popular MP3 player in use under Windows, and its popularity continues to grow at this writing. So what's so great about Sonique? Well, two things: its incredibly trippy interface and its very high-quality decoder. Night 55 claims that Stardust, its MP3 decoder engine, is the best-sounding player available, and our subjective tests confirmed this claim. The audio quality of the Stardust decoder is particularly noticeable when listening to tracks encoded with lower bitrates, which sound less "swishy" than do the same tracks played through WinAmp. Night 55 attributes this quality difference to a form of audio antialiasing run built into Stardust, along with a close code similarity to the original Fraunhofer decoder.

Sonique also represents a fascinating experiment in discarding user interface conventions—Night 55 has constructed a look and feel from scratch, completely unrelated to the underlying operating system. Gone are all notions of title tabs, scrollbars, zoom boxes and the like, and in their place are organic, rounded surfaces, animated interactions, flowing menus, and unusual navigational tools, as shown in Figure 3-4. Of course, discarding all UI conventions has always been a controversial move (witness the battles that raged surrounding the initial release of Kai's Power Tools in the mid-90s). The problem with this sort of interface is that it forces the user to learn everything anew, and indeed, even experienced users will have to experiment for a while before they discover just how they're supposed to operate an application like this.

The pay-off is an aesthetic experience not often duplicated (though the MP3 world certainly does have more than its fair share of UI mad scientists). There's probably no better way to familiarize yourself with Sonique's UI than to play with it, but here's a brief guide to get you started.

---

\* The Compact Disc Database, an Internet database storing track numbers, titles, and lengths for hundreds of thousands of popular compact discs. More details on the CDDB can be found in Chapter 5.





Figure 3-4. Sonique's animated interface



Users interested in unique interface concepts may also enjoy checking out K-Jöfol, a freeware multimedia player for 95/98/NT that is among the fastest Win32 players available. See [www.kjofol.org](http://www.kjofol.org) for more information.

Sonique sports three main window styles, or "mode states": "small-state," "mid-state," and "enlarged" or "nav-state." Each mode gives you access to basic player controls, but differ in form and in the number of additional goodies to which you have access.

There are two ways to move amongst the three modes:

- Double-click in any blank space in the application's frame.
- Use the set of four small, round icons containing arrows. The up arrow takes you to nav-state if you're in mid-state, and to mid-state if you're in small-state, while the down arrow does the opposite. Double up or double down arrows will jump you across two levels at once.

The following descriptions will allow you to choose the best mode to suit your needs:

*Enlarged or nav-state*

This mode lets you view the finest level of detail available on the currently playing file, including timing information down to the one-hundredth of a second and a bps (bits-per-second) readout that updates itself in real time when VBR files are being played. Perhaps most importantly, however, the enlarged mode lets you view the current visualization method at the largest possible size (a full-screen mode is also available). When in nav-state, the main screen is by default occupied by a collection of icons representing online tools, the playlist editor, size mode, setup options, info, and audio controls. Once you access any of these, a sideways arrow will appear amongst the navigational icons, which will return you to full mode's main menu. So you can think of nav-state as harboring a collection of additional horizontal options, and of the two other states as merely different aesthetic form factors. A few of the options available from nav-state bear further explanation. Since navigating amongst Sonique's many modes can be a little... odd, a wide range of navigation and playback hotkeys are provided; see Table 3-1.

The other two states are simple players, and should require no explanation. Select these if you want Sonique to consume less of your desktop real estate.

*Setup Options*

Perhaps the most important module available from the full mode view is the Setup Options panel, from which you can establish many aspects of Sonique's eye-candy settings such as smoke, blur, zoom, and effects levels. In addition, you'll find five "tabs" available along the right side of this mode. The general settings let you configure aspects of Sonique's windowing and playlist behavior, as well as letting you establish Sonique as your system's default audio CD player. If your machine has multiple sound cards installed, you can select which device Sonique should send its output to from the Audio tab.

The Visual tab lets you control the speed and smoothness of Sonique's window and menu animations (note that this control is separate from the control for audio visualizations, described previously). If Sonique's animations start to make you a little queasy, you can easily turn them off by dragging the Animation Speed slider to zero. This can also be useful if you have a slower machine and are finding that the sound skips or gets "crunchy" on occasion. Since the animation effects are somewhat CPU-intensive, turning them off can save you a few precious cycles (note that Sonique only runs on Pentium-class computers and higher). The File Types tab lets you specify which audio file formats are registered in Windows as being handled through Sonique by default. The Audio Controls tab brings Sonique's graphical equalizer forward, so you can fine-tune the frequency distribution across 20 sub-bands, as well as manipulating the volume (amplitude), balance, and pitch of the currently playing track. More details on equalization are in Chapter 4.



If you spend a lot of time at the command-line, note that Sonique can be launched and controlled from a DOS prompt, like so:

`Sonique -play`

This will cause Sonique to start playing the last current song when Sonique was last closed. Other allowable flags are:

```
appendonly <filename> (adds filename to current playlist)
clearplaylist
loop
play
stop
resume
removecursong (removes current song from playlist)
pause
seeknext
seekprev
seekto[track#] (jumps to an arbitrary numbered track in the
playlist)
shuffle
```

Of course, the path to Sonique's installation directory must be in your system path variable, or you must run these commands from within Sonique's directory itself.

Table 3-1. Sonique Hotkeys

Hotkey	Function
B	Next song
C	Pause
D	Delete current song
L	Load song
P	Playlist editor
Q	Equalizer toggle
V	Stop
X	Play/stop
Z	Previous song
Ctrl+D	Clear playlist
Ctrl+E	Enlarged mode
Ctrl+I	Announce to IRC toggle
Ctrl+M	Mid-state mode
Ctrl+N	Nav console
Ctrl+O	Online tools
Ctrl+P	Playlist
Ctrl+R	Setup options

Table 3-1. Sonique Hotkeys (continued)

Hotkey	Function
Ctrl+S	Play/stop
Ctrl+T	Always on top toggle
Ctrl+,	Small-state mode
Ctrl+Tab	Change state
1-9	Volume settings
^(~)	Mute volume
0	Full volume
Ctrl+double-click frame in enlarged mode	"Dexter" mode (try it!)

### WinAmp

[www.winamp.com/](http://www.winamp.com/) Without question, NullSoft's WinAmp (Figure 3-5) is the most popular MP3 player available for the Windows platform (and thus, by definition, for any platform). While other very capable players are available, and debate continues as to whether the free WinAmp offers sound quality as good as some of its competitors, WinAmp got a big leg up in the early popularity game, and had hundreds of thousands—if not millions—of installed units before other GUI MP3 decoders even started to make headway. There's good cause for WinAmp's popularity, too—when it comes to the number of features, options, and supported goodies, nothing even comes close. Later versions of WinAmp even come with a built-in "mini-browser," which will let you search for MP3 music at Amazon.com by name or genre.

Because of its wide support for skins and plug-ins, WinAmp has spawned an entire cottage industry of innovators creating and contributing additions and extensions to the base program. Nor is WinAmp limited to playing only MP3 files—between its native capabilities and its many available plug-ins (most of which are freeware), WinAmp can play most any audio format under the sun, and can be set up as the default player for audio files, overriding Windows' built-in audio player.

Amazingly, for a program with as many possibilities as are offered by WinAmp, the program (at this writing) ships with almost no documentation whatsoever. WinAmp's interface, for the most part, is intuitive enough that most options can be figured out just by fiddling around and paying attention. In addition, the WinAmp web site hosts an array of overview information and FAQs. The Playback Basics earlier in this chapter should be all you need to get up and running with 90% of the things you'll want to do with WinAmp, so we'll just take a look at a few of the less obvious features here.



In early 1999, NullSoft was sued for \$2 million by PlayMedia Systems, inventors of the original Amp engine around which the original version of WinAmp was built. Modern versions of WinAmp use the "Nitrane" decoder, which NullSoft claimed to have developed on their own. The controversy boiled down to a question of whether Nitrane contained enough intellectual property left over from the Amp days to justify an IP case on PlayMedia's behalf. The suit was later settled out of court... just before NullSoft was purchased by America Online, along with Spinner.com. Shortly after the purchase agreement, WinAmp became a 100% free player.



Figure 3-5. NullSoft's WinAmp wearing a third-party skin and displaying its graphic equalizer, a playlist, and a visualization plug-in

### Setup

After installing WinAmp, you'll be presented with the WinAmp Setup → Settings panel, which will let you configure the way WinAmp is configured on your system. If you accept all of the default options in this panel, you'll be establishing WinAmp as the default sound player for your system, both for audio file formats and for audio CDs. If you'd like to experiment with the program before committing yourself, you may want to deselect these options. You'll also be able to specify whether a WinAmp icon should be added to the Start menu, the Desktop, and the Quick Launch bar (Win98/IE4 only). You can always access the Preferences panel again later by tapping Ctrl+P.

### *WinAmp options*

Beyond the obvious interface elements that can be accessed by clicking buttons and dragging files onto the player or into a playlist, all additional functions and features can be accessed in one of three ways:

- Right-click any blank space in the interface itself.
- Left-click the “control” button at the top left of the WinAmp window.
- Tap Alt+F.

Any of these three actions will activate WinAmp’s main context menu, from which all additional features can be accessed. In addition, WinAmp is highly keyboardable, and most features can be activated via hotkeys. To access a complete list of WinAmp hotkeys, click the lightning bolt in the lower-right corner of the interface and select the “Keyboard Info” tab.

Only non-obvious entries in the WinAmp context menu are covered here:

#### *Play Location*

WinAmp is capable of playing MP3 files being streamed or broadcast over the Internet (Chapter 8, *Webcasting and Servers: Internet Distribution*). To open an MP3 URL, just tap Ctrl+L and enter or paste in the URL (typically including a port address), and hit Enter.

#### *View File Info*

As discussed in Chapter 2, *How MP3 Works: Inside the Codec*, MP3 files may include additional “meta-data” that store information such as the track’s artist, the song name, the album from which the track came, the year of publication, and more. This information may be stored in ID3v1 or ID3v2 format, though WinAmp supports ID3v1 files only at this writing (ID3v2 tags will not stop WinAmp from playing—it just won’t see the tags). Access this menu item (or tap Alt+3) and a panel will appear displaying all available ID3 tags. Edit this information if you like, click OK, and your edits will be saved into the file permanently.

#### *Options → Preferences*

Accessing this item (or tapping Ctrl+P) launches a very detailed, multi-tabbed dialog from which you can establish dozens of preferred behaviors for WinAmp, or override its default settings. These options are covered in detail in Chapter 4.

#### *Options → Skin Browser*

Accessing this item (or tapping Alt+S) launches a chooser from which you can select any of your installed WinAmp skins. By default, WinAmp looks in a subdirectory of its own folder called “skins” for skins folders. However, there are two drawbacks to sticking with this default location. First, if you ever decide to uninstall WinAmp and delete the WinAmp directory,

you'll wipe out your entire skins collection. Second, since other players and other operating systems can all share a single collection of skins, it makes sense to keep them in a separate, sharable location (such as `C:\DATA\SKINS`). You can change the skins directory by clicking the Change Directory button on this panel.



Tapping Alt+S will bring up the Skin Browser quickly.

---

*Options → No Playlist Advance*

When this item is selected and you're playing files via the WinAmp playlist, the next file in the playlist will not be played automatically when the current one is finished.

*Options → Always on Top*

This option affects the way WinAmp interacts with other applications. When selected, WinAmp will stay in the foreground even when another application has focus. In other words, it will let you type in your word-processing application even though the WinAmp window is front-most.

*Playback → Previous/Play/Pause/Stop/Next*

These entries are essentially just another way of accessing the normal functions of the player controls and the playlist. Note that each entry has an associated hotkey, so you can navigate and control your currently playing list without ever reaching for the mouse (knowing that you can quickly tap the V key whenever your phone rings is especially handy). The next six items in the list let you navigate with even more precise control. For example, you can jump forward or backward five seconds in the current file, which can be handy when trying to catch an obscure lyric, for example.



The last two items in the Playback menu let you jump to any arbitrary point in time or to any arbitrary file in your playlist. For example, if you want to jump to a point exactly 3:14 into the file, tap Ctrl+J, enter 3:14, and hit Enter.

---

*Visualization → Mode*

This item lets you control the graphical display representing the signal's visualization in the small box just below the time readout. Analyzer gives you a sub-band "real-time analyzer" or "equalizer"-style display, with

peaks being represented at each of 20 frequency bands. Toggling to Scope mode will cause this display to appear similar to that of an oscilloscope, with a single horizontal line waving and peaking in sync with the signal.

#### *Visualization → Misc*

The rest of the Visualization sub-menu is occupied by dozens of options for customizing the precise behavior of the visualization, whether it be in Analyzer or Scope mode. Note also that additional options are given for customizing the behavior of the visualization when WinAmp is in WindowShade mode. If you find the visualizations distracting (because of their constant flickering and dancing), you can also turn them off completely. The Refresh Rate lets you specify how frequently the display should be updated. If you've got a lot of CPU clock cycles to spare, you may as well set this high to get the most accurate display possible. The bottom portion of this menu lets you enable, disable, or configure various WinAmp plug-ins. Plug-ins are covered in detail in Chapter 4.



Click the region in the lower right of WinAmp's interface (usually demarcated by an icon but possibly blank, depending on the current skin) to access a tabbed dialog containing movie, credits, shareware registration information, WinAmp tips, and an excellent list of links to key MP3-related web sites. If your list of links appears blank or out-of-date, click the Update Links button to download an updated list from NullSoft. Really, despite all of the excellent MP3 "links" sites out there, this hierarchical list is one of the best and most complete you'll find anywhere—the cream of the crop.

#### *Windows Media Player*

[www.microsoft.com/windows/mediaplayer/](http://www.microsoft.com/windows/mediaplayer/) While it's nothing to write home about in terms of features and appearance, your version of Windows may come bundled with a perfectly capable MP3 player in the form of the Microsoft Media Player, which gained MPEG support after the release of Windows 98, as shown in Figure 3-6. You'll find the Media Player in the Accessories → Entertainment program group. If Media Player gives you an error message when an MP3 file is dragged onto its interface, then you've got an older version. To update, go to the Microsoft URL listed earlier and download an update. Its controls are straightforward and easy to use.

Significantly, upgrading to a newer version of Media Player may install DirectSound3D onto your system, thus enabling supporting applications to share the sound stream.



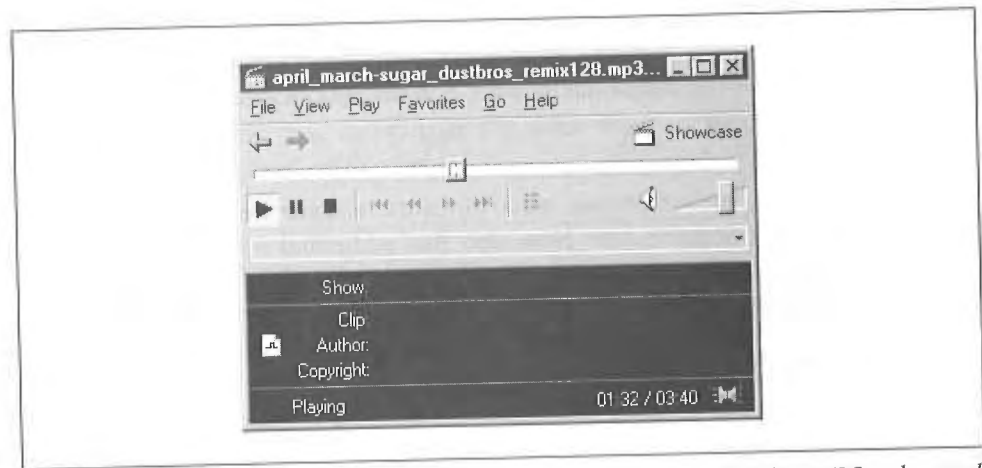


Figure 3-6. Later versions of Windows Media Player include support for the MPEG codec, and thus can play MP3 files

There is, however, one big gotcha to look out for if you use a recent version of Media Player and do most of your web surfing with Internet Explorer. Upon installation, Media Player will set itself up as the default MP3 handler without asking your permission. While you can easily take the association back through the preferences panel of another MP3 application, you'll still find that when you click a link to an MP3 file from within Internet Explorer, the file will be automatically streamed to Media Player, which means you'll get to listen to it immediately but won't be able to save the file for future use... or at least that's what Microsoft would have you think.\* In reality, MP3 files streamed through MediaPlayer are indeed saved to your hard drive, though MediaPlayer gives you no indication as to where, and does not present any kind of download location file panel. The secret they apparently don't want you to know is that the downloaded file is kept in your cache. After listening to a file through MediaPlayer, open Windows Explorer and navigate to `C:\WINDOWS\LOCAL SETTINGS\TEMPORARY INTERNET FILES\CONTENT.IE5` (that is, if you're using Windows 98 or IE5—the location for other versions of Windows/IE should be similar). You'll see thousands of small files representing all the HTML pages you've browsed with Internet Explorer that are still in cache. Sort this directory by Type and you'll be able to find all of the MP3 files in your cache. You can then move them to a more logical, and permanent, location.

A short-term work-around for this situation is to *right-click* MP3 links in Internet Explorer and choose "Save Target As..." from the context menu. The author has not been able to locate instructions for a permanent fix to this problem. Either you do all of your MP3 downloading with another browser, remember to use the right-click trick, or be ready to go digging through your cache for the goods.

\* Downloads started through Netscape Navigator are not affected by this peculiar behavior.

### *Configuring the Default Handler for MP3 Files in Windows*

All associations between filetypes and preferred applications are handled in Windows through Explorer's File Types panel:

Windows 95/NT4: In Explorer, pull down View → Options → File Types

Windows 98/IE4: In Explorer, pull down View → Folder Options → File Types

While one might expect to be able to locate a filetype directly, the arrangement of this panel instead sorts types by their current associations, which means you may find the MP3 entry under "MP3 File" if no player is currently associated with files ending in *.mp3*, or with the name of the current MP3 player, if you've set up associations through another program. In other words, the filetype you're looking for may be called "WinAmp Media File," "Sonique File," or something else entirely; thus, you can't necessarily just navigate through this panel for a definitive filetype. Instead, you need to know the name of the current association before you begin. If you don't already know the name of the filetype Windows is currently giving to your MP3 files, navigate through Explorer to an MP3 file, make sure Explorer is in Detail view, and find the name under the "Type" column. With any luck, a future version of Windows will correct this backwards state of affairs.

Once you've located the correct filetype, double-click its entry and you'll be facing a panel from which you can change that type's icon, descriptive name, MIME type, default extension, and associated actions. The actions that concern us here are "open" and "play." Double-click "open," click Browse, and navigate to the location of the MP3 player of your choice. Click OK, then re-open the "open" option, where you'll see the path to the executable surrounded in quotes. After the end of the quotes, add a space and the string "%1" (including the quotes). Thus, a correct entry in this field might read:

```
"C:\Program Files\MediaScience\Sonique\Sonique.exe" "%1"
```

if you want Sonique to be your new MP3 player. %1 indicates that the program is able to accept filenames as arguments passed to it from the Windows shell. Click OK and then OK again to make your changes stick. While all of this is the "official" way to establish filetypes in Windows, most mature MP3 players include some kind of mechanism for establishing themselves as the preferred handler programmatically. In most cases, life will be easier if you simply dig around in your favorite player's options or preferences and let the app set up associations for you.

Remember that if you use Windows Media Player and Internet Explorer, you'll have an unbreakable association between the two. See the MediaPlayer section previously for workarounds.



Media Player, naturally, is also capable of playing Microsoft's challenger to the MP3 format, WMA/ASF (part of the Windows Media package). The ASF format is covered in Chapter 9, *Competing Codecs and Other File Formats*.

## Linux

Linux is a tremendous server-oriented operating system working very hard to become an excellent desktop/workstation OS as well. Linux may not be an ideal platform for audio, but it's stable, flexible, and the hacker culture that embraces it means there's no shortage of MP3 options for those willing to roll up their sleeves and get their hands dirty. In addition, there's a natural affinity between the open source culture of the Linux community and the prevailing winds of "share and share alike" in the MP3 world.

Fortunately, the simplest of Linux players are a piece of cake to set up and start playing with. Unfortunately, if you want to get the really cool Linux players happening on your machine, you may have to do a bit of tweaking first (but hey, that's why you use Linux, right?). Due to its command-line underpinnings, there are many very effective players available that work from your favorite shell. In fact, these players tend to be the most efficient, so if you've got a slower machine or just aren't an X fan, you're all set. Many (but by no means all) of the GUI players for Linux function simply as graphical front-ends to command-line players such as mpg123. Thus, if you plan to experiment with a variety of Linux players (and you should), it's highly recommended that you get at least one command-line player running first.

### mpg123

[www-ti.informatik.uni-tuebingen.de/~hippm/mpg123.html](http://www-ti.informatik.uni-tuebingen.de/~hippm/mpg123.html) One of the most basic, and yet most flexible, command-line MP3 players for Linux is the free mpg123. Installation is as simple as the usual routine:

```
./configure
make
make install
```

after which you should be able to type `mpg123 filename(s).mp3` from any prompt. Make sure that mpg123 is in your path, especially if you plan to use it as a back-end to a graphical frontend later. mpg123 can take a ton of command-line options; for a quick listing of possible arguments, type `mpg123 --help`. For complete usage details, type `man mpg123`. Here's a rundown on a few of the most useful/common arguments.

To have mpg123 report back with additional details as your MP3s are being played, use the `-v` switch (for verbose), as in:

```
mpg123 -v file.mp3
```

This will let you know the bitrate, frequency, channel status (mono or stereo) and, interestingly, an ongoing frame count as the file progresses. Of course, it's not often that you want to play just a single file. More likely, you've got hundreds or thousands of files, which may or may not be referenced from a playlist (see Chapter 4 for more on playlists). If you've already got a playlist, feed it to mpg123 with the `-@` flag:

```
mpg123 -v -@ /path/to/playlist.pls
```

The same switch with a single trailing hyphen can be used to feed the decoder a list of files from standard input. Combined with your friendly neighborhood `find` command, you can play everything in a given directory structure with a simple command like:

```
find ~/shacker/mp3/ -name \*.mp3 | mpg123 -v -@ -
```

The value of using the `find` command over `ls` here is the fact that it will find everything in all subdirectories of the named parent directory automatically, whereas `ls` will list only the files in the named or current directory. Playing MP3 bitstreams over the Internet is covered in Chapter 4.



If you have an older machine and CPU cycles are at a premium, you can force mpg123 to use a different frequency ratio with the `-2` (2 to 1) or `-4` (4 to 1) switches. For example, a standard 44kHz stream will be played at 22kHz if run with `-2`, or at 11kHz with `-4`. Both options degrade sound quality noticeably, and the latter option will probably cause output quality to diminish to unacceptable levels. A potentially more satisfactory solution would be to use the `-m` switch for "singlemix," which will cause stereo streams to be played back in mono, again reducing processor resource consumption. Owners of relatively modern machines won't need to worry about these options at all, unless you're using mpg123 to feed an icecast server (see Chapter 8).

To stop mpg123 (as with any command-line program), tap `Ctrl+C`. Note, however, that if you're playing more than one file at a time (and you usually are), this will only stop the current file, and mpg123 will move on to the next one. To stop all activity, tap `Ctrl+C` twice in rapid succession. On some machines, playback will not stop immediately, as the audio buffers must be flushed first (though this seldom takes more than a couple of seconds).

**Output options.** If your audio hardware is not supported by mpg123, you can send output to `stdout` (standard output), rather than to the default system audio device. Thus, if you have a bizarre sound card at `/dev/somedevice`, you can tell mpg123 to output to the standard stream with the `-a` switch:

```
mpg123 -v -a filename.mp3 | /dev/somedevice
```

You can achieve some fairly interesting audio effects by speeding up or slowing down playback with the `-d` (doublespeed) or `-h` (halfspeed) switches, followed by a numerical factor. For example:

```
mpg123 -v -d 2 filename.mp3
```

will play your file twice as fast as normal, while

```
mpg123 -v -h 3 filename.mp3
```

will play the file at one-third of its normal speed. Negative values are not accepted. Note that these options merely chop out or double up on every *n*th frame, rather than changing the pitch of the currently playing file.

There are a large number of command-line MP3 (and general audio) players for Linux; mpg123 is only used here as an example. Most of them are fairly similar in their basic functionality, though their various switches and arguments may differ from one to the next. All should have documentation sufficient to get you up and running. In addition, most players come with source code you can tweak if you're in the mood, though most of the CLI players have already gone through quite a bit of evolution and are already well-optimized.



In addition to the ever-popular command-line players, there are also dozens of GUI players for X Windows. However, configuring a fresh Linux distribution to be able to run some of them may require more effort than you expect. While some, like FreeAmp, will run right out of the box on a RedHat distribution (depending on the packages you chose when you installed Linux), others require additional libraries or toolkits to be installed before they can be compiled. Before you grab the source code, note that many players are distributed in both source and binary formats, so you don't necessarily have to worry about compilation at all, though you still may require additional libraries to get up and running.

### *FreeAmp*

[www.freeamp.org](http://www.freeamp.org) Released as open-source software under the GNU Public License (GPL), FreeAmp is available for Linux and Windows (Figure 3-7), while development of versions for Mac OS, BeOS, and Solaris was underway at this writing. FreeAmp's development efforts are funded by emusic ([www.emusic.com](http://www.emusic.com)).

Programmers wishing to contribute source code to the FreeAmp development effort will find full documentation and details on accessing the FreeAmp CVS repository on FreeAmp's site.



If you have trouble compiling FreeAmp, you may need to install the egcs-c++ package. If you use a distribution other than RedHat, you'll need to install glibc 2.07 or higher as well. If you run into problems during the make step, you may need the XFree86-devel package installed too.



Figure 3-7. FreeAmp is an open-source player that runs under both Linux and Windows

Usage is very straightforward:

```
freeamp file1.mp3 file2.mp3 ...
```

Of course, you can just use a wildcard to play a whole directory of MP3s at once, as in `freeamp /path/*.mp3`, or pass FreeAmp a playlist instead. You can navigate amongst the songs you've loaded from the command line by clicking the Back and Forward buttons in FreeAmp's interface. You'll also find two vertically positioned dials in FreeAmp's interface—the one to the left of the main display controls volume, while the dial to the right let's you “scrub,” or skip through the tune. Note that you cannot close FreeAmp by clicking the close button in your X window manager—you must use the close button within FreeAmp's own interface.



Another excellent audio player for Linux is Andy LaFoe's AlsaPlayer ([www.alsa-project.org/~andy/](http://www.alsa-project.org/~andy/)). While AlsaPlayer is essentially a PCM player, it can work with plug-in modules that allow it to handle other formats. Unsurprisingly, an MP2/MP3 plug-in is included. Many Linux users appreciate AlsaPlayer for its low latency, variable pitch control, concurrent visual scopes, and multi-threaded design for better responsiveness.

### Xmms

[www.xmms.org](http://www.xmms.org) One of the most popular MP3 players for the Linux platform is, perhaps unsurprisingly, a more or less identical clone of NullSoft's WinAmp, as shown in Figure 3-8. Xmms, which once went by the name x11amp, supports the

vast majority of WinAmp's hotkeys, skins, and functions. And while Xmms also includes a plug-in architecture similar to WinAmp's, the specific plug-ins used with Xmms are, of course Linux native. Because WinAmp is covered thoroughly in the Windows section later in this chapter, see the WinAmp section of this chapter for usage details.



Figure 3-8. Xmms clones NullSoft's WinAmp more or less identically for the Linux platform

Xmms is also very keyboardable, and features the following hotkeys, as shown in Table 3-2.

Table 3-2. Xmms Hotkeys

Key	Function
z	Last (previous) song
x	Play
c	Pause
v	Stop
b	Next song
n	Eject (opens file requester)
Alt+S	Skin browser
Alt+O	Options
Alt+P	Preferences

Developers and hackers interested in contributing source code and modifications to Xmmms will find complete instructions on accessing the x11amp CVS repository at Xmmms' site and its mirrors.

### *XAudio*

[www.xaudio.com/](http://www.xaudio.com/) Bridging the gap between command-line and GUI players, XAudio can work in several ways—as a CLI-only tool (with switches similar to mpg123's), as an ugly-but-perfectly-functional GUI, or as a back-end player on top of which another GUI player can sit. To run XAudio from the shell, just use the standard syntax:

```
xaudio file(s).mp3
```

or invoke a playlist, like so:

```
xaudio -input=m3u /path/to/playlist.pls
```

To run the XAudio GUI, use the `mxaudio` command instead. The GUI interface is self-explanatory, and includes a very complete built-in help system, available from the Help → Help Browser menu.

Note that, at this writing, XAudio does not handle variable-bitrate-encoded files well at all, and you may hear a fair amount of “swishing” and skipping with downloaded files. While development of XAudio appears to have slowed down, the XAudio SDK (software developers kit) is still available for download, and does support VBR. The XAudio SDK, in fact, is used as the basis for a number of players and implementations, including MACAST, WindowsCE players, and embedded players in several small-footprint devices.

### *Mac OS*

The venerable Macintosh has a long history as the favored platform for audio content creation and consumption, but that hallmark may be beginning to fade. Fewer MP3 encoders and players are available for Mac OS than for Windows or Linux, and the tools that do exist come face-to-face with limitations in the Mac OS that may make efficient playback difficult on some machines. Despite the excellent floating-point processor performance provided by the RISC chips in the PowerPC family, Mac OS utilizes a cooperative multitasking model, which puts much of the burden of CPU sharing on applications rather than on the operating system, as is done in pre-emptively multitasking environments such as BeOS, Linux, and Windows NT (Windows 95/98 uses a little bit of both). As a result, Mac OS version 8.5 and earlier does not multitask effectively, and interrupts being called by other programs or by the operating system may cause MP3 players to cut out momentarily, or worse, to crash the system. For example, the popular MACAST decoder works very nicely, and supports visualization plug-ins similar to those available for other



platforms. But access one of MACAST's menus while a visualization plug-in is running, and the visualization will stop dead in its tracks until the mouse button is released. This is no fault of MACAST, but a limitation imposed by Mac OS itself. Fortunately, Mac OS X is scheduled to include much better interrupt handling, so all of this may be a moot point by the time you read this. The Mac holds its own in MP3 creation and playback, and the available players are of high quality.



At this writing, there are no MP3 players that work on 68 K Macs, as the system requirements of decoding MP3 are simply too great for 68 K machines running Mac OS (although 68 K Amigas did have MP3 players, so this isn't strictly a hardware limitation). However, you can use a tool such as SoundApp (covered later) to convert MP3 files to AIFF, and then play the AIFF files. Note, however, that in doing so you will be obviating the compression advantages offered by MP3, since the resulting AIFF files will be around ten times larger than the MP3 source files.

SoundJam, SoundApp, and MACAST hew pretty closely to the general playback basics outlined earlier in this chapter. Both work via drag-and-drop—MP3 files can be dropped directly onto the application interface or into a playlist. MACAST has a much nicer interface and supports skins, while SoundApp is (at this writing) the more stable of the two players. And don't forget: QuickTime 4 handles MP3 files quite nicely as well, though it may not offer the range of features you'll get from a dedicated MP3 playback utility.

### *SoundJam MP*

*www.soundjam.com* At this writing, Casady & Greene's SoundJam MP was the only complete, all-in-one MP3 solution for the Mac user, including encoding, playback, and visualization components into a single tool, as shown in Figure 3-9. In fact, most Mac users agree that SoundJam is the best MP3 solution for the Mac, period. If the default interface looks familiar, it's because the company has modeled it on the QuickTime 4 "brushed aluminum" look you've probably seen before. Don't like the appearance? SoundJam can be dressed up in skins—a sizable and growing collection is already available at SoundJam's site.

As with most Mac OS apps, playing files in SoundJam is a simple matter of drag and drop, and the interface controls are likewise intuitive. If you're a keyboard fan, note that SoundJam can be paused by tapping the Spacebar, or stopped by hitting the Apple Key. If you're entertaining at a party by playing music through SoundJam, you can run any number of visualization plug-ins, to provide a psychedelic light show in time with the music—tap Command+F to run your visualization plug-ins full-screen to complete the effect, and be sure to look for additional plug-ins on SoundJam's site.

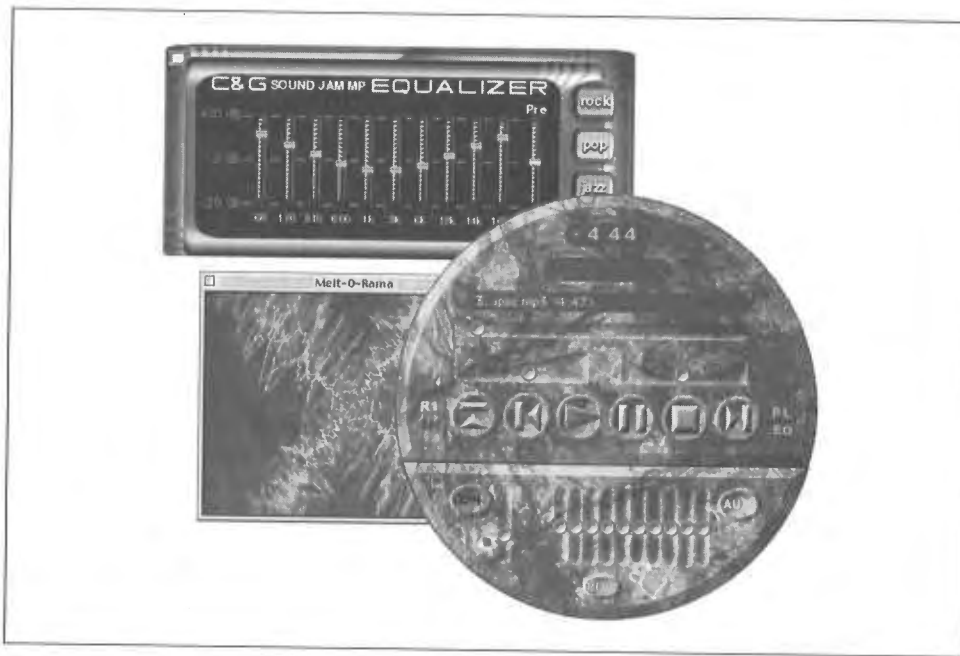


Figure 3-9. Most Mac users agree that SoundJam is the best all-around MP3 solution

SoundJam automatically maintains a “Master List” of all the tracks you’ve played, regardless where they live on your system, so once you’ve played a track in SoundJam, you’ll never again have to remember where you’ve stored it (though SoundJam makes this easy, too, by automatically placing the files it encodes directly in the Music folder inside the SoundJam application folder\*). If you don’t want new files added to the Master List, you can disable this option in the preferences, and new songs will be added to a new playlist window instead. Of course, you’re not limited to the Master List—just tap Apple+N to create a blank playlist window and drag your files into that instead. You can create as many custom playlists as you like.

SoundJam playlists will attempt to read the ID3 tags out of your files on the fly, so you’ll get a nice, sortable database, with separate columns for artist, track title, album name, and so on. Click column headers to specify which ID3 tag to show in each column, then use the left-most column header to specify the sort order. Tap Apple+F in a playlist window and you’ll be able to search through your database on the same criteria.

\* While this default location for audio files may be convenient, it’s a bad idea in general to store data files alongside application files; if you ever decided to delete the application folder, all of your data files are going to go with it. The makers of SoundJam really should know better.



When controlling the sort order via the picklist in the left column, pay special attention to the Play Order view—it's the only view that will let you drag songs up or down in the list, giving you manual control over playback order.

Like many MP3 players, SoundJam is fully “skinnable” (see Chapter 4 for more on skins). As you can see in Figure 3-9, SoundJam skins can have completely irregular shapes, and are not limited to a standardized layout, as are WinAmp skins. This is great from an aesthetic perspective, but does mean you may have to relearn the user interface a bit each time you try a new skin, as skin designers have the ability to move the main controls around, or even leave them out altogether. In fact, the default SoundJam skin doesn't even include a volume control. Instead, use the “Preamp” slider on the SoundJam equalizer for volume control when using the default skin.

SoundJam's built-in equalizer is similar to equalizers for other players (though one should take with a grain of salt the documentation's over-hyped claim that the equalizer delivers the software equivalent of \$2,000 worth of equalization hardware). The equalizer does, however, include a neat handful of EQ presets, including Arena, Classical, Hall, Hip-Hop, Jazz, Latin, Lounge, Piano, Pop, Rock and Vocal. You can even tweak these presets to custom values and save them to your hard drive for future use. See Chapter 4 for more on equalization techniques.

When you install SoundJam, it will automatically become your default CD player, which is absolutely fine by most people, since it has so many more features than Apple's built-in CD player. If you don't want this association made, you can undo it after the fact by using Apple's Monitors and Sound control panel. Conveniently, SoundJam can also play QuickTime 4 audio files, though it won't create a default association with that filetype, since the chances are you'll want to retain the QuickTime player itself for that. In addition, SoundJam will automatically associate itself with *.M3U* and *.pls* files so you can stream music down from Internet radio stations and sites such as MP3.com (see Chapter 8 for more on that). You'll find details on ripping and encoding from CD with SoundJam in Chapter 5.

#### *MACAST, MACAST Lite*

[www.macast.com/](http://www.macast.com/) @soft offers MACAST in two versions, which are similar in appearance and function but differ in the number of additional features offered. For example, the full version includes a graphical spectrum analyzer, an equalizer, complex skins, visual plug-ins, and is a closer competitor to NullSoft's WinAmp (Figure 3-10), while MACAST Lite (a.k.a. MALT) provides only basic playback functions. Both versions support MP2 and MP3 files (MP2 is favored by some owners

### *Setting MP3 File Associations in Mac OS*

For better or for worse, Mac OS has this peculiar notion that files "belong" to particular applications. That means that rather than having a collection of thousands of MP3 files that can be launched in your system-wide preferred MP3 player, each MP3 file will be launched in the MP3 player associated with that file. Typically (but not always), this will be the last player that handled that file. Thus, you may have three MP3 players on your system, and the files in your MP3 collection may all be launched in different players when double-clicked. If you're a ResEdit jock you may be able to modify your system's Finder application to associate all MP3 files with a single application. But working with ResEdit is not for the faint of heart, and we recommend that you always make a backup copy of any file you intend to modify, and do enough research so you know what you are doing. Your best bet for changing the application associated with an individual MP3 file is either to open the file using the "open" command in a given application's File menu (which will set the type to that application), or dig around in your favorite MP3 player's preferences panels and look for an option labeled something like "Set type on playback" or similar (in MACAST, the option is labeled "Set Types/Creators"). Then simply drag your MP3 files into the player and let it handle the filetyping for you, without your having to resave the files. There are, of course, several third-party utilities that offer custom filetype control in one form or another. File Buddy ([www.skytag.com/FileBuddy.stuff/FB\\_home\\_page.shtml](http://www.skytag.com/FileBuddy.stuff/FB_home_page.shtml)) is a good place to start.

Mac users may also find that Netscape Navigator automatically wants to play MP3 files via QuickTime. If this is not what you want, pull down Navigator's Edit → Preferences menu. Click on "Navigator," and then "Applications" to reach the helper apps configuration panel. Scroll down to audio/mpeg, audio/x-mpeg, or "QuickTime Audio MPEG" and double-click. In the "Suffixes" field enter .mp3, and click "Application," then "Choose." Navigate to the MP3 player you want to be associated with MP3 downloads. Set the "File Type" to "MPG3," and click OK to dismiss all of the dialog boxes.

of older Macs, partially because there were no decent MP3 encoders for the Mac until later in the game), as well as CD audio, variable-bitrate-encoded (VBR) files, and a range of other common audio file formats.

MACAST can be extended through the use of "engines," of which there are two types: some engines bring additional file format support to MACAST (e.g., MIDI), while others provide additional visualization support. You'll find a small collection of engines already installed in the Plugins folder inside the MACAST folder; downloaded engines should be placed here as well.



Figure 3-10. @soft's MACAST was once one of the most popular MP3 players for Mac OS

### MACAST Performance Tips

If you own a non-PowerMac machine and have trouble hearing any sound at all, make sure you don't have any other audio applications open in the background—they could be hogging the audio channel. PowerMac users should not encounter this problem. If you suspect that an MP3 file is corrupt, grab the MusicVac utility from [musicvac.MACAST.com](http://musicvac.MACAST.com), which will analyze MP3 files for unnecessary filename extensions, stray version resources, or ID3 comments where they shouldn't be. If you have a slower Mac and want to minimize MACAST's performance impact on your system, there are two important things you can do: Don't display the graphical spectrum analyzer, and select "Low CPU Mode (no equalizers)" in MACAST's Preferences → Toggles panel. You may also want to make sure that MACAST is allocated at least 10,000 K (10 MB) in the minimum memory size box located in the application Info panel (select the MACAST application icon with a single click and tap AppleKey-i).

### SoundApp

[www-cs-students.stanford.edu/~franke/SoundApp/](http://www-cs-students.stanford.edu/~franke/SoundApp/) While not as pretty as MACAST, Norman Franke's free SoundApp utility plays seemingly every known audio file format under the sun, not the least of which is MP3. As previously mentioned, SoundApp is also known for being a rock-solid stable performer, and may prove

more reliable for some users than other offerings, as shown in Figure 3-11. In addition, SoundApp works on 68 K Macs, though it does not decode MP3 on those machines—you'll need to use its built-in utility to convert MP3 to another format first. SoundApp also supports AppleScript, so you can create your own custom solutions (for example, you might create a script that jumps straight to a certain point in a certain file and plays a three-second segment every time a given event occurs on your system).

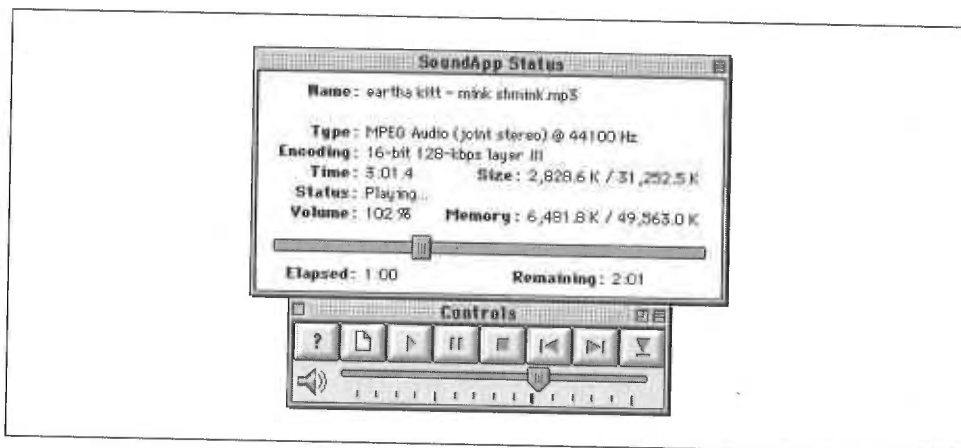


Figure 3-11. Simple and true (not to mention free), SoundApp is a trusty, reliable audio player that handles a huge list of audio file formats

SoundApp usage is very straightforward, and its controls are simple. Note that you can “scrub” through tracks as they’re playing by dragging the progress slider. To get more information on a currently playing track, option-click the Get Info button in SoundApp’s Controls palette.

## BeOS

The Be Operating System may be an underdog in terms of total number of users, but it deserves a place in this book for one big reason: BeOS is optimized from the ground up for handling audio/video streams like nobody’s business, and is quickly becoming a new focal point for the digital audio community. More germane to this book, Be, Inc. has stated that they intend to make BeOS the best platform for MP3 creations and playback, bar none. If you’re not acquainted with BeOS, cruise through their web site, at [www.be.com](http://www.be.com), or head straight for their “User Explanation Track,” at [www.be.com/products/beos/user-info-track.html](http://www.be.com/products/beos/user-info-track.html). You may also find the author’s own *BeOS Bible* from Peachpit Press a useful reference.

So what can you do with MP3 in BeOS that you can’t do in other operating systems? Digital audio enthusiasts will appreciate the fact that BeOS is capable of addressing audio hardware with extremely low latencies. For example, where a

Windows application might require 20–30 milliseconds to get a response back from the sound card, a BeOS application on the same machine can address the same audio hardware with 2–6 ms latency, making the system very attractive to audio application developers and high-end users alike. BeOS includes an mpeg codec in the operating system as a “media node,” which means users can save MP3 audio directly out of any audio application, including the system’s built-in CDPlayer and SoundRecorder apps. Because the BeOS filesystem functions much like a database, users can copy ID3 tags into “attributes,” so they can organize and query on custom collections of MP3 files in ways that aren’t possible in other OSes, without using special tools. Unlike Windows’ Explorer or the Mac OS Finder, Be’s Tracker talks directly to the Internet’s CDDb and displays audio CD tracks directly as intelligently named WAV files, so there’s no need for BeOS encoders to “rip” files prior to encoding. Because the system incorporates a unified sound architecture, users don’t need to jump through hoops to play lots of MP3 tracks simultaneously, or to create impromptu MP3 mixers. Meanwhile, the BeOS user can start playing half their MP3 files at +323% speed, or –227%, run a few movies, play tracks from audio CDs backwards, and so on. Thanks to BeOS’ fine-grained multithreading, the rest of the system stays responsive, whereas Windows users will start to notice the OS and applications reacting sluggishly long before this point.

In case you’re wondering, BeOS is every bit as easy to use as Mac OS. At the same time, the system offers a bash command-line shell for those who want it. The power of Unix, the grace of the Mac OS, and media-optimized from the ground up. What more could an MP3 hound ask for? At this writing, there are three primary MP3 players for BeOS, one bundled with the operating system and two provided by third-party developers.

### *SoundPlay*

*[www.xs4all.nl/~marcone/be.html](http://www.xs4all.nl/~marcone/be.html)* The definitive MP3 player for the BeOS is Marco Nelissen’s SoundPlay, a full-featured, highly customizable audio player that handles a broad range of audio file formats (Figure 3-12). SoundPlay is one of the few MP3 players in the world that can not only play MP3 files backward and forward, but can play them at up to +400% or –400% of the original speed, not to mention being able to play as many as a dozen MP3 files at once (and note that CPU usage isn’t the issue here—SoundPlay is very efficient—it’s the physical act of accessing so many disparate points on the hard disk that slows things down when you start trying to play more than 12 files at once).

Because BeOS audio applications address audio hardware with such low latency, don’t be surprised if you find you can do things like “scratching” your MP3 files DJ-style with a far greater sense of responsiveness than you can under Windows

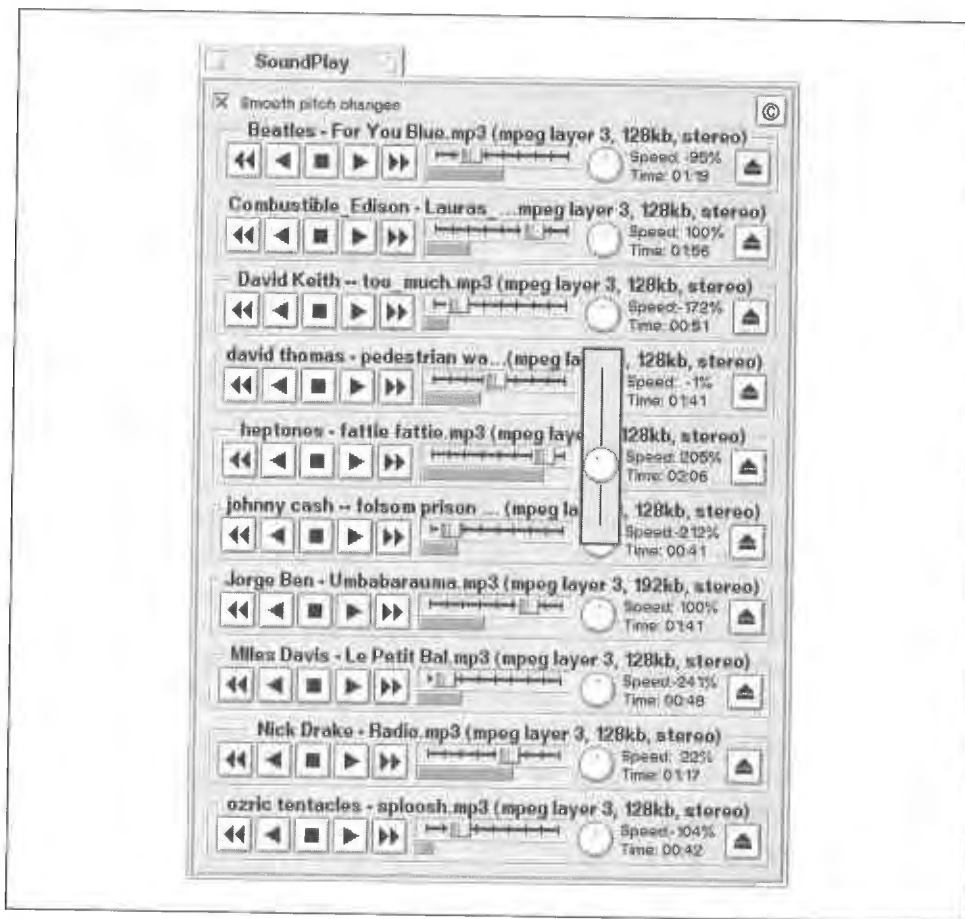


Figure 3-12. SoundPlay can take several other forms; load it up with multiple MP3s simultaneously for an impromptu multitrack console

on the exact same hardware. Try grabbing the blue progress indicator in SoundPlay and dragging back and forth quickly; it's not quite a turntable, but it's not a bad simulation. To increase the degree of control you have over scratching effects, try checking and unchecking the "Smooth pitch changes" checkbox. Keeping the box checked will give you snappier, crisper pitch changes, while checking it will make the sound more "rubbery."

When dragging audio files onto SoundPlay's interface, note a subtle but important distinction: If you drag your files into the lower part of the control panel, the dropped files will play in sequence. If you drop them into the upper portion, the files will all be played at once. The interface will change before the mouse button is released to let you know whether the dropped files will play sequentially or simultaneously.



### *Interface options*

In addition to the default layout, SoundPlay also has several other masks. From the BeOS "Be menu," select Show Replicants, then drag the small red handle at the lower right of a SoundPlay window. As you drag, an outline of the application will follow your mouse. Release over the Desktop and SoundPlay will be embedded in the Desktop itself. As with any BeOS Replicant, you can now close the SoundPlay application, and the player will keep on chugging, even between reboots. When SoundPlay is running as a Replicant, it adopts the color of the Desktop behind it and all of the window borders disappear. SoundPlay will now appear to be built into the operating system itself, as an integral component. Only its controls will be visible, floating in the Desktop.

Since SoundPlay also handles "BMessages" from BeOS color utilities, you can drag a color swatch from the color palette in just about any BeOS paint, graphics, or color manipulation program (Becasso, ArtPaint, and Nelissen's own ColorSelector are all good examples).

Finally, SoundPlay fully supports WinAmp skins, so you can keep on using the collection you've built up while experimenting with other MP3 players. By default, SoundPlay looks for your skins collection in `/boot/home/config/settings/skins`, though you can customize this location via the Preferences panel. To access your installed skins, pull down the Skins menu and select either a WinAmp skin or the default BeOS look and feel. When in WinAmp mode, just click in the lower-right corner of the interface to access the same menu.

### *Crossfade Settings*

Pull down Settings → Preferences and you'll be looking at a panel filled with options, most of which are self-explanatory. One of the most interesting options here is a button labeled "Crossfade Settings." When playing files sequentially (as when playing from a playlist or a collection of dropped files), SoundPlay will fade one file's volume down when it's a few seconds from completion, and simultaneously fade the next file's volume up. The fadeout start, fadeout duration, and next file start time can all be configured for all file sequences by selecting the Default Fade Setting entry in this panel and entering times for these parameters (though the defaults are quite good).

But if that sounds cool, hang on, because it gets better. The BeOS filesystem supports "attributes" of arbitrary size and type. Attributes are simply data that is associated with a file but is not a part of the file itself. SoundPlay takes advantage of filesystem attributes in several clever ways, and one of them is to let you drag any MP3 file into this panel and custom-set its crossfade settings. If these crossfade attributes are found on a currently playing file, they'll be used instead of the default settings, which lets DJs and other serious audio

heads tweak out playlists to a level of customization not possible in any other player on any other operating system. Custom crossfade attributes can be removed by selecting a file and clicking Clear Settings of Selection.

Crossfade settings can also be accessed by right-clicking the title of the currently playing song and selecting Special → Crossfade from the context menu. It's important to note that crossfade settings are independent of particular playlists, and will apply regardless of which playlist a particular song happens to show up in.

### Media Player

Naturally, Be includes a Media Player application with the system, capable of playing any audio or video stream that has a corresponding codec installed, as shown in Figure 3-13. Since BeOS understands MPEG natively, the MediaPlayer is capable of playing MP3 files. Media Player is nothing fancy—just a simple progress indicator/scrub bar and a set of standard controls. If you're downloading MP3 files from the Internet, MediaPlayer will snap into action automatically as soon as the download is complete, as long as you haven't associated your MP3 filetype with another player (see sidebar, "Associating MP3 Files with BeOS Applications"). You can of course, also drag MP3 files onto Media Player or navigate to them via the Media Player's File → Open menu.

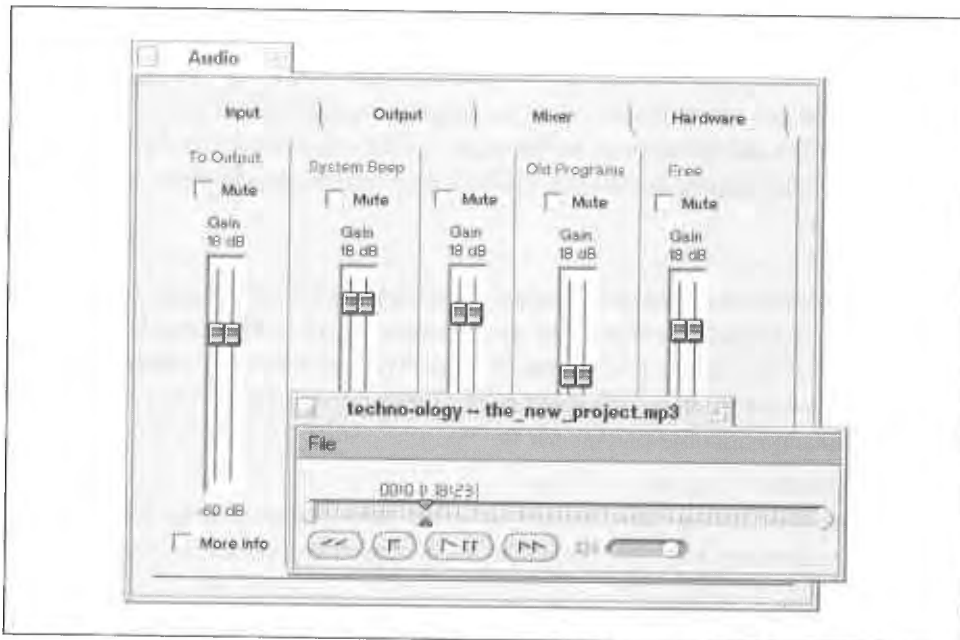


Figure 3-13. The BeOS Media Player application can play any audio/video media stream for which an installed codec is found, including MP3

### CL-Amp

[www.be.com/beware/](http://www.be.com/beware/) Claes Löfqvist's CL-Amp is a very capable MP3 player very similar in form and function to NullSoft's ever-popular WinAmp. CL-Amp (Figure 3-14) is intended as a free WinAmp clone for BeOS—it doesn't take advantage of native BeOS performance and UI features like SoundPlay does. CL-Amp is fully compatible with the thousands of skins available for free download (Chapter 4), so you can customize CL-Amp's appearance 'til the cows come home. Playing MP3 files with CL-Amp is a simple matter of dropping them onto its icon or directly into its interface or playlist window. Alternatively, drop a pre-existing playlist onto CL-Amp.

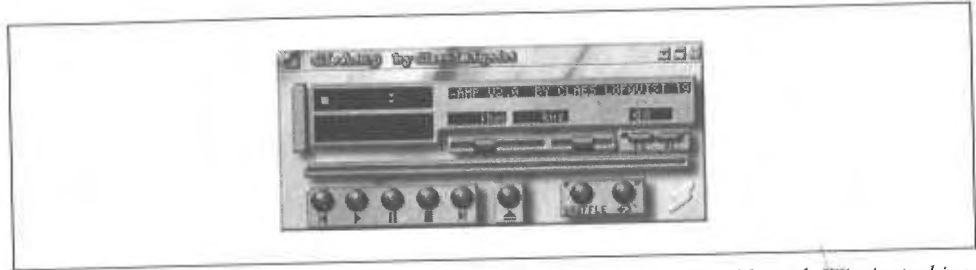


Figure 3-14. CL-Amp for BeOS: A freeware MP3 decoder fully compatible with WinAmp skins

To access CL-Amp's dozens of configurable behaviors, right-click anywhere in its interface. Items in the menus are self-explanatory, but here's a quick rundown:

#### Double mode

Double mode (Alt+D) will cause CL-Amp to spring to twice its usual size, which may make it easier to access some of the less-visible interface controls. You may also appreciate double mode if you run your monitor at a very high resolution.

#### Skin selector

Accessing the skin selector launches a small menu listing a combination of all skin folders found either in the Skins subdirectory, CL-Amp's installation directory, or in `/boot/home/config/settings/skins`. The latter location makes it possible for CL-Amp to share a collection of skins with SoundPlay (this chapter) or other skins-compatible players for BeOS.

#### Drag and Drop

From this menu you can control whether files dropped onto CL-Amp are added to the currently playing playlist or *become* the playlist.

#### Spectrum Analyzer Mode

Like most Amp-oriented players, CL-Amp presents a graphical display of the current signal levels as your music is playing, much as you would see on some audio equipment. This mode menu lets you control the appearance of this graphical display.



If you use BeOS, Linux, and Windows on the same machine, why duplicate your skins collection on multiple drive partitions? Since BeOS can read and write to Windows partitions as easily as to its own BFS filesystem, and can read Linux ext2fs and Windows NT's NTFS filesystems natively, you can simply make your BeOS skins directory a link to your Windows skins directory. Mount your Windows partition by right-clicking the Desktop and selecting it from the Mount submenu. Right-drag your Windows skins folder into `/boot/home/config/settings` on your BeOS partition, and choose Create Link from the context menu. Rename the symlink to "skins" and you're done. Note, however, that you'll need to make sure your Windows volume is always mounted when you're in BeOS. Choose Settings from the Mount submenu and choose All Disks in the Boot section of the Mount Settings panel.

#### *Speed Button Snap to 100%*

CL-Amp will let you play audio streams at anywhere from 50% to 200% of their original speed, either temporarily or throughout the course of an entire track. When this option is selected and you manipulate the pitch bender, the speed will snap back to 100% as soon as you release the mouse. With it deselected, the pitch will stay "bent" until you select another speed or select this menu item.

#### *MP3 and the Tracker*

Speaking of attributes, the BeOS filesystem lets you do some pretty amazing things when it comes to organizing, sorting, and querying for data associated with your MP3 files. Every time SoundPlay encounters an MP3 file without custom attributes, it writes the file's total playing time and bitrate to two new attributes.

In addition, a number of BeOS utilities are capable of extracting the song's Title, Artist, Album, Year, and Comment information from its ID3 tag information (Chapter 4) and writing this data to five additional attributes. The result is the ability to use the BeOS file manager, the Tracker, to sift, sort, edit, and organize your files along criteria Windows, Mac OS, and Linux can't even dream about (Figure 3-15).

To enable this functionality in BeOS, download *tag2attr* and *attr2tag* from [www.be.com/beware/](http://www.be.com/beware/) and install them in `~/config/add-ons/Tracker`. Select any collection of MP3 files in the Tracker, right-click your selection, and choose *tag2attr* from the Add-Ons submenu. Of course, you can also edit your tag information directly in the Tracker as well. Want to make your customized attribute data stick, so that the file's *actual* ID3 tag data will be converted as well? Just edit your attributes as you would anything else in the Tracker and then run the *attr2tag*

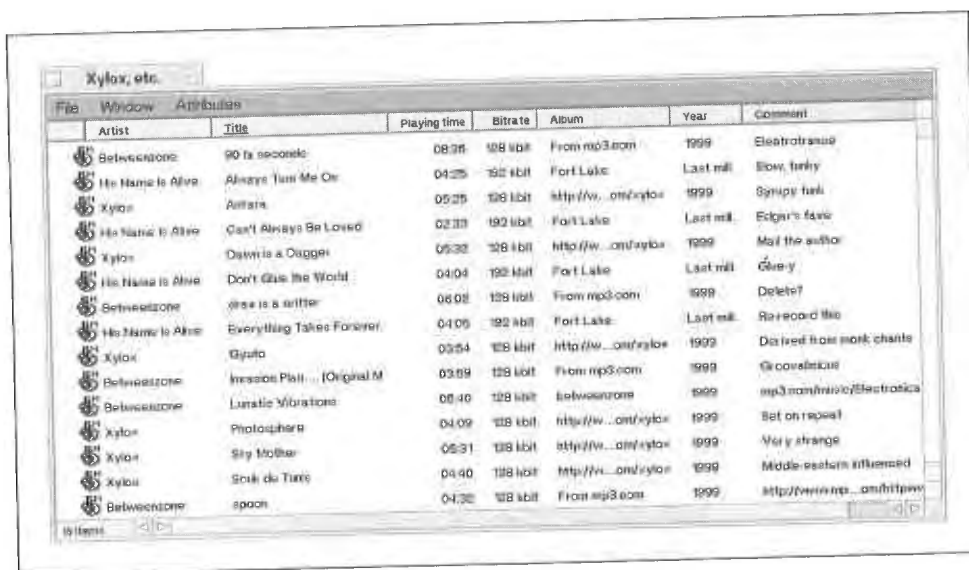


Figure 3-15. Tracker windows are like a customizable MP3 database

Add-On. Move the file to another operating system, open it in an MP3 player, and you'll see that the changes you made from the Tracker in BeOS will have been transferred right into the file itself.

If you prefer to work from the command line, download `id3attr` from [www.bebits.com](http://www.bebits.com) and install the `id3attr` binary in `~/config/bin`. Launch a Terminal window, `cd` to a directory containing MP3 files, and type:

```
id3attr *.mp3
```

All ID3 tags for all MP3 files in the current directory will be copied to corresponding attributes in the file system.

Simply copying ID3 tags to attributes will let you sort and sift through fields in the Tracker, but if you want to be able to run sophisticated system queries on ID3 data, you'll need to make sure these attributes are being added to the system index as they're created. While you can do this manually if you like, make things easy on yourself and download a copy of Jonas Sundstrom's MP3 AI, which will automate the creation and management of indexes and attributes. Sundstrom also makes available MP3 Mix, MP3 Flashlight, and the add-ons mentioned earlier. Taken together, they're an unbeatable MP3 management system for BeOS.

## Obtaining MP3 Files

Before you begin building up your MP3 collection, it's important to understand the basic legal issues behind the distribution of what is the intellectual and creative property of artists and the recording industry. As with most technologies, MP3 has

### *Associating MP3 Files with BeOS Applications*

Of course any MP3 player for BeOS will accept any file dragged onto its interface, but how do you get an MP3 player to come up in your favorite MP3 player automatically when double-clicked? BeOS uses an expansion of the Internet's MIME-typing convention to give all files appropriate types, and lets users associate these types with any capable application, either globally (system wide) or locally (on a file-by-file basis).

To set the global association in BeOS, open the Preferences → FileTypes panel and navigate to the Audio → MPEG Audio File entry, then use the Preferred Application picklist to select from any of the MPEG-capable players on the system. Double-clicking any MP3 file will now cause it to be launched in that application, *unless* you've overridden the preferred app for that file at the local level. To change the preferred handler for an individual file or group of files, select it or them in the Tracker, right-click the selection, choose Add-Ons → FileType, and select the preferred application.

Note that two different MIME types are commonly used for MP3 delivery on the Internet: *audio/x-mpeg* and *audio/mpeg*. Technically, the former is more correct, though there are plenty of the latter type out there as well. In any case, you may need to make the changes above for both filetypes when setting your global preferences.

both legal and illegal applications. Despite the excitement surrounding the prospect of the public undercutting the traditional distribution mechanisms of the recording industry, and no matter how you might feel about the moral rectitude of the profit-mongering of these traditional music supply chains, the mere fact of an MP3 being available to you does not automatically make it legal to help yourself. It's easy to steal candy bars, and you may feel they're too high-priced, but that fact doesn't make it legal to steal. Keep these points in mind as you begin to build up your collection:

- For the same reason that it's legal to make backup copies of software that you've purchased (so you still have access to it in case the physical media is damaged), it's perfectly legal to create and listen to MP3 copies of music that you've purchased. It is not, however, legal to make these copies available to other people via the Internet, through portable devices, or on any other storage media. You bought the rights to own a copy of that music—you didn't buy the rights for the rest of the world to own a copy as well.
- Many artists and record companies make their music available for download from the Internet (either for free or for a small fee) as a way to promote themselves. There are many sites that legitimately distribute hundreds or thousands

of these tracks, and it's perfectly legal to download and listen to them. At the same time, hundreds of thousands of ordinary Joes make their MP3 collections available over the Internet for other people to download. It is illegal for them to do this, and it's illegal for you to download and come into possession of these tracks. If you're in doubt about whether music you're about to download is being legally distributed or not, poke around on the site in question looking for the fine print. If the site is distributing legal music, it will say so, or will be clearly endorsed by the artist or the label. Sites distributing music illegally will generally say nothing, or may even be brazen about their illegality.

You can read much more about all of the legal bits surrounding MP3 are discussed in detail in Chapter 7, *The Not-So-Fine-Print: Legal Bits and Pieces*.

### *Via the Web*

Most MP3s distributed online are accessed by simply clicking a link to an MP3 file, as illustrated in Figure 3-16. Click a link to an MP3 file in your browser and a panel similar to (1) should appear. Select a location on your hard drive and the download will proceed (2). Once complete, launch WinAmp (3) or your favorite player and drag the file onto its interface, access its file panel to navigate to locate the downloaded file, or through command-line options (see the rest of this chapter for details and examples). Turn up the speakers and close your eyes.

Depending on the way your browser and preferred MP3 player are configured, downloads may be streamed directly to your MP3 player without being saved to a preferred location first. The most common culprit of this problem is the Microsoft MediaPlayer. After installing a recent version, clicking links to MP3 files in Internet Explorer will result in direct playing of the file, thus making it difficult to save the file for later use. To sidestep this situation, try *right-clicking* MP3 links instead, and choosing the "Save Target As..." from the context menu instead. In Netscape Navigator, you can always hold down the Shift key when clicking any link to force the file to be downloaded to a preferred location rather than being sent to a helper application.

### *Streaming MP3*

Increasingly, many sites are dishing up MP3 as streaming broadcasts, rather than as direct downloads. In these cases, the preferred MP3 player will always receive the audio stream directly—saving to disk is not an option without resorting to clever hacks (some of which are in Chapter 4 and elsewhere in this book). You'll generally know if you're visiting a streaming/broadcast site because it will clearly say so. A good way to tell before clicking a link is to hover your mouse over an MP3 link and glance at the browser's status area (generally in the lower-left corner of the

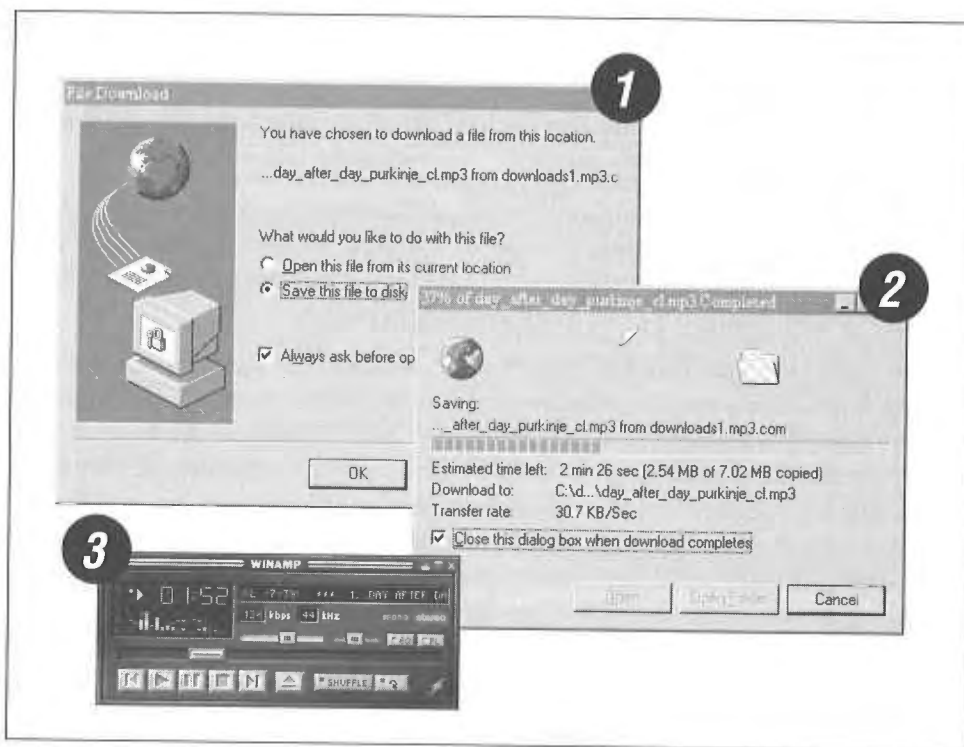


Figure 3-16. Your basic download

browser window). If the URL is of the form *http://www.somesite.com:8000* (has a port address appended to the end), or includes some heinous combination of port addresses, IPs, and domains, and includes pls extensions:

*http://www.somesite.com/cgi-bin/playlist.pls?addr=219.147.18.44:9000&file=filename.pls*

then you know you're about to access a broadcast stream. See the section "Working with MP3 Streams" in Chapter 4 for more information.

### Via FTP

FTP, or file transfer protocol, is the Internet's most basic means of making files available for transfer between a server and a client, and predates the world wide web by a couple of decades. It's important to note that pirated files are often made available via FTP, so be careful. If the files are being provided legitimately by an artist or label, the chances are very high that they'll offer them through a standard web site, not through an FTP server. In addition, pirate FTP sites generally have very brief lifespans, so links and messages pointing to them are often out of date by the time you try to access them. Trying to hunt down a working FTP site can



be a time-wasting wild-goose chase, and often leads only to poorly recorded encoded songs that don't match your own taste. The best way to find MP3 FTP sites is to scan Usenet groups. In particular, *alt.music.mp3* often carries many listings (although, again, you'll find that the vast majority of them are dead and gone by the time you try to access them).

If you're serious about building up a good MP3 collection, you'll probably have much better results downloading from legitimate web sites and by encoding your own tracks. That said, there *are* legitimate instances of MP3 files being made available through FTP sites, so a basic introduction is in order.

FTP sites will be made available either on a pre-established domain, or on a specific machine's IP address. In other words, an FTP address may take the form *ftp.somesite.com* or something like *103.241.10.98*. An FTP address given as a URL will be the machine's address prepended by *ftp://*, as in *ftp://ftp.somesite.com*. Such a URL can be accessed through most browsers, which will set up a browser-specific interface onto the site, and files can be downloaded from the site in the usual way. However, you'll have much more control and flexibility over your FTP session by using a dedicated FTP client. Most importantly, FTP clients will let you download multiple files, or even entire directory structures at once. Good, graphical FTP clients include WS\_FTP or CuteFTP for Windows, Fetch for Mac OS, Igloo for Linux (Figure 3-17), and NetPenguin for BeOS, though there are many, many more options than these. Check your favorite software library for options.

FTP sites will either be set up for anonymous access, in which case you can simply log in with the username "anonymous" or "ftp" and use your own email address as a password, or will have a specific username/password pair set up by the site's maintainer, which should be provided on the site or in the message providing access to the site.



---

If you can't find a username/password pair listed and anonymous access doesn't work, try entering the word "mp3" for both the username and password.

---

Once logged in, files can generally be dragged from the remote server to your local machine. Some clients require navigating directory structures on both the remote host and the local client and then pressing a transfer button of some kind.

To use them, just open a shell or Terminal window, type `ftp address`, enter your username and password, and use the `cd`, `ls`, and `pwd` commands as you would on your own machine. To retrieve a file, type `bin` to make sure you're in binary mode, `hash` to make sure you see a progress indicator as the transfer proceeds, and `get filename` to retrieve a specific file. Typing `mget` will let you retrieve

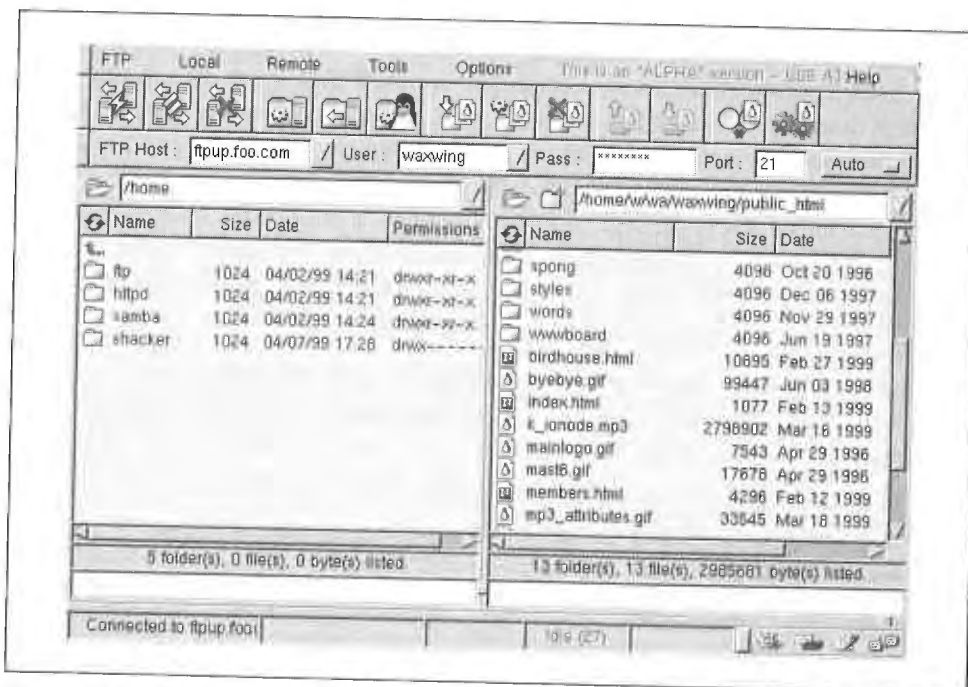


Figure 3-17. Igloo for Linux is one of many graphical FTP clients that make it easy to transfer files en masse from a remote server to your local machine

multiple files at once. For example, `mget *.mp3` will let you retrieve all MP3 files in the current directory. However, to make this option useful, you'll want to turn off prompting before downloading, so that you aren't asked for confirmation before transferring each file. Type `prompt` to turn off prompting, and optionally type `prompt` again later to turn it back on. Transferred files will end up in the directory from which you started the session. To change the download directory on your local machine, type `lcd pathname`.

## Download Managers

If you've got a slow Internet connection, or if you'd like to download a lot of files from a lot of different sites while you sleep, you may want to get yourself a good download management utility. These will let you copy (or even drag in) links to files you want to download from any number of different web and FTP sites. Establish a download directory, fill the queue with URLs, and let 'er rip.

Windows users should check out ReGet ([www.reget.com](http://www.reget.com)), while Linux and BeOS users should have a copy of wget. With wget installed in your path, just create a simple shell script consisting of wget lines followed by URLs. Save the script in your path with a name like "hoover," change to your preferred download directory, and execute hoover. Go to bed. If wget isn't on your system already, you'll be able to find it easily at your favorite software library.

### *Via Usenet*

One of the least convenient ways to download MP3 files is via Usenet. Because Usenet is text-based, it's not possible to post binary attachments directly into a Usenet message. To get around this, a system called uuencoding is used to transform binary files into raw text. To humans, uuencoded binaries appear as a lengthy stream of ASCII garbage, and a tool called a uudecoder must be used to transform this stream back into its original binary state. Fortunately, most modern newsreaders include the ability to decode uuencoded messages with the click of a button. Consult your news client's documentation for details.

If you want to do it by hand, you'll need a uudecoder installed on your system. Linux and BeOS users will find this tool already installed on their systems, while Windows and Mac OS users will need to download a separate uudecoder. Save the message in question to a text file on your system, giving it a name such as *filename.uu*. From a shell prompt, type:

```
uudecode filename.uu
```

A new MP3 file should appear in the same directory. Again, note that Usenet is a terrible way to distribute and access binary data in general. Use it if you need to, but I don't recommend wasting your valuable time when other distribution mechanisms are readily available. If you want to uuencode your own files for posting to Usenet, note that there are two subtle variants of the uuencode command floating around out there. On some systems, you'll simply be able to type:

```
uuencode filename.mp3 > filename.uu
```

However, some versions of uuencode require you to specify an additional filename, to be written into the file's header. This name will become the name of the resulting binary when the user decodes the file on their end. For example, if you want the recipient to end up with a file called *grotto.mp3*, you would use this syntax:

```
uuencode filename.mp3 grotto.mp3 > filename.uu
```

### *Roll Your Own*

There are a number of advantages to encoding your own MP3 files. First, you don't have to deal with hunting around and waiting for lengthy downloads, and all bandwidth questions simply vanish. Second, you don't have to sift through other people's musical tastes—just encode what you like from the albums you already own and build a perfect collection with no dreck. Third, you have complete control over the quality of your encodings. You can set a bitrate with a higher quality than the de facto standard of 128 kbps, clean up your recordings in audio manipulation software prior to encoding, and so on. Finally, by encoding your own tracks

you have 100% assurance that your entire collection is legal. With the quality and speed of some of the MP3 ripper/encoder packages out there, you can encode an entire album in a fraction of the time it would take to actually listen to it—the process really is a piece of cake. Ripping and encoding is covered in detail in Chapter 5, but here's the bullet-point brief:

#### *Ripping*

If recording from CD, download and install a “ripping” utility that will let you access the real audio data on your CDs, rather than the 42-byte *.cda* handlers that normally appear when an audio CD is viewed through your file manager (only Windows users will see *.cda* handlers). If recording from another source, install audio software capable of saving the incoming audio stream to WAV, AIFF, or RAW audio format.

#### *Encoding*

Download and install an MP3 encoding utility, which will take the audio data you've ripped or saved and run it through the perceptual coding algorithms covered in Chapter 2, spitting MP3 files out the other end of the process.

While that's the basic process between all ripping and encoding, most people use tools that handle both processes for you behind the scenes, so you can rip and encode at the same time and never worry about the intermediate step. With the right tools, the entire process becomes a no-brainer. It's really that simple, though there are still many options and caveats to consider if you're looking for excellent results.

### *Why Is It Called “Ripping?”*

The practice of extracting music from an audio CD to a digital audio file on your hard drive is called “ripping.” Why? Because, as you'll see in Chapter 5, audio CDs don't use a filesystem corresponding to that of any particular operating system. Your OS's file manager can't see the tracks like it sees normal files on a data CD, so you either need to trick your operating system into seeing the tracks' file “handles” as actual WAV or AIFF files, or use a special utility to read the raw data from the audio CD and turn it into a WAV or AIFF file. Because special techniques or tools are required and you're not doing a straight file copy, another name for the process was required. No one seems to know where the term “ripping” originated, but it sounds intuitive and has been floating around in the computer vernacular for many years.

Don't worry: ripping is a very simple process, and many tools completely hide the process for you, so you may not need to think about it at all.

## Organizing Your Collection

Before your collection starts to grow, it's worth giving a bit of thought to establishing file naming conventions and an organizational system. While you can always use Explorer/Finder/Tracker to organize your collection later, you may find it useful to think a bit about filenames and directory structures a bit before you start downloading and encoding, as a number of variables enter the process. You'll be happier later on.

### Naming Your Files

When encoding your own files, you should have total control over the way your files are named, although many players do impose some limitations in this department, such as disallowing spaces in filenames even for those who choose to use them. Dig around in the options and you should be able to find controls for choosing among many possible formats (more on that in Chapter 5). If you're downloading files from the Internet, however, it's a different story—files arrive with whatever naming convention was used by the person or organization who posted the file to begin with. Many files you'll find out there are simply named after the song, e.g., "Blue\_Jay\_Way.mp3". Would it make more sense to name a file with the artist's name as well, e.g., "Beatles-Blue\_Jay\_Way.mp3"? That depends on how anal you are. First off, keep in mind that the name of the artist is usually—but not always—stored in the file itself, in the form of its ID3 tags. Thus, even when the artist's name isn't in the filename, it will appear in the MP3 decoder's player's interface as the file is being played. However, there are files floating around out there with no information whatsoever stored in ID3 tags (Figure 3-18), so if you don't give your files a meaningful name when you download them, you may never discover the actual artist or track name.

The second question to consider is whether to allow spaces in your filenames. While all modern operating systems allow filenames with spaces in them, spaces can present certain difficulties in some operating systems when working from the command line. For example, Unix/Linux/BeOS shell environments treat spaces as separators between distinct commands or filenames. You might tell the Linux decoder mpg123 to play three files in a row with the command:

```
mpg123 file1.mp3 file2.mp3 file3.mp3
```

But trying to play the file from the previous example in this way:

```
mpg123 Beatles - Blue Jay Way.mp3
```

would cause mpg123 to hork, since it would look for files called "Beatles", "-", "Blue", "Jay", and "Way.mp3". There are two possible solutions to this situation: Surround the actual filenames in quotes:

```
mpg123 "Beatles - Yellow Submarine.mp3"
```

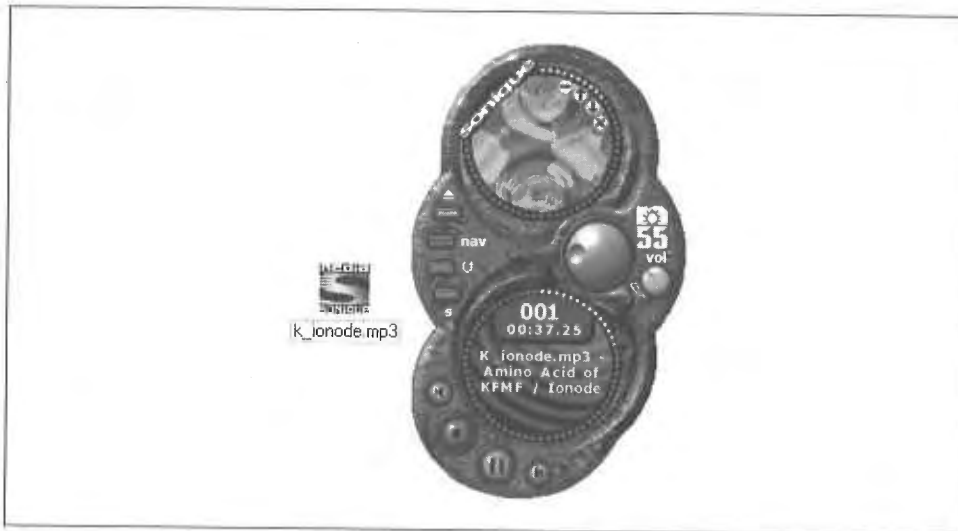


Figure 3-18. The artist's name and song title is embedded in the file itself, eliminating the need for their mention in the filename

or Simply replace all spaces in the filename with underscores and separate the artist from the track title with single or double dashes, e.g., "Beatles--Yellow\_Submarine.mp3". For this reason, you'll find that many popular MP3 download sites, such as MP3.com, offer files for download with underscores in place of spaces. Whether you choose to replace these with the more aesthetically pleasing spaces or not is up to you and your needs. As a general rule, users who plan to play or manipulate files from the command line may prefer to retain the underscores. This is especially true if you plan to use scripts to organize, sort, or create playlists/indices for your MP3 collection, as pre-existing scripts may assume that no spaces are present in filenames.

### *Building Logical Directory Structures*

Users intent on building up a large collection—especially collections including complete or partial albums—may want to create a directory/folder structure that makes it easy to group and organize artists and their albums. For example:

```
c:\data\mp3
c:\data\mp3\albums
c:\data\mp3\albums\Beatles
c:\data\mp3\albums\Beatles\Magical_Mystery_Tour
c:\data\mp3\albums\Beatles\Magical_Mystery_Tour\Blue_Jay_Way.mp3
c:\data\mp3\albums\Beatles\Magical_Mystery_Tour\I_Am_the_Walrus.mp3
c:\data\mp3\singles
c:\data\mp3\singles\Rolling_Stones--Paint_It_Black.mp3
```

### Notes on Numbering

If you're encoding your own albums, it's a good idea to include track numbers from the CD in the filenames themselves. Without them, the files will appear in the encoding output directory in alphabetical order, rather than in album order. But an album is a complete entity, not just a pile of singles. Artists put a great deal of effort into determining the ideal order in which tracks should be heard. If you choose to respect this prerogative of the artist, have your encoder prepend track numbers to the filenames so they appear in Explorer/Finder/Tracker in the proper order. When you drag the folder onto your MP3 player, you'll hear the album as it was meant to be heard, not willy-nilly. And remember: If there are more than 9 tracks on the album, you'll need to make sure the encoder uses "01" rather than simply "1," etc.—otherwise, your computer will order track 10 before track 1, and so on.

These are DOS/Windows-style paths, but the same principle of organization applies equally to Mac OS or Unix users. Note that in this example, songs stored under the albums hierarchy are named after song titles only, but songs stored in the singles directory are given both the artist and song name, since artist names in filenames are redundant if the directory structure is logically organized. In addition, some users may want to further subdivide their hierarchy into musical genres. For example:

```
c:\data\mp3\albums\rock\Beatles
c:\data\mp3\singles\jazz\Miles_Davis
```

After much experimentation with various organizational hierarchies, I've finally settled on a scheme that works well for me. Rather than separating out albums and singles, I realized that most of the time I only encoded portions of albums, and that as I gathered singles, a lot of them were by the same artists. My hierarchy now looks like this (BeOS/Linux-style paths used here):

```
~/mp3/artists
~/mp3/artists/Captain Beefheart/Trout Mask Replica
~/mp3/artists/Captain Beefheart/Trout Mask Replica/11 - China Pig.mp3
~/mp3/artists/Captain Beefheart/Trout Mask Replica/20 - Ant Man Bee.mp3
~/mp3/artists/Captain Beefheart/Lick My Decals Off/09 - Petrified Forest.mp3
~/mp3/artists/Hank Williams/...
~/mp3/encode
~/mp3/downloads
~/mp3/moonshine
```

Because each artist has a directory of their own, it doesn't matter whether I'm storing singles or albums—when I want to hear a particular artist, I just drag their folder onto my MP3 player. I don't even bother making playlists for artists anymore. However, if you tend not to encode complete albums very often, it may not

make sense to create album subdirectories under the artist names—you may want instead to just include the album in the filename and put all tracks by a given artist into a single directory. Some users also like to store the source's format in the filename, designating whether the track came from an LP, CD, 7-inch single, etc. The *encode* directory is, obviously, used as a container for tracks currently being encoded in the background. Because I've configured my encoder always to create artist and album subdirectories if they don't exist, I can easily drag the newly created folders into the artist hierarchy after I've listened to the encodings and determined that their quality is satisfactory. The *downloads* directory is used as a temporary holding bin where downloaded tracks live until I've decided whether to keep or delete them. Finally, the *moonshine* directory is used to store miscellaneous tracks. As soon as there are two or more tracks by the same artist in moonshine, I create an artist folder for them in the artists hierarchy.

However, it may not be necessary to organize this tightly, since (as you'll see in Chapter 4), you can always create custom playlists based on artist, genre, and other arbitrary criteria. Tools are even available that will help you to generate custom playlists on specific criteria by hoovering ID3 tag information out of your files, thus treating your entire collection as a queryable database. Nevertheless, as your collection grows, you'll find that you appreciate having a well-kept directory hierarchy more all the time.

In the end, how you choose to organize your collection is completely up to you and your personal preferences. If you're the type who stacks hundreds of CDs up in random piles throughout your house, you may be perfectly happy with a single giant directory containing thousands of individual MP3 files. If, on the other hand, you're in the habit of meticulously organizing your LP/CD collection by genre and artist, you'll probably want to do something similar for your digital music collection.

### *Storage Requirements*

While the average MP3 file is a mere one-tenth to one-twelfth the size of its corresponding uncompressed raw audio track, don't think for a moment that the size of your collection won't add up faster than your last grocery receipt. At roughly 1 MB per minute, expect a typical album to occupy 40–70 MB of disk space. At that rate, a one-foot stack of CDs will hoover up about a gigabyte, and if you opt to encode your MP3s at a higher-than-average bitrate\* than the average in order to achieve better audio quality, you can expect that space to be consumed even more quickly.

---

\* Remember: Bitrate is a measure of the amount of storage space that will be allocated to each second of audio, and is measured in kilobits per second, or kbps. Higher-bitrate files sound better and consume more disk space, and vice versa.



Table 3-3 demonstrates the results of a real-world test to determine the relationship between file size and encoding bitrate. Of course, audio quality corresponds directly to file size, and the files encoded below 128 kbps carried enough audible compression artifacts on our test system to be not worth keeping, and the very low bitrate files sounded like total dookey. It's worth noting, however, that there are legitimate uses for very low bitrate encoding, notably in the field of voice recording (as opposed to music), where fidelity is of less importance. Voice recordings—lecture notes, for example—may also be very lengthy, which further encourages the use of low bitrates to minimize disk space.\*

*Table 3-3. MP3 File Size is Directly Related to the Bitrate at Which It's Encoded (File Sizes and Compression Ratios Have Been Rounded)*

Bitrate (kbps)	Filesize (megabytes)	Compression Ratio
Original file	40.08	1:1 (none)
16	0.46	87:1
32	0.92	44:1
64	1.83	22:1
96	2.74	15:1
128	3.64	11:1
160	4.55	9:1
192	5.46	7:1
256	7.28	6:1
320	9.10	4:1

In Table 3-3, a sample file of 3:52 in length (Jorge Ben's "Ponta de Lanca Africano," a.k.a. Intel's "sock monkey" song) was converted to WAV format, creating a (more or less) exact bit copy of the data on the audio CD, and weighing in at a hefty 40.08 MB. The WAV file was then encoded at various bitrates using Music-Match Jukebox with default settings. As you can see, even when encoding is done at the highest possible bitrate, thus delivering a file indistinguishable in quality from the original CD, the MP3-encoded file is less than a quarter the size of the original (though it still gobbles 9.1 MB of disk space). At the Internet's de facto standard bitrate of 128 kbps, the resulting MP3 file is only one-eleventh the size of the original. Using lower bitrates achieves even more impressive file sizes, though audio quality degrades quickly below 128 kbps. If you're going to be encoding your own music, this book strongly recommends using bitrates equal to or higher than 128 kbps. More details in Chapter 5.

\* But keep in mind that MP3 is not the ideal format for voice recordings; if you do a lot of this, you may be better served by going with RealAudio rather than MP3.

Clearly, if you're going to get serious about MP3 collecting, you won't be storing everything on your machine's hard drives for long before a more flexible storage solution is in order. While hard drives just keep on getting bigger and cheaper, having more storage available tends to lead people to just start encoding more—the author purchased a 36 GB drive just for MP3 storage, and had it one-quarter full in three weeks after going on a marathon encoding binge. Fortunately, a plethora of external and removable storage options have become relatively affordable in recent years. Don't bother with Zip or other 100–200 MB solutions—at the rate those fill up, you may as well just use the original audio CDs, as you won't gain any advantages in shelf-space consumption or the convenience of access. Think instead about compact disks and cartridge media such as the Iomega Jaz or the even higher-capacity and less-expensive Orb drive from Castlewood Systems ([www.castlewoodsystems.com/](http://www.castlewoodsystems.com/)). There are also a handful of tape storage units on the market that offer far faster access than the pokey data tapes of yore, and which can be accessed as a normal drive volume on the system. While tape storage often offers the lowest cost-per-megabyte ratio of any medium, their relatively low market saturation compared to other removable media means that MP3s stored on tapes may be harder to transfer to other machines if necessary.

Perhaps the best single solution for long term, affordable MP3 storage is the recordable CD-ROM. These units are available in either write-once (CD-R) or write-many (CD-RW) alternatives, and buy you 640 MB or more of disk real estate for a buck or two (not counting the cost of the drive mechanism itself). Recording to CD is covered in Chapter 5.

## *Equipment Considerations*

This book assumes that you already have basic audio working on your machine, but since you may be about to start listening to real music through your computer on a regular basis, you may want to invest a little time and/or money into making sure you're getting the highest quality possible from your system.

### *Sound Cards*

Most computers sold today ship with sound cards preinstalled and configured, but the cards in many discount and mid-range machines leave something to be desired from an audiophile perspective. Because the sound card is a fundamental bridge between what goes on in software and what ends up in your ears, it pays to make sure you've got a good one—overall quality can be drastically affected by the quality of your audio device.

While any recommendations on specific cards would quickly go out of date, keep in mind that better sound cards will also include digital signal processing chips (DSPs), which can take some of the computing load off your machine's CPU.

Good DSPs can go a long way toward minimizing the performance impact of MP3 playback, particularly if you've got a slower machine. If you intend to do any serious recording and/or mixing, look for a "full-duplex" card, which is capable of handling incoming and outgoing audio streams simultaneously. If you want to interface your card with external devices such as high-end digital-to-audio Converters (DACs) or home theater systems, look for a card with digital output jacks as well as the standard analog outputs.

When hooking up your speakers, most sound cards offer two analog outputs: one that is pre-amplified and is suitable for connecting directly to a set of unpowered computer speakers or headphones, and another that is unamplified, and is thus suitable for being connected to an external amplifier, such as a powered satellite/subwoofer system or your home stereo. This latter jack, labeled simply "Line," may offer a cleaner signal due to the fact that it hasn't been run through the sound card's (usually) suboptimal amplifier. If you're playing MP3 through a high-quality system, use the Line jack for output: the difference in quality could be noticeable.

There are many more nuances and subtleties between the many sound card flavors available out there. If you care about such things, it pays to do some research online before plunking down your money for a sound card.

## Speakers

Since speakers are the last link in the chain, and because they vary so radically, speaker choice is crucial to ensuring the best MP3 experience. If you can afford it, avoid the cheapo plastic speakers sold in many discount stores and catalogs at all costs. If possible, look into a powered satellite/subwoofer combination, which will let you keep small, compact (but relatively high-quality) midrange tweeters on or near your desk, while putting the bulky subwoofer beneath it, or elsewhere out of the way. Because it's very difficult to locate bass frequencies in space, the placement of the subwoofer is nearly insignificant. It's a different story with midrange and high frequencies, however, and speaker placement can be very significant for your satellite speakers. Note that if you have a four-speaker setup, you will probably require additional operating system support to get sound out of all four speakers. Under Windows, for example, you'll need DirectSound3D installed. This support should come along with drivers supplied with the sound card, though you may need to download it separately from Microsoft.



---

Note that merely getting four speakers up and running does not automatically give you true spatialized sound with all forms of audio. MP3 provides only standard stereo separation though the MPEG-4 standard will support multi-channel audio (see Chapter 9).

---

### Speaker placement

While it can be difficult to place speakers optimally in cramped or cluttered quarters, the same rules that apply to home stereo speaker placement apply to your computer's sound system. Divide a room into imaginary thirds and place your speakers at the dividing lines. Keep speakers at least six inches away from walls to avoid bass response anomalies,\* and angle your speakers slightly toward the center of the room, as illustrated in Figure 3-19. Experiment with this "toe-in" to achieve the best quality positioning of instruments in space. Keeping speakers away from side walls will help to avoid soundstage distortion, and it's important to make sure you leave adequate clear space between the speakers. At the very least, try to place your speakers on either side of your sitting position so that stereo effects will be perceptible.

Of course, in a computer-at-desk situation, you may be sitting closer to the speakers than the optimal listening position shown in Figure 3-19. In this case, it may not make sense to spread your speakers out to the 1/3 points of the room. Instead, aim for the more general rule of creating an equilateral triangle between the listening position and the two speakers.

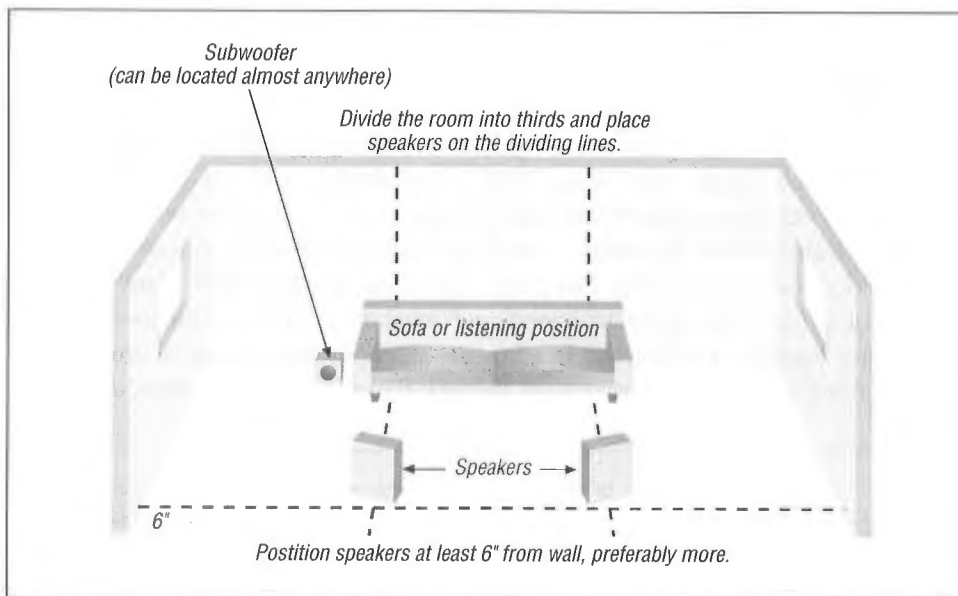


Figure 3-19. Optimal speaker placement

\* However, note that some speakers are designed to be placed against a rear wall, and these, of course, should be. Consult your audio dealer if unsure.

### *Connect your computer to your stereo*

If your computer is near your home stereo, you can easily direct your computer's audio output into your stereo. You'll need a special adapter and a length of RCA cable long enough to bridge the distance. The adapter should have a 1/8" male jack on one side to plug into your sound card's line out jack, and two female RCA jacks on the other. Connect your RCA cable into the adapter's female jacks, and run them into the Auxiliary input on your home stereo's amp or pre-amp (actually, any input on your stereo besides Phono will suffice, but Auxiliary is often the only input available). Turn your amp or pre-amp's input selector to Aux, and you're all set. If your sound card has digital outputs and you have an external DAC, DAT deck, MiniDisk unit, or home theater system with digital inputs, go for the all-digital connection to bypass the DAC built into the sound card (in most cases, the DAC in your digital amplifier or other device will be of a higher quality than the one built into the sound card).

### *Headphones*

If you work in an office or other environment where you can't disturb others, or if you want audiophile sound without spending big bucks on computer speakers, it's possible to get high-quality headphones very affordably. While any set of headphones can be hooked up to your sound card's output, you may be amazed by the radical improvement in audio quality you can get by spending \$50 or more on a decent set of ear goggles. Go to your local audio store and ask to audition models in your price range. Take along some music with which you're well familiar, and listen closely for nuances that may be audible with one set of headphones that aren't with another. Consider comfort as well—some headphones (even some with tremendous audio quality) can chafe, or start to feel tight after many hours. If you've got some extra bread to spend and want to go for the highest possible quality, get yourself a dedicated headphone amplifier and a pair of true audiophile "phones." The sound you can get through these is absolutely breathtaking.



---

One thing to keep in mind about headphones if you're into bass-heavy music is that, while good headphones can deliver great bass, you'll miss that bodily experience of having your guts rumbled, as you would at a live show or with speakers that move a lot of air.

---

If you're new to auditioning either speakers or headphones, keep in mind that it's easy to be fooled by aspects of the sound that jump out at you. Speakers that overemphasize certain frequencies may sound more direct, punchier, or brighter than more subtle speakers. While these characteristics may sound impressive in an

audio showroom, they may grow fatiguing and unnatural-sounding after lots of listening time at home. Remember: A brighter light is not necessarily a better light, and a brighter speaker is not necessarily a better speaker. The ideal playback system does not add to the incoming signal in any way—it merely passes it on in the most neutral and natural way possible. While playback equipment *always* colors the sound stream in some way, look for equipment that minimizes the effect, particularly if you listen to music with lots of subtleties (rock fans may appreciate a more direct sound, for example).

There are, of course, other ways to get MP3 audio out of your home stereo than just hooking it up to your computer. Increasingly, MP3 audio is finding its way into dedicated, external players that can be connected to your home stereo just like you would connect a CD player or other component. In addition, home CD players capable of playing CDs that store MP3 files are beginning to appear. We'll cover that ground in Chapter 6, *Hardware, Portables, Home Stereos, and Kits*. Finally, you can always use your computer to burn regular audio CDs that will play in any CD player (though you'll lose MP3's compression advantages in the process). We'll cover that in Chapter 5.

# 4

## *Playlists, Tags, and Skins: MP3 Options*

### *In this chapter:*

- *MP3 Options and Considerations*
- *Equalization and Sound Quality*
- *ID3 Tags and Playlists: The Virtual Database*
- *Skins: Dressing Up MP3 Players*
- *Plug-ins: Extending Your Reach*
- *Listening to MP3 Streams*
- *Performance Considerations*

The stock \$35 SoundBlaster-compatible card, driving a pair of \$25 plastic desktop speakers, has become a de facto standard throughout the '90s. But as computing power increases in accordance with Moore's law,\* media production and playback functionality takes a more central role in the consumer computer. With the advent of audio cards like the SoundBlaster Live! and the increasing popularity of satellite/subwoofer systems and computer-to-stereo connections, quality file-based audio becomes a realistic possibility for millions of users. MP3's natural home may be the wildlands of the Internet, but its reach is quickly being extended into other domains as well. As homes embrace the "convergence" model, computers, stereos, and televisions are blending together into all-in-one, networked infotainment centers, and stereo component manufacturers are introducing hardware-based MP3 players. Sales of portable units such as the Diamond Rio and Creative NOMAD are skyrocketing, and more and more cars are being fitted with MP3 playback units as well. As a result of all this, it becomes more important than ever to think beyond the simple task of encoding and playing MP3 files. Increasingly, people need to move large amounts of data between their computers and their playback devices, optimize the quality of their MP3 collections, maintain a well-organized filing system or MP3 database, make sure all of their files are tagged with useful meta-data for future reference, parse meaningful playlists out of collections extending into the gigabytes, and extend the reach of their MP3 players to handle unusual situations.

---

\* See the glossary for definition and more information.

## MP3 Options and Considerations

Once you've mastered the basics, you'll want to expand your horizons a bit and start checking out the many advanced capabilities, cosmetics, playlist generation techniques, ID3 tagging tools, plug-ins, and other toys available. There's a lot out there, and we'll only have space to touch on a handful of examples of useful "peripheral" software here. Do some searching through your favorite MP3 site or software library and you'll find hundreds of tools not discussed in this book.

Before we get to the goodies, however, we'll discuss some of the issues affecting the quality of MPEG audio. Because the compression format discards some data, MP3 already stands on shaky ground from a fidelity standpoint. That's not to say the quality of MP3 stinks, as some critics claim—but there are some things you can do to optimize the quality of your MP3s during encoding and during playback.

On the lighter side of things, MP3 players can often be dressed up in "skins"—small collections of bitmap images that sit on top of your player to give it a customized appearance. You can download skins from the Internet or create your own, although the process is admittedly a bit tricky. Don't worry—we'll show you how to create your own skins from start to finish.

Two of the most important "peripheral" technologies you'll meet in this chapter are ID3 tags and playlists. We discussed ID3 tags in a technical vein in Chapter 2, *How MP3 Works: Inside the Codec*; they're the extra space in MP3 files that let you store "meta data" about a file, including the artist, album, and track names, as well as genre, year, and personal comments. An up-and-coming modification of the ID3 specification, called ID3v2, is much more powerful, and lets you store a nearly unlimited amount of additional information (up to 256 MB). We'll check out the many ways in which ID3 tags can be created or edited, either directly through your MP3 player or via separate software.

One of the most enjoyable aspects of building a large MP3 collection, and one of the things that makes it so different from building a tape or CD collection, is the fact that you can mix and match songs into a customized sequence at a moment's notice. These personalized "albums" are called playlists, and consist of simple text files referencing the locations of tracks scattered across your system. Playlists can be created by dragging tracks one at a time into an MP3 player's playlist editor, by dragging entire directory structures onto your MP3 player, by trawling your disk with scripts, by running command-line queries, or by scanning through your collection with a database-like solution like Helium.

Ensuring that your files have solid ID3 tag data (accomplished either while encoding or after downloading) can be very important later on when you want to start creating playlists based on query criteria, such as "Create a playlist of all songs on all of my disk volumes by either Neil Sedaka or The Carpenters written between 1971 and 1975 that don't have the word 'schmaltzy' in the comment field."



The functionality of many MP3 players can be extended by installing plug-ins, much as you would for Netscape Navigator or Adobe Photoshop. A bewildering variety of plug-ins is available, which will let you do everything from controlling your MP3 player via infrared remote control to applying sound effects to displaying wild visualizations of the music as it plays. We'll take a look at a few of the more popular plug-ins later in this chapter as well.

An increasingly popular way to listen to MP3s is not to save them to disk or to encode them, but to listen to them in real time as they're broadcast out over the Internet. While the process of running your own Internet broadcast is covered in Chapter 8, *Webcasting and Servers: Internet Distribution*, we'll show you how to tune into the two main types of streamed MP3 (broadcast and on-demand) in this chapter.

Finally, you'll want to make sure your system is well-tuned for MP3 encoding and playback, so we'll talk about the hardware requirements—what kind of processor you need, sound and video (yes, video) card interactions, disk speed issues, and the like. We'll also show you how to benchmark MP3 players and encoders in case you have an older machine and want to make sure you're not dragging down the rest of your system's performance.

## *Equalization and Sound Quality*

While true audiophiles may never accept lossy compression formats as a home standard just for the convenience factor, most of us *do* place a very high priority on the many conveniences offered by the MP3 format. So we're left with a question: How do we make sure we're getting the most from our MP3 collections?

### *Can Quality Be Measured?*

Before even *thinking* about evaluating standards of quality, it's critical to understand that "this way lies madness." Audiophiles have wrestled for decades to achieve "objective" measurements that would fairly represent the quality of a recording. And while there are many objective measurements that can be made for any given signal, it's important to understand that the mathematics of audio measurement and the subjective experience of quality are two different animals. Furthermore, the mechanics of lossy compression (discussed in Chapter 2) more or less nullify the validity of just about any objective criteria. In other words, the only tests that matter for MP3 audio are subjective tests, i.e., real-world listening tests. Your ears don't lie. Even so, it's possible that the file you encode to your own standards today could end up being played on much better equipment tomorrow, so it pays to shoot for quality a little above your own thresholds.

The most important thing to keep in mind when testing for MP3 quality is that your computer probably isn't the best place to do it. How's that? Unless you've invested a fair sum in high-end computer audio equipment, chances are that your home stereo sounds a lot better than your computer. And if you're planning to keep your MP3 collection around for a long time, your home stereo may very well be the ultimate destination for your MP3 files. Because your sound card and computer speakers may mask a whole lot of subtlety, it's important to eliminate those components from the testing chain so that you can tell which limitations are introduced by your hardware and which are introduced by the MP3 encoding itself. In other words, you want to isolate the encoding as much as possible and give it an optimum environment in which to be tested.

### *Conducting listening tests*

Since "CD quality" is our benchmark here, you want to get your MP3s onto an audio CD. To run this test, you'll need access either to a CD burner or to a dedicated MP3 playback device. Find a couple of songs with which you're well-familiar, preferably ones that include a wide range of instruments, including those from the brass and strings families. It's often easier to detect limitations in the MP3 codec with non-electronic music, and the brass and string families do an especially good job of putting the codec to the test. However, it's also important that you choose music you enjoy personally. To be well rounded, you might want to choose one each of your favorite jazz, classical, and rock tracks, for example.



---

If you have a portable or home stereo MP3 player, just patch it into your stereo as described in Chapter 3, *Getting and Playing MP3 Files*, and load it up with encodings created as described in the next section. The same principles described in this section apply whether you're working from a test CD or from a dedicated playback device.

---

Next, you'll need to save a reference track for each song, meaning you should rip uncompressed versions of each track to your hard drive (typically in WAV format, though Mac OS users will want to use AIFF format). See Chapter 5, *Ripping and Encoding: Creating MP3 Files*, for details on ripping tracks directly from CD without encoding them. For the actual compression, you'll want to use the highest quality encoder you can find (again, see Chapter 5),\* and encode each track a

---

\* The test described here can also be used to test various encoders against one another. The only difference is that you'll want to encode the reference track at the *same* bitrate, but with different encoders. In fact, you may want to encode at various bitrates with each encoder in your test, so you can determine whether a given encoder excels at low bitrate encodings but falls down at higher bitrates, for example.

number of times, at different bitrates. As you encode, give each file a descriptive name, including its bitrate. For example, you might use the following scheme:

```
Improv.Spacious.64.mp3
Improv.Spacious.128.mp3
Improv.Spacious.160.mp3
Improv.Spacious.192.mp3
Improv.Spacious.256.mp3
```

### *Automating Test Encodings*

If the process of encoding the same track over and over again sounds tedious, it is. Here's a script you can use under Unix/Linux or BeOS to automate the process, assuming you have the LAME encoder somewhere in your path. This should be very easy to modify to work with other encoders:

```
#!/bin/sh
FileName="$1"

# Start a loop and encode once at each bitrate specified
for i in 64 96 128 160 192 256; do
    echo Creating "$FileName".$i.mp3
    lame -ms -b$i "$FileName".wav "$FileName".$i.mp3
    echo
done
```

If you're on a DOS/Windows machine, here's a batch file that will achieve the same effect:

```
Echo off
rem example
set filename=%1

if not exist %filename%.wav goto error
rem The following must go on a single line -
rem We had to wrap it to two lines for this book
for %%B in (64 96 128 160 192 256) do lame -ms -b%%B %filename%.wav
%filename%-%B.mp3
goto exit
:error
echo File %filename%.wav does not exist
goto exit
:exit
```

Whether you're running the the bash or the DOS version, save the script as a file in your system's path with a filename like "TestEnc" and then run:

```
TestEnc filename
```

assuming your input file is called filename.wav. Of course, these scripts assume that LAME can be found in your system path as well.

Once you've encoded each sample at a variety of bitrates, you'll need to translate them *back* to WAV format and burn the resulting tracks to an audio CD (although your burning software may take care of this conversion for you; see Chapter 5 for details). Because the filenames won't be present on the audio CD, take care to print out or write down the track numbers and the corresponding bitrates so you'll have a reference sheet later on. Once your audio CD is complete, find the highest quality stereo you can (yours or a friend's), grab your reference sheet, dim the lights, relax, and listen through the CD. While you may want to skip back and forth between the reference tracks (which will sound identical to the original CD audio tracks) and the encoded tracks later on, you should first listen to the disc all the way through. Don't be too quick to flip back and forth between tracks, as that will only lead to schizophrenia and ear fatigue. The idea is to internalize the "flavor" of the tracks relative to one another. Evenings may produce the best listening environment, as you'll likely get less ambient noise and less sensory competition from vision (assuming some darkness). If you've been listening to loud music or noises during the day, save the test for another day—your ears will already be fatigued. Self-run audio tests using your own hearing as a "benchmark" should always attempt to isolate variables and incorporate a best-case scenario, against which everything else should be measured.



If your home stereo or entertainment system includes an on-board digital-to-analog converter which offers settings for surround sound or for various room effects (concert hall, small room, etc.), it's important to disable these options and make sure your system is set on the defaults. If you use an equalizer or other audio "enhancing" device, disable it or run it flat (unless it's a "set-and-forget" processor carefully tuned to neutralize a particular room). You're testing your encodings here, not your equipment. Artificial interferences in signal flow will compromise the integrity of the test.

As you listen, take note of the most subtle sounds in the reference track, and try to determine whether those sounds are still present in the various encodings. Can you hear the clarinetist taking that breath just before his solo? The sound of the guitarist's hands gliding over her strings? What about the sense of space (sound-stage expansiveness)? Can you pinpoint the location of various instruments in the recording studio as accurately with files that have gone through the MP3 process as you can with the reference track? What about dynamic range? Are you detecting certain subtleties of the high and low ranges being chopped off or smeared out? Perhaps most importantly, do the MP3 encodings have a sense of "presence?" Can you close your eyes and imagine that the performance is taking place there in your living room? Does a given encoding make the performer sound sort of

“mechanical” or “electronic?” Does the performance seem to be more or less intense with any of the encodings? Do any of them do a better job of making you want to tap your foot or sing along or dance, or do any of them make you feel edgy, or bored? Do the MP3 encodings sound “hollow” or “swishy” to you? Chances are, you’ll be amazed to discover just how much quality has been lost in the 128 kbps and lower encodings.

The bitrate at which you literally cannot tell the difference between the reference track and the MP3 encoding is the bitrate at which you should start encoding your MP3 tracks, provided you’re willing to part with the extra disk space this will consume (of course, if you’ll be burning a lot of tracks to writeable CD for permanent storage, this may be less of an issue). To be really safe, and to make sure your tracks will sound great no matter what playback system they end up on one day, encode at a bitrate *higher* than the one at which you couldn’t tell a difference.

If you don’t have access to a CD burner, then your next best bet is to connect your sound card’s line-out jack to one of your home stereo’s inputs, and try similar tests. However, keep in mind that unless you use a fully digital transfer system (i.e., using digital coax or optical cable from the sound card to your digital receiver or external DAC), you’ll be hearing the limitations of your sound card’s digital-to-analog converter.

### *Digital to Analog (and Back)*

As you know, computers store and manipulate information—including audio information—digitally. But your ears are analog devices, and so are speakers.\* Somewhere in the process of getting information out of your computer and into your ears, the signal must be converted from digital to analog. The device that affects this conversion is called the *digital-to-analog converter*, or DAC, and its role is crucial. The quality of the DAC plays a huge part in the overall quality of the resulting audio pouring from your speakers or headphones.

In the case of most stock sound cards, the DAC is a part of the card’s chipset. However, some higher-end sound cards offer digital outputs, which can be hooked up directly to the digital input of a digitally equipped stereo receiver or pre-amplifier. If your receiver or amplifier doesn’t have digital inputs (they’ll be clearly marked), you can purchase an external DAC or use the DAC built into a DAT deck, Minidisc player, or DCC, which will then sit between your computer and your receiver/pre-amp. The quality of external DACs is often directly related to their cost; a very high quality DAC can cost \$1,000 or more, though you should be able to pick up a suitable model for a few hundred bucks.

---

\* Or, at least, most of them are; completely digital PCM speakers are beginning to appear on the market.

### Audio Test Checklist

To set up a test like this in a hurry (actually, it's best not to hurry when doing things like this), remember these simple steps:

1. Choose a reference track from your collection with which you're familiar. Look for a reference track with a wide variety of instruments and not too much density. A wall of guitar distortion is not a good choice, though you'll get mixed results with straight electric guitar under various encoders. Well-recorded acoustic guitar, vocals, strings, and brass will show you more limitations in the MP3 codec than hard rock. Rip the reference track to uncompressed WAV or AIFF format.
2. Using the best encoder you can find, encode the reference track at a variety of bitrates, using default settings in the encoder. Alternatively, test various encoders at the same bitrate.
3. Decode the encodings back to WAV or AIFF (see Chapter 5 for details).
4. Burn the reference track and the decoded encodings to an audio CD. Keep a list on paper of all the tracks going onto the CD.
5. Find the best stereo you can, relax, close your eyes, and just enjoy the music. Don't start analyzing carefully or paying special attention to subtleties and nuances until after you've relaxed into the music.

In any case, by turning the job of conversion over to a dedicated unit, you can bypass the typically poor quality built-in DAC on most sound cards, and get a far better resulting sound stream out of your system.\* When purchasing a sound card, make sure it has digital outputs in addition to the standard line out. If you're serious about quality, do some research online to find out which cards have the best on-board DACs. If you're really, really serious, consider a high-end outboard (external) DAC.

On the flipside, the reverse process—*analog-to-digital conversion* (ADC) may also come into play, depending on your needs. If you plan to encode music from analog sources such as tapes, LPs, or microphones, you'll also appreciate the extra input quality you'll get by making sure your source signal is stored in the computer after having passed through a high-quality ADC (either onboard or outboard), rather than the one built into your average sound card.

---

\* Of course, the rest of your system should be of high quality before you go spending money on a DAC. Putting an external DAC in a system with poor speakers, for instance, probably won't result in noticeably better sound.

## Equalization

Most GUI MP3 players include some sort of equalization mechanism, typically accessed by clicking a button on the player's interface labeled "EQ." Bringing up the equalizer usually displays a row of sliders (most often 10, though there could theoretically be any number of them). So, what exactly is equalization, and what does it get you? The simplest way to conceptualize equalization is to think of the bass and treble controls on your home stereo, and imagine that you had a separate knob for each of many much finer slices of the frequency spectrum, affording you much tighter control over the resulting output signal.

### *Why equalize?*

The point of equalization, of course, is to make the resulting audio sound better. EQ can be applied to compensate either for poorly recorded audio, for limitations in your speaker quality or design, or for the speaker placement or acoustics in the listening room. The ultimate objective of equalization is to obtain a "flat" frequency response (hence the name "equalization")—one in which the signal does not sound "colored" by the signal processing chain, storage media, or playback equipment.\* Equalization can also be used to eliminate audio artifacts stored in the original recording. For example, you may be able to partially correct a recording with an annoying 60Hz hum, or to partially remove some of the scratchiness of encodings taken from LPs (though the latter can be pretty tricky, since scratchiness typically crosses many different frequencies, and you don't want to throw the baby out with the bath water).

Despite these advanced capabilities, many (if not most) people use equalizers not so much to correct for anomalies as to boost the frequencies they like and to diminish the ones they don't. For example, hip-hop fans may want to pump up the bass frequencies, while classical listeners may want to "sweeten" the strings.

Before you go tweaking your equalizer's sliders with wild abandon, consider the equalizer as something to be experimented with gently and systematically. Listen carefully to the resulting sound after each adjustment, and pay attention to how manipulating one sub-band affects your perception of others—diminishing one sub-band can cause others to sound more prominent. Note also that if you find you have to equalize heavily to compensate for room acoustics, poor speakers, or a poor sound card, you should consider addressing these problems directly. Equalization is a band-aid, not a panacea. Be aware of the potential for over-equalization, and of the limitations of EQ principles in general:

---

\* Note, however, that many recording studios use equalization to *add* color, not to remove it—a fact which drives some audiophiles nuts.

- Without a lot of expensive test equipment, you have no concrete goal to aim for—you're just tweaking until something in the sound "feels" better for one reason or another.
- EQ bands come in predefined widths and slopes, but your equalization needs don't—they're typically more "fuzzy" than the usual set of EQ bands allows for (it's kind of like trying to apply mascara with a hairbrush... and no mirror).
- Analog equalizers can introduce distortions, especially phase distortion. Digital equalizers can minimize some of these effects.

It's easy to confuse louder and brighter sound for better sound. Optimum quality is achieved when music sounds as natural as possible—equalize with an ear toward a natural sound atmosphere. Over-equalization can easily degrade sound quality. In other words, the answer to the question, "Why equalize?" is often "You shouldn't, unless the situation demands it." It's easier to ruin the sound than to improve it. Experiment with your EQ, but don't overdo it.

### *Artificial Audio Enhancers*

Every now and then, you'll come across a product that claims to improve the quality of computer audio dramatically. Typically, these products (QSound's iQ, [www.qsound.com](http://www.qsound.com), being an example) work by running a series of proprietary algorithms over the outbound audio stream and applying digital effects to the stream just before it heads for your sound card. Much like the sample effects that come with some sound card drivers, or the presets found in some home theater amplifiers, these products create the illusion that the sound stage extends out beyond the edge of your speakers, generating a 3D sound stage on the fly. These products can also take a monophonic sound stream, such as you'll typically get when listening to SHOUTcast or icecast streams, and "turn them into stereo."

Do these products work? Well, there's no question that the effects are quite dramatic, and you'll most certainly notice a difference over normal playback. But the fact is, these products are creating a signal that isn't present in the source by playing auditory tricks with channel shifting and phase relationships (bouncing sound waves against one another). What these products do is unnatural and unrealistic, and while they may make a cheap computer sound system sound more dramatic, they certainly won't bring you any closer to the artist's vision of how the music was intended to sound.

If you choose to experiment with products like iQ, keep in mind that the signal you'll be hearing is not "true." This may or may not matter to you, depending on your perspective. My position is that effects like these are best reserved for use with games, but should be disabled when listening to music.



### The Fletcher-Munson curve

While you should, of course, equalize as necessary to hit the sweet spot that works for a given situation, there's one commonly used configuration of which you may want to be aware: The classic "Fletcher-Munson" curve, which takes advantage of the fact that the ear is not as sensitive to low and high frequencies when music is being played at low volumes. Thus, boosting these frequencies somewhat above normal can help music to sound more natural at lower volumes (Figure 4-1).



Interestingly, the function of the loudness button on your receiver is similar; apply an Fletcher-Munson equalization curve to the music makes it sound more natural at lower volumes.

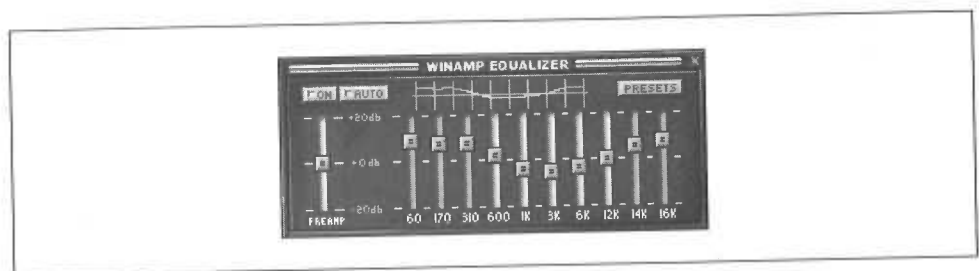


Figure 4-1. Fletcher-Munson curve, for use in low-volume situations

Note that many MP3 equalizers, such as WinAmp's, let you store equalization curves as presets, which can be loaded up again later without painstakingly re-creating them. Depending on your player, it may also be possible to store a prime equalization curve directly in the MP3 file itself, so you can optimize all of your songs' equalization curves independently. This capability is dependent on the fact that the ID3v2 specification allows for a huge array of meta-data storage capabilities in MP3 files; in other words, you must be using an ID3v2-compliant decoder capable of storing EQ presets. At this writing, this capability remained purely theoretical, though it should become a realistic possibility as the ID3v2 spec builds more momentum. See the ID3 section of this chapter for details.

### Working with EQ presets

Because different tracks may require different equalization curves, WinAmp lets you associate an EQ preset with any given track. Here's how to associate an EQ curve with a track:

1. Play the track normally, and press AUTO in the EQ interface.
2. Dial in your EQ curve.
3. Click the Presets button and navigate to Save → Autoload Preset.
4. Allow WinAmp to associate the current track's filename with the current EQ curve.

After creating an association between a track and an EQ curve, the association is stored by WinAmp in a file named *winamp.q2* in the WinAmp installation directory (the file cannot be edited by hand). Any time AUTO is enabled in the equalizer and that track is played, the preferred EQ curve will snap into action.

### *Uncooking Bad Transfers*

Every so often, you may end up with an MP3 file that's been corrupted during transfer somehow, usually by being transferred in ASCII mode rather than binary. If you should encounter one of these, search the Internet for *uncook.exe*, *Uncook95*, or *OK Uncook*, all of which do essentially the same thing—repair the damage that was done to the file during transfer and make it playable again. Linux and BeOS users may want to keep a copy of *mp3asm* ([www.ozemail.com.au/~crn/mp3asm.html](http://www.ozemail.com.au/~crn/mp3asm.html)) around for the same purpose.

If you're using a recent browser and the web server you're accessing has been set up properly, this should never happen, though it's nice to know there are tools available to correct the situation if it does.



In most cases, the equalizers built into MP3 players will have no effect on the audio stream when playing audio CDs, since CD audio is often routed directly from the CD player to the sound card without moving through software first. In this case, the MP3 player is just used as a convenient interface onto the CD transport's control buttons. However, there are plug-ins available that will "steal" the CD audio signal and route it through the MP3 player (the NullSoft CD/Line Input plug-in for WinAmp, with the "sampling" option enabled, for example). See the "Plug-ins: Extending Your Reach" section in this chapter for details. Once the CD plug-in has been enabled, you should be able to use the equalizer and various visualizers to interact with CD audio.

## *ID3 Tags and Playlists: The Virtual Database*

ID3 tags and playlists are two of the most important extended topics for people who take their MP3 collections seriously. Taken separately, both technologies are important in their own right, but taken together, these two functions can complement one another in very powerful ways, giving you the means to sort, store, organize, name, and build highly customized virtual collections. Because these two technologies are so synergistic, and because many tools exist to help you organize both of them simultaneously, we'll treat both of them together in this section.

### *What Are ID3 Tags?*

Every MP3 file has the ability to store "meta-data" related to the track in the file itself, in the form of what are known as "ID3 tags." For example, a file's ID3 tags may store the song's artist, album, year, genre, and comments in ID3 tags. Many MP3 players have the ability to read ID3 data out of your files, and to display this information in the MP3 playback interface. Thus, giving your MP3 files descriptive names isn't the only—or even necessarily the best—way to identify the tracks in your collection.

Conveniently, ID3 tag information can either be included in the file at the time it's encoded or added in later. Most of the better encoders will provide a number of options to let you control whether and how ID3 tag data should be written to the file. While some MP3 players include the ability to edit, as well as to display ID3 tag information, you may want to download and install additional utilities that specialize in ID3 manipulation, for maximum control. We'll look at some examples of ID3 tag editors later on.

### *ID3v1 vs. ID3v2*

It's important to understand that there are two basic flavors of the ID3 specification, conveniently named ID3v1 and ID3v2. Of the two, ID3v2 is vastly superior, for three reasons:

- Unlike ID3v1, ID3v2 imposes no arbitrary limitations on the amount of storage space available for meta-data, which means there's plenty of space for storing images, complete lyrics, performance notes, equalization presets, et cetera.
- ID3v2 data is stored at the beginning of the file, rather than at the end. This means that ID3v2 is much better for use in streaming situations, where it's important that users be able to see and use the meta-data while the song is being played, rather than after the download or broadcast is complete.



As useful as it can be to have ID3v2 data appear at the beginning of the file, this state of affairs also presents a problem—especially for older players. A player built to the ISO spec should simply skip over the ID3v2 data that it doesn't understand, and move ahead to the valid MP3 data. But exactly how far should the player seek through the file for an MP3 header frame before giving up and deciding that it's not currently dealing with an MP3 file after all? Unfortunately, the ISO spec is vague on the matter, which leaves the door open to interpretation by developers. Some believe that a player should seek forever (after all, ID3v2 data can occupy as much as 256 MB of space), while others feel that a player should give up after a certain number of bytes and assume that they're not dealing with a valid MP3 file after all. The user could have innocently tried to play a giant AVI movie through his MP3 player, for instance, which would cause a player that seeks forever to basically hang. Since the spec is not clear on the matter, the debate may never be resolved. The best solution may be store this data separately, in a hypothetical, centralized MP3ID database, for example, although that's just one of the ideas being floated around.

- The ID3v2 spec is open-ended and flexible, meaning it can be extended in the future to accommodate as-yet-unseen needs. Users may even be able to develop their own custom solutions, given the availability of future ID3v2 tag editors and management solutions. For example, the built-in "Popularimeter" field could be used to let users rate all of their tracks on a scale of 1 to 100, and then create custom playlists based on a scale of favorites.

The number of possible applications for ID3v2 data is staggering. Right off the bat, it gives artists and labels a place to store copyright information, terms of use, and proof of ownership. There are even possibilities for cryptographic security being embedded in MP3 files, thanks to ID3v2. For the end user, ID3v2 means the ability to store full-size, color scans of album covers, complete lyrics (even synchronized lyrics, for use in Karaoke situations), multiple URLs for further information, music reviews, extended information on the recording session, and so on.



While users of ID3v1 players sometimes discovered that the spec, with its 32-character limitation, didn't even give them enough space to store very long artist or album titles (such as Stereolab's album "Cobra and Phases Group Play Voltage in the Milky Night"), the ID3v2 spec allows for a maximum of 256 *megabytes* of additional tag storage space, meaning users won't bump into arbitrarily imposed limitations. Of course, if you fill a file with 256 MB of ID3 data, don't expect many users to want to download it from your site.

Because a surprising number of older players are *not* ID3v2 aware, and because of the problem mentioned earlier regarding the length of time an older player should seek for MP3 data before giving up, problems do arise. For example, if you use MusicMatch Jukebox to embed an image of an album cover into an MP3 file using Jukebox's ID3v2 tag editor, you'll find that the file will no longer play in the command-line mpg123 player, while Xmms may be able to play the file but won't show any ID3 tags for it.

Despite these issues, ID3v2 is a spec with huge potential. One of the most popular applications to take a stab at the ID3v2 space is MusicMatch JukeBox. At this writing, ID3v2 support in MusicMatch was limited to a minority subset of the total frames available (although these were the most significant ones). In addition, MusicMatch was taking a proprietary approach to storing some categories, including tempo, preference, and mood, which some feel could have been better handled had MusicMatch gone with standard ID3v2 frames. Finally, MusicMatch's implementation of the Content Type frame was still incomplete, and resembled ID3v1 more than ID3v2. Still, somebody had to be first on the dance floor, and the company's commitment to driving the standard forward should be appreciated. However, there's a lot of work remaining to bring their product into full compliance with the specification.

So how do you tell which ID3 spec is in use on any given file? There's no way to get this information just by looking at a file; the best way is to open the file in a tag editor capable of reading both versions of the spec, and looking to see where in the editor's fields information shows up. Any older MP3 file can be "upgraded" simply by opening it in an ID3v2-capable player or editor and adding extended information to it. In fact, some utilities will automatically migrate all ID3v1 data to equivalent ID3v2 fields. Opening an ID3v2-enhanced MP3 file with an older player will probably result in you seeing nothing in the ID3 info fields. The file itself, however, should still play properly. If it doesn't, you should consider changing or upgrading your MP3 player software.

Having accurate information in your files' ID3 fields is important for several reasons. If you ever edit or screw up your file's names, you can always just open the file in an ID3-capable player, read the author and song title, and rename the file accordingly. Some tools will even do this for you, automatically and programmatically, so you can assign ideal file names to every MP3 track on your system. ID3 tags are displayed in many playlists, so that you can display only song names, only artists, or only the comments. ID3 tags are commonly used by hardware MP3 players such as portable "Walkman"-type devices and home MP3 units. And some programs and utilities will let you treat the ID3 info in your files collectively, as if your entire collection was a huge, multi-field database.



For more information on the ID3 spec and its evolution, see the ID3 Standards Group web site at [www.id3.org](http://www.id3.org).

## What Are Playlists?

After all the talk about how the MP3 revolution represents a rebirth of the “custom cassette” revolution of the '70s and '80s, the arena of playlists is where you finally get to see that promise become a reality. In their simplest form, playlists are simply text files containing paths to MP3 files on your system—one line in the file for every track referenced. The beauty of playlists is that they can be constructed to reference arbitrary tracks, rather than being limited to entire directories. For example, you might have a playlist referencing only the tracks in your collection in the Free Improv genre, or all tracks by The Carpenters, or all tracks you're temporarily storing in a holding bin until you decide what to do with them. There are many ways to construct playlists. Most users will create them by simply dragging tracks (or entire directories) out of the system's file manager (Explorer/Finder/Tracker, etc.) and into an open playlist window, then saving the new playlist to the hard disk. However, playlists can also be constructed from the command line by way of scripts, or from database searches. As you'll soon see, real playlist flexibility comes into play when you use your files' array of ID3 tags to search on specific criteria, then save the resulting found set as a playlist file.

Because they're constructed with such simplicity, playlists are nearly universal; for the most part, a playlist created with any method will work in any MP3 player (on the same operating system). Of course, the exact format of the lines comprising a playlist depends on the path style on your operating system. For example:

```
DOS/Windows - C:\DATA\MP3\ALBUMS\CAN\DELAY\PNOOM.MP3
Mac OS - Main HD:Data:MP3:Albums:Can:Delay:Pnoom.mp3
Linux - /usr/local/home/shacker/data/MP3/Albums/Can/Delay/Pnoom.mp3
BeOS - /gorgonzola/home/data/MP3/Albums/Can/Delay/Pnoom.mp3
```

So while a playlist will work on any MP3 player on your operating system, it wouldn't on another OS. (This really isn't much of an issue though, since people organize their collections very differently and it thus makes little sense to share your playlists with others, unless you're trading around data CDs full of MP3 files). Even if you're a Linux or BeOS user storing your MP3 collection on a Windows partition and then accessing it from Linux or BeOS, the paths to the files on the Windows partition will still appear differently, so you won't be able to share playlists between different OS partitions.

It's worth noting that most users will seldom see the actual text of a playlist file. While you can open a playlist in any text editor for viewing or hand-editing, it's almost always easier to view and manage your playlists directly in your MP3 player's built-in playlist manager, since these typically give you drag-and-drop, shuffle, sort, total playback time, and other MP3-specific features you probably won't find in a text editor. Playlist manager/editors generally do not show you the full paths to your MP3 files, displaying only the actual track names. In many cases, playlist editors will read the ID3 tags out of each file in the list and display those (if found) rather than the filenames. As you'll see in Chapter 8, playlists may also contain the URLs to MP3 files on the Internet, so that you can stream known favorites "out there" directly to your MP3 player.



Because most MP3 players can accommodate multiple audio file formats, including WAV, RealAudio, AAC, and others, there's no reason you should restrict your playlists to referencing MP3 files alone. If you've been collecting multiple audio file formats on your system and your player supports them all, feel free to create playlists referencing multiple file formats. If such a playlist does get fed through a player that doesn't understand some of the referenced formats, it should gracefully skip those files (assuming it's well-designed).

### Playlist formats

The form of playlist described above is known as an M3U list,\* and is named as *playlist.m3u*. In some cases, you may also encounter playlists with a *.pls* extension. These files are virtually the same, with a couple of internal differences. They begin with an identifying block marking the file as a playlist, precede each track with a numerical identifier, and close by listing the total number of tracks referenced. *.pls* files don't always reference the *full* path to each track, so if you move a playlist to another directory, disk volume, or hard drive on the same system, it won't work. *.pls* files are often dished up by streaming MP3 SHOUTcast/icecast servers, since the *.pls* format allows for the distribution of meta-data.



When creating playlists on your own machine, you'll almost always want to stick with .M3U for maximum compatibility with a wide variety of MP3 players. If you do need to create a *.pls* file, you'll find the option in the "Save Files as Type" picklist in the file panel when you access your playlist's Save function.

\* For "MPEG Layer 3 URL"; the URLs contained by these files can be either local files on the user's system or remote files "out there" on the Internet. The MIME type for M3U files is *audio/x-mpegurl*.

Finally, there are a few applications that create their own custom playlist formats. A good example of this is the Apollo MP3 player for Windows (*apollo.mp3-2000.com*), which combines an MP3 player and sophisticated playlist manager into a single application. While Apollo offers a number of cool playlist features that others do not (such as ID3v2 support and the ability to create lists from arbitrary ID3 criteria), its playlists are only useful with Apollo itself.

### *Loading and manipulating playlists*

Playlists can be loaded into your MP3 player in several ways, depending on the platform and the specific player. In most cases, the easiest way to load a playlist is to simply double-click its icon. If it's been properly associated with your MP3 player, it will launch the player with that list preloaded. Playlists can usually be dragged onto your MP3 players' interface, or loaded from the File → Load button in the player's playlist editor. If you're a command-line junkie, you can usually load a playlist by typing:

```
<path to player> <path to playlist>
```



WinAmp users can right-click a playlist file in Explorer and choose Enque from the Context menu. If a playlist is already playing, this one will start playing when the current one has finished. You can queue up a pile of playlists to play sequentially in this way.

Most playlist editors include a number of functions designed to let you sort the order of tracks by song name, artist, length, and other criteria, though "Randomize" seems to be far and away the most popular means of "organizing" playlist tracks. If you're using Windows and are dropping files into a playlist editor, note that a peculiarity of Explorer could end up frustrating your attempts to get them to appear in same order you selected them in. However, if you select the last file first in Explorer, hold down the Shift key, and then highlight the first file; you can drag the whole block into your playlist editor and have the selections appear in the right order. In other words, the order in which you begin your selection in Explorer is the reverse of the order in which the selection is passed to other applications. You gotta love it.

Most playlist editors include a set of buttons (or a pull-down menu) separate from that of the MP3 player itself. WinAmp's playlist editor, for instance, includes buttons that will let you add entire directories or URLs to the playlist window, remove all selected files or all "dead" files (i.e., references to files which no longer exist at the specified location), sort your playlist by a variety of criteria, or save your playlist. To access the ID3 info for any file in WinAmp, select it in your playlist and tap Alt+3.





You can also generate HTML playlists directly from WinAmp's playlist editor. With the editor open, click Misc Options and select Generate HTML Playlist (or just tap Ctrl+Alt+G). The current list will be sent directly to your default browser. You can then pull down File → Save in the browser to store the resulting HTML document for future reference.

### ID3 and Playlist Editors

There are seemingly infinite numbers of ways to create interfaces onto ID3 tag data (Figure 4-2), and we can't hope to cover a fraction of them in this book, but here are a few examples of the creative ways ID3 tags can be accessed, displayed, used, and edited, in various operating systems.

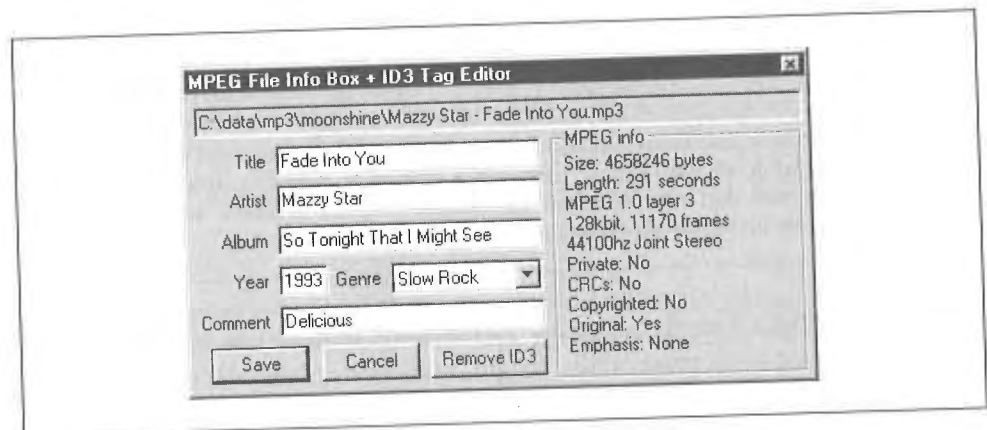


Figure 4-2. A WinAmp's built-in ID3 tag editor

### Windows

MP3 collections are becoming larger, and large projects lend themselves well to database-like solutions. Since the array of all tags in all of your MP3 files, considered together, pretty much amounts to a database, the real trick is in building a good interface onto that database, letting you view and edit ID3 tag data, rename files based on existing ID3 tags, see the collection of all ID3 tags in a database-like manner, search that database on custom criteria, and generate playlists from the results. If you want to do all of these things, you have two choices. You can scan your favorite software library for individual tools that serve one or more of these functions (and probably get it all done for free, but create a lot of busy work for yourself), or you can grab an all-in-one tool like Helium ([www.citymob.rit.se/belium/](http://www.citymob.rit.se/belium/)). While most of the all-purpose tools will probably cost you a few bucks,

they'll more than pay for themselves in the time they'll save you in the long run. Since, at this writing, Helium is one of the best all-purpose ID3/database/playlist tools available, and because it covers the general principles involved in performing all of these tasks, we'll focus on Helium in this section. Note, however, that there will likely be other, perhaps better, tools available for your operating system by the time you read this.



Figure 4-3. The main Helium window

The main Helium window plays host to any number of what are known as "Browsers," where a browser is simply a table of MP3 files: a sort of fancy playlist (Figure 4-3). To create a new browser window:

1. Double-click on the Helium background. Point the empty browser to an MP3 directory or collection of files by clicking the Add Files or Add Dir buttons on the left.
2. All MP3 files in that directory and its subdirectories will appear in a table, along with any ID3 tags stored in those files. If the referenced files don't include MP3 tags and you want to give them some, select those files in the browser and pull down Tools → Edit Tags (or tap Ctrl+E).

3. An ID3v2 tag editor will appear for the first selected file, offering tons of fields, picklists, and text boxes into which you can type or paste lyrics, select genres, and so on (Figure 4-4).
4. When finished editing a file's tags, click Save, then Next to move to the next file in your selection.

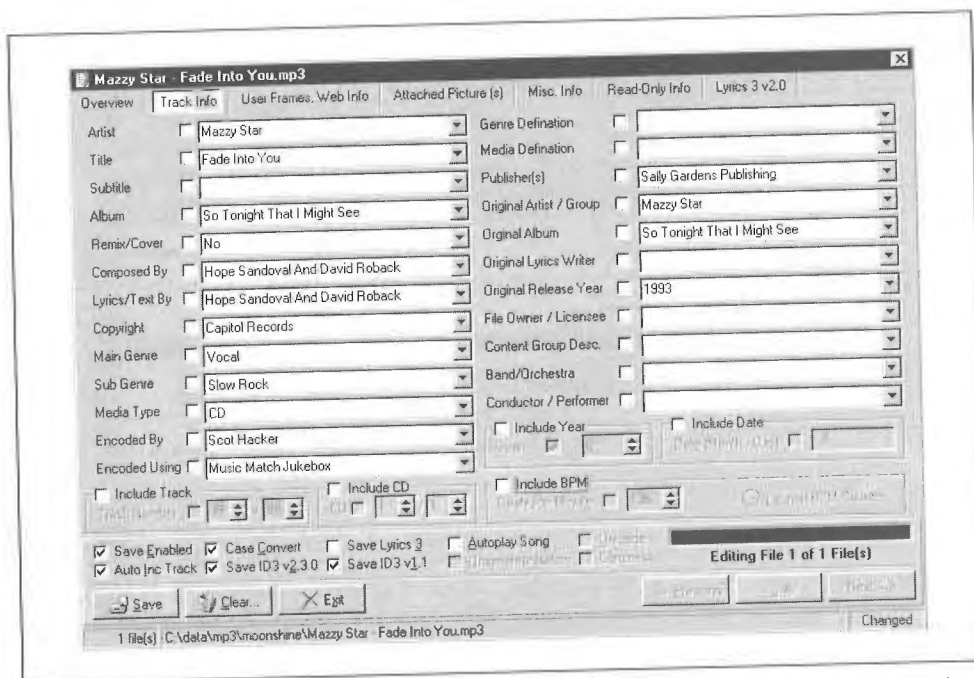


Figure 4-4. Helium's tag editor offers the majority of ID3v2 frames; buttons along the top let you navigate amongst major classes of allowable ID3v2 data

Helium is also capable of renaming files singly or in batches, based on the contents of their ID3 tags (though the version we tested did not offer special character or blank space substitution, which we feel should be a mandatory feature). To access the renaming function, select a file or files and pull down Tools → Rename Files. As with other ID3 renaming tools covered in this section, you have partial control over the format of the filenames that will be created, via a template function accessed from Helium's Output Editor. Conveniently, Helium lets you store as many naming templates as you like. A couple are built in, but you'll probably want to create your own as well. Click the New button, and then the Tag Help button to see a list of all allowable field strings. Artist/Group is represented by the symbol %00, Title by %01, Album by %05, and so on. So if you've got a file called *file1.mp3* and its ID3 tags specify that this is "Blue Jay Way" from the Beatles' *Magical Mystery Tour*, you could set up a template of the form:

```
%00%05%01
```

Renaming your file with this template selected will result in a file called:

```
BeatlesMagical Mystery TourBlue Jay Way.mp3
```

That, however, is probably not what you want. To add separators between the fields, enter them in the Prefix and Suffix fields.

Helium is capable of exporting lists in two ways. Before you do, however, you'll probably want to use some of Helium's filtering and sorting capabilities—here's where the real power of a database begins to come to light. Let's say you've got hundreds of folders full of MP3 files scattered around on your hard drive, and you want to make a playlist of all songs with their Genre tag set to "Booty Bass." If you haven't already assigned Genres to your tracks, use the Tag Editor to do so. You'll also need to make sure the Genre column is showing in the Browser table—you can enable this from Helium's preferences panel. Now pull down Edit → Search, type in "booty," make sure the "Copy matches to new window" checkbox is selected and click Search. All songs in the main browser window that are tagged as Booty Bass songs will appear in a new Browser. Now all you have to do is pull down File → Save Playlist, select either the *.M3U* or *.pls* format, and save. Bingo— instant custom playlist.

In addition to playlists, Helium is capable of generating HTML or text documents from its browsers, so you can keep documents on your desktop that list your entire collection of MP3 files, or any custom subset thereof. Select a set of files or create a new browser containing customized search results, and click the List button. The output format of Helium's list generator can be completely customized by specifying HTML or other header data, and then inserting ID3 tag symbols into the Main List section of the List Generator.

Because of the inherent power of database-driven tools, you'll find that applications like Helium become increasingly indispensable as your MP3 collection grows larger.

### *One-click master list*

How would you like to be able to double-click a single icon on your Desktop, have it scan your system for MP3 files, generate a master playlist, and launch that playlist into WinAmp or another MP3 player? A batch file (that's DOS-ese for shell script) like the following one can make this easy. To do this, you'll need to invoke the ancient masters of your early DOS training.

Generating a playlist from the DOS command line (i.e., DOS prompt) is a matter of constructing a command that will find all files ending in *.mp3* in a specified directory and all of its subdirectories, that will not spew out all kinds of extra data you don't need, and that won't list directory names themselves. Here's what you want:

```
dir c:\data\mp3\*.mp3 /s/b/a-d > c:\data\playlists\thislist.m3u
```

If you scatter MP3 files all over your system (which wouldn't be very good house-keeping), you can change the initial path to simply *C:\*. In case you're curious about those switches, */s* tells *dir* to recurse down through subdirectories. */b* tells *dir* to output only a simple list of pathnames, without file sizes, or foreshortened filenames. */a* tells *dir* to handle special attributes, as specified in its subsequent arguments. *d* is one of the attributes */a* can handle, and since we precede it with a - instead of :, we're telling *dir* to ignore directory names themselves (directory names in playlists will simply be ignored by most players, but they're still messy). The *>* symbol is a redirect, and tells DOS to dump the output of the command to a new file. Of course you'll need to substitute in the path to your collection of MP3 files, as well as the preferred path to and name of your generated playlist file.

If you want to do this sort of thing in a batch file, so you can simply double-click an icon on your desktop to have the whole thing run automatically, you'll want to create something like this:

```
@echo off
echo.
echo Creating playlist from d:\data\mp3 ...
:: Remember to edit the paths below to match
:: those on your machine! The second line is optional,
:: and shows how you can have this batch file scan through
:: multiple drives and directories and append the results
:: to the same playlist file. Note that if any of your directory
:: or filenames are longer than 8.3 you'll need to surround
:: them in quotes so DOS doesn't get confused.
dir d:\data\mp3\*.mp3 /s/b/a-d > "d:\data\playlists\alltracks.m3u"
dir e:\hold\music\mp3\*.mp3 /s/b/a-d >> "d:\data\playlists\alltracks.m3u"
echo.
echo Launching the new list in WinAmp...
:: The next line can be the the full path to any MP3 player
:: on your system that accepts a playlist as a command line
:: argument. WinAmp handles this just fine.
start "C:\Program Files\Winamp\winamp.exe" "d:\data\playlists\alltracks.m3u"
exit
```

To test your new system, double-click the batch file in Explorer. Did it work? Good. Unfortunately, you will have noticed that a DOS shell is launched automatically when you run the batch file, and doesn't close itself when you close WinAmp. The trick to fixing this annoying behavior is to click the small Properties icon in the DOS shell's menu bar and check the "Close on exit" checkbox (you may also want to select "Run: minimized" to keep the DOS window from flashing briefly on screen). Close the Properties panel, and a *.PIF* file will be created in the same directory as the batch file, appearing as a standard Shortcut file. Drag this shortcut to your Desktop or wherever, and from now on you can just click that to launch your script. The DOS shell will close as soon as WinAmp launches.

### Do it with Perl

Search the Internet and you'll find tons of homebrew playlist solutions created by ordinary people, using free tools. One of the most common and useful such solutions is the Perl-based playlist creator, such as the one created by Patrick Hearon and available at [www.owl.net.rice.edu/~patrickh/playlist.html](http://www.owl.net.rice.edu/~patrickh/playlist.html). Point the script at a directory tree and it will "walk" through its contents, taking note of any MP3 files it finds in the path. When finished, it will generate either a plain text *.M3U* playlist file, suitable for use with any MP3 player, or a collection of HTML files, one in each directory, each listing the contents of that directory. Because each file automatically links to the others, you can browse through your MP3 collection easily, or post the HTML documents on a web site for others to peruse.



Perl junkies may also want to search CPAN ([www.perl.com/CPAN/](http://www.perl.com/CPAN/)), a.k.a. the Comprehensive Perl Archive Network, for all sorts of useful Perl modules. At this writing, MP3.com was in the process of contributing some of their own modules into the CPAN database, including modules to give Perl built-in ID3v2 and MPEG Layer 3 info support (a less complete module already exists).

Because Perl is available for nearly every platform known to humankind, these scripts can be easily tweaked to work with your platform (though all you'll have to change are the magic cookies pointing to the Perl interpreter, and the path style used on your system).

### Mac OS

Most of the popular MP3 players for the Mac include built-in ID3 tag editors, and usage is straightforward, as you might imagine. In MACAST, for example, make sure the playlist is showing, select a track, and click the Info button. Fill in or edit the fields as desired, then click Save to save your changes directly into the file, as shown in Figure 4-5. At this writing, MACAST supports only ID3v1 tags, though ID3v2 support may be available by the time you read this.\*



Do not attempt to edit ID3 tags in a currently playing track. MACAST will issue a warning and offer to move to the next track in order to make changes to the current track, but in our experience this froze the Mac.

\* In fact, a Lyrics field was available in the ID3 tag panel in the version of MACAST we looked at, and lyrics are officially a part of the ID3v2 spec. However, ID3v2 offers a much broader array of options than just lyrics, so we consider this to be only a partial implementation of ID3v2 support.

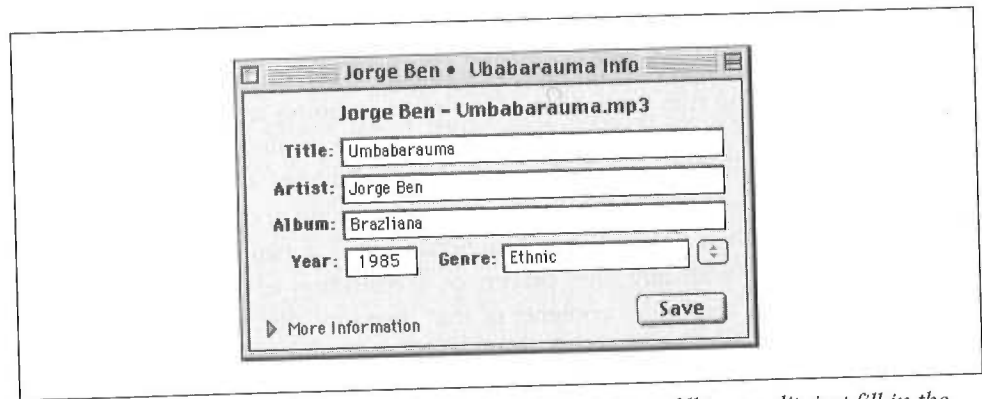


Figure 4-5. A MACAST window's Info button produces fields to fill in or edit; just fill in the fields and close the MATE window to save your changes

More powerful than the built-in tag editors is a piece of freeware called mp3tool, which will scan a specified folder for MPEG files and let you edit ID3 tags in any files it finds along the way. While mp3tool won't create playlists from the files it finds, it offer very straightforward tag-editing capabilities. Search your favorite Mac OS software library for mp3tool.

Since Mac OS doesn't have a command-line environment, it's not possible to create scripts that perform similar functions to the DOS batch file above or the Unix shell script below. However, if you're familiar with AppleScript, it should not be difficult to store references to your MP3 files in a commercial database (such as FileMaker Pro) or spreadsheet and extract the results of database searches to a plain text file. Alternatively, you can use the Finder's Find panel to scan your system for all MP3 files, then drag files in the resulting list onto the Desktop or into a folder. From there, you can drag files directly into your MP3 player. However, because you can't drag files directly out of the Find panel and into a player, this process essentially means moving your MP3 files around, which is probably not what you want to do. However, you *can* drag files out of the Find panel and onto the MP3 player application icon. In the end, you're probably better off using the MP3 finding tools built into the players themselves.

### Linux

The power of Unix-based operating systems is often at the command line, and ID3 handling is no exception. One of the most popular tools available for handling ID3 tags under Linux is id3ren ([tscnet.com/pages/badcrc/apps/id3ren/](http://tscnet.com/pages/badcrc/apps/id3ren/)), a multi-purpose tool that can not only read and write ID3 tags, but can also rename files based on the ID3 tags those files contain. While id3ren is most popular under Linux, note that BeOS and Windows ports of id3ren do exist, and can be found at your favorite software library. The real power of a tool like this is in its ability to be embedded in shell scripts and run over batches of files at once.

To see the tags currently attached to `filename.mp3`, type

```
id3ren -showtag filename.mp3
```

To rename a file with the default (built-in) naming template, just type:

```
id3ren filename.mp3
```

If the file's Artist tag was "Beatles" and the Title tag was "Taxman," `filename.mp3` will be instantly renamed to `[Beatles].[Taxman].mp3`. Don't ask me where that goofy default format comes from, but fortunately, you can easily change this by means of the `-template` argument. Templates in `id3ren` are a means of defining the naming structure of the files to be renamed, where the following identifiers are available.

Tag	Identifier
%a	Artist name
%c	Comment
%s	Song name
%t	Album title
%y	Year
%g	Genre

So, for example, if your file's Artist tag is set to "Beatles" and the Title tag is set to "Taxman," you can have the file renamed as *Beatles--Taxman.mp3* with the following command:

```
id3ren -template="%a--%s.mp3" filename.mp3
```



If you always want to use the same template, and don't want to have to specify it every time, create a text file called `.id3renrc` and place it in your home directory (see `id3ren`'s documentation for more on allowable locations). In this file, specify the same template parameters, but leave out the quotes. `id3ren` will now always use your preferred template, unless you use the `-nocfg` flag in your command. Windows users will need to replace the % symbols with \$ in the template examples here.

The simplest way to add to or change the tags associated with a given file is to type:

```
id3ren -tag filename.mp3
```

You'll be prompted to enter a string for each of the possible tags. Alternatively, you can add multiple tags at once with a single command, like this:

```
id3ren -tag -song "Mink_Shmink" -artist "Eartha_Kitt" filename.mp3
```



You'll then only be prompted for any tags you don't enter in the command (including `-comment`, `-genre`, and `-year`). If you don't want to be prompted for other tags, add the `-edit` argument. If all you care about are the Artist and Title tags, and don't want to be prompted for others, use the `-quick` flag. If you don't want the file to be renamed, and only want to edit tags, use `-tagonly`. To simply remove all tags, use `id3ren -striptag filename.mp3`. You get the idea. There are many more options available, but they all follow this basic model—just use `id3ren -help` to see them all.

The real power of `id3ren`, of course, comes into play when dealing with batches of files at once. In any of the example commands, wildcards or variable names can be used in place of `filename.mp3`. Thus, you can easily run through an entire directory of files at once. Since a directory might very well represent an entire album, you'll probably want to specify the artist and album names in the initial command, so you don't have to type them in over and over again. For example, let's say you just ripped and encoded all of *Magical Mystery Tour*, but the MP3 files you've created have been given names like *Track1.mp3*, *Track2.mp3*, and so on. The following command will give the files appropriate names, add the Artist and Title tags, and prompt you only for the song titles, skipping the year and comments:

```
id3ren -tag -artist "Beatles" -album "Magical Mystery Tour" -nogenre \
-nocomment -noyear -template="%a-%s.mp3" *.mp3
```

`id3ren` will prompt you for the name of each song in the current directory, write the song, album, and artist tags to each file, and rename each song as *Beatles-Taxman.mp3* and so on. If you don't mind being prompted for the genre tag with each song as well, you could use the `-quick` flag for a somewhat shorter command.

The following script can be used to sift through all of your MP3 files for just those files whose ID3 tags match criteria you specify. This sample script looks for matches on the Artist tag, though you can tweak it to search on the Genre tag, for example, "find me all songs that are filed under the Genre ('Gangsta'). Further adjustments could yield a script that searches on two criteria at once ("all Soul tracks from 1971"), or just about anything else you can imagine:

```
#!/bin/sh

# Get the artist name passed in from the command line
Artist="$*"

# Edit the next line to equal the full path to your main MP3 directory.
# To search your whole system, you could even make it " / ",
# But the more specific you are, the faster the script will be.
MP3Path=~/.mp3
```

```

# Set this to the location in which you want your playlist files
# to appear. This universal HOME variable should work on all
# Unix-based systems, though you'll probably want
# to customize it.
PlayListDir=${HOME}

# Now we'll get a list of all MP3 files living in the MP3 hierarchy.
# The list will be saved to a temp file for further parsing.
find "$MP3Path" -name *.mp3 > $PlayListDir/.id3-tmp

# Now we're going to open that temp file and examine each track
# it references, trying to determine whether its Artist tag matches
# the one the user specified on the command line. We have to do
# a little fancy work with grep and sed to get just the info we need
# and nothing else.
{
while read ThisFile; do
    ThisArtist=$(id3ren -showtag "$ThisFile" | grep "Artist\::" | sed s/Artist\::/)
    //)
    if [ $ThisArtist = "$Artist" ]; then
        echo "$ThisFile"
    fi
done

# Input is the temp file, output is the final playlist
} < $PlayListDir/.id3-tmp > $PlayListDir/$Artist.m3u

# Remove the temp file
rm $PlayListDir/.id3-tmp
exit

```

### BeOS

There are a number of tools for BeOS ID3 tag display and manipulation. All of the major MP3 players support normal ID3 display within their own interfaces, but BeOS also supports some ID3 manipulation not possible on other platforms. Because the Be file system supports "attributes" (data associated with a file that is not a part of the file itself), ID3 tags can be read out of MP3 files and displayed directly in the Tracker (the BeOS file manager), as shown in Figure 4-6. This provides an excellent database-like interface you can use to sort, query, and organize your MP3 tracks along user-selectable criteria.

One of the most useful tools you'll find out there is a collection of Tracker add-ons and applications by Jonas Sundström (see [www.be.com/beware/](http://www.be.com/beware/)). Install Tag2Attr and Attr2Tag in your `/boot/home/config/add-ons/Tracker` folder, then right-click any MP3 file or collection of MP3 files and select Tag2Attr. All of the selected files will be examined for embedded ID3 tags, and attributes will be written to the file system. You can now enable the MP3 attributes you want to view from the Tracker's Attributes menu, and edit your attributes directly in the Tracker. Because other operating systems don't recognize BeOS attributes, you'll want to make sure



Figure 4-6. BeOS lets you store ID3 tag info as filesystem "attributes," which can then be displayed and edited directly in the Tracker

these attributes are written back into the file itself if you edit any of them (and plan to make them available to users of other OSes); hence, Attr2Tag works exactly the same, but in the opposite direction.

While this pair is excellent for batch jobs, you may find you want more control. Sundström is also the author of TagWorld, which is also a Tracker add-on, but lets you examine and edit the collection of tags for each file individually (see Figure 4-7).

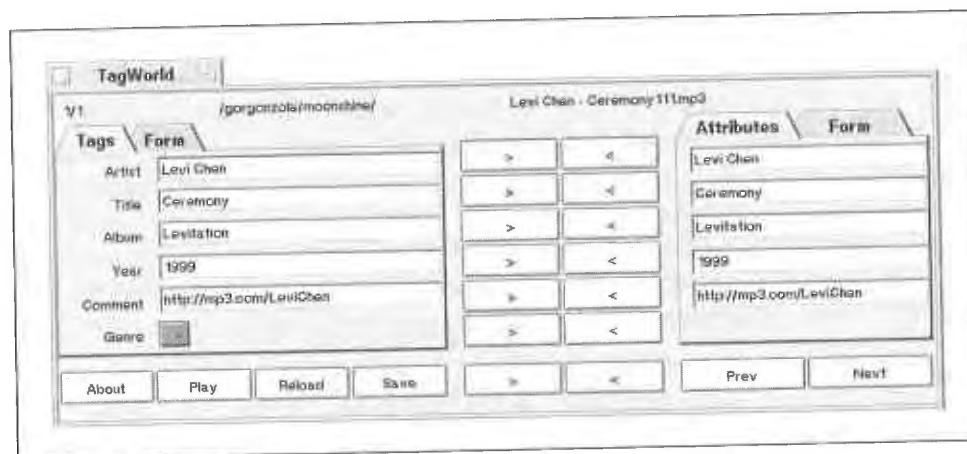


Figure 4-7. TagWorld lets you copy ID3 tags to attributes and back again

In order to query your BeOS system on ID3 data, you'll need to tell the filesystem to maintain an index of the ID3 attributes. Sundström also provides MP3 AI to help you create and manage the necessary indexes, as well as MP3 Flashlight, which

makes finding MP3 files even easier than it is from the system's Find panel. Taken together, Sundström's collection of tools exploits native BeOS features to gain functionality similar to that found in Windows with tools like Helium (covered earlier in this chapter). Many BeOS encoding tools will write ID3 tags to attributes as they're encoding, and can even create custom folder layouts based on GUI templates. More on that in Chapter 5.



If you prefer to work from the bash shell, or if you'd like to script some of these behaviors, download a copy of Ari Pernick's `id3attr` from [www.be.com/beware/](http://www.be.com/beware/). Once installed, you can copy ID3 tags to attributes by launching a Terminal and typing:

```
id3attr *
```

The author also keeps this alias in his `~/profile`:

```
idren='id3ren -template="%a - %s.mp3" -space=" "'
```

By using the `id3attr *` command followed by the custom `idren` command, he can "database" and give ideal names to a batch of freshly downloaded MP3 files in seconds.

Once your MP3 files have attributes attached, there's almost no limit to the ways you can create custom playlists. For example, if you want a playlist consisting of all the MP3 files in your collection written in 1971, just do a normal system query (pull down the Be menu and select "Find") on "MP3 → Year contains 1971." Select All in the query results window and drag the selection into an open playlist window. Save the new playlist, and you're done. Of course, you can extend the same principle to create playlists of any complexity. For example, you could create a playlist in three seconds flat consisting of all Neil Sedaka tunes that have a Comment field including the word "smarmy." Pretty cool, huh?

Of course, if you want to get more hands-on from the command line, you can also use bash shell scripts like the one shown in the Linux section earlier in this chapter, though the BeOS query function gives you the same functionality directly in the GUI, but without the need to edit configuration files.

## *Skins: Dressing Up MP3 Players*

As shown throughout this book, WinAmp and other MP3 players can be "costumed" with alternate interfaces called "skins." Because skins are merely collections of bitmap images with pre-specified names and sizes, skins are not tied specifically to WinAmp, or even to Windows. While WinAmp is by far the most common place to find WinAmp skins in use, you'll also find MP3 players for Mac OS, BeOS and Linux that wear WinAmp skins. In addition, there other MP3 players

out there with even more radical interfaces (such as Sonique and K-Jöfol) that also wear skins. Note, however, that these players don't wear WinAmp skins; see their sites for availability of skins for those players.

### *How to Get WinAmp Skins*

There are a number of large skin repositories on the Internet. The definitive collection is on WinAmp's own site, at [www.winamp.com/skins/](http://www.winamp.com/skins/). There you'll find nearly 3,000 downloadable skins made by WinAmp users like yourself, categorized into groups like "Computer," "Game," "Stereo," "Anime," and even "Ugly" (and believe me, some of them deserve their place in that category). You'll probably find the highest quality skins in the "Best Skins" category, a group that has been selected by WinAmp employees for their finesse and professional look. There are (at this writing) three other notable skins sites out there: [www.1001winampskins.com](http://www.1001winampskins.com) is a site run by WinAmp fans for other fans (not connected to NullSoft), while [www.skinz.org](http://www.skinz.org) and [www.customize.org](http://www.customize.org) offer skins for dozens of products, not just WinAmp (though their WinAmp skins collection is smaller than that of the other two sites).

Each skin collection arrives as a single zip file, and becomes accessible to WinAmp simply by living in the skins directory specified in the WinAmp preferences (which by default is in `C:\PROGRAM FILES\WINAMP\SKINS`). If you have an older version of WinAmp, you'll need to unzip each collection into a subdirectory of the skins directory. For example, a skin archive called "Evo" might need to be decompressed in `C:\PROGRAM FILES\WINAMP\SKINS\EVO`. However, if you're using WinAmp 2.04 or higher, you don't need to unzip the archive at all, since WinAmp is smart enough to unarchive the files it needs on the fly (there is no noticeable performance penalty for doing this).



If you do choose to decompress your skins, note that many skin authors carelessly zip up their collections without also zipping the parent directory, which means you can easily overwrite skin components of the same name if you don't manually create an appropriate subdirectory first. If you use WinZip, a good way to do this is to right-click the zip archive and select "Extract To..." from the Context menu.

Once installed, skins can be activated by right-clicking anywhere in WinAmp's interface and choosing Skin Selector from the context menu (tapping Alt+S will also bring up the Skin Selector). A panel will appear listing all installed skins. Single-clicking an entry will cause WinAmp's interface to instantly adopt the new look. After donning a new skin, try enabling the WinAmp playlist and equalizer

functions. Depending on how the author created the skin, these modules may or may not also take on the theme of the new skin. If skin components for these modules were not provided, they'll appear with the default WinAmp look and feel.

Please note that some skins are distributed as shareware, rather than as freeware. You should find a file called *readme.txt* in each skin's directory. If the skin is shareware, it will say so in this file. If you like the skin and decide to keep it, pay the author his or her due.



If you have multiple operating systems installed on your computer, there's no need to store the same skins collections multiple times. For example, since both BeOS and Linux can read Windows partitions, you may want to store all of your skins on your Windows partition and just create a symlink in your BeOS or Linux MP3 player directory called "skins" pointing to the location of your Windows skins directory. Of course you'll need to make sure your Windows partition is mounted automatically at boot time if you want this technique to work transparently.

### *WinAmp Easter Eggs*

Like most heavily evolved software, WinAmp has a few "Easter Eggs" tucked away—non-obvious "features" that one might only stumble upon by accident... or if one read about them in a book. Try these:

- Right-click in WinAmp's interface and choose "NullSoft WinAmp" to bring up the About panel.
- Click the WinAmp tab to see the animated WinAmp character.
- With that animation running, hold down Ctrl+Alt+Shift and click on the copyright notice at the bottom of the panel. The character will be overlaid with a bitmap of WinAmp's original developer, Justin Frankel.

Older versions of WinAmp had additional Easter Eggs, though these seem to have disappeared after NullSoft was acquired by AOL.

### *Creating Your Own Skins*

If you really want to call yourself a WinAmp pro, using other people's skins won't cut it—you've got to create a few of your own. While creating original skins can become tedious, time-consuming work—especially when you get into customizing regions and colors—the payoff can make the effort worth it. The only software

you need is a good image editor, such as Adobe Photoshop, Paint Shop Pro, the GIMP (Linux and BeOS only), or something similar. You can even create skins in Microsoft Paint, though you'll probably find Paint underpowered for the job.

Next, you'll need to understand the purpose of each of the *.bmp* files in a skins directory. While you can use any skin as a starting point, the definitive collection of templates can be downloaded as *Base.<version>.zip* from [www.winamp.com/skins](http://www.winamp.com/skins). You'll also find a skin-creation tutorial at that site. Unpack *base.zip* and you'll find three type of files inside: *.bmp* files, which are the actual graphics that will comprise your skin's interface; *.cur* files, which are standard Windows cursor files to be activated when the user's mouse hovers over different parts of the interface (ignored when the skin is used by another operating system); and *.txt* files, which can be customized to specify the colors of non-graphical elements and transparency behaviors.



You do not need to edit—or even include—a version of every single file in this directory. The behavior of elements for any files you don't include will be supplied by WinAmp itself. In other words, the default WinAmp interface is always present behind your creation.

### *Skin graphics*

Open the various *.bmp* files in your image editor, and the purpose of each will become immediately obvious—you'll see fragments of familiar interface elements in each one, while each file's purpose is also declared by its file name (Figure 4-8). For example, *volume.bmp* displays a column of colored horizontal glowing strips, representing the colors of the volume slider as you move it right and left. *Titlebar.bmp* likewise shows all possible incarnations of the WinAmp titlebar (e.g., as it appears when maximized, when in zoom mode, and with and without the visualizer enabled). All you have to do is edit these graphics to your liking, taking care to preserve their dimensions and division lines precisely. Because many of the elements, such as the fonts in *text.bmp*, may appear very, very small, you'll probably want to get familiar with your image editor's zoom (magnifying glass) function.

Because just creating new bitmaps for every possible interface element in WinAmp can be a huge job, most skin creators call it a day after finishing the graphics portion. If you really feel like gettin' jiggy wid' it though, you'll want to dig into the *.cur* and *.txt* collections as well.

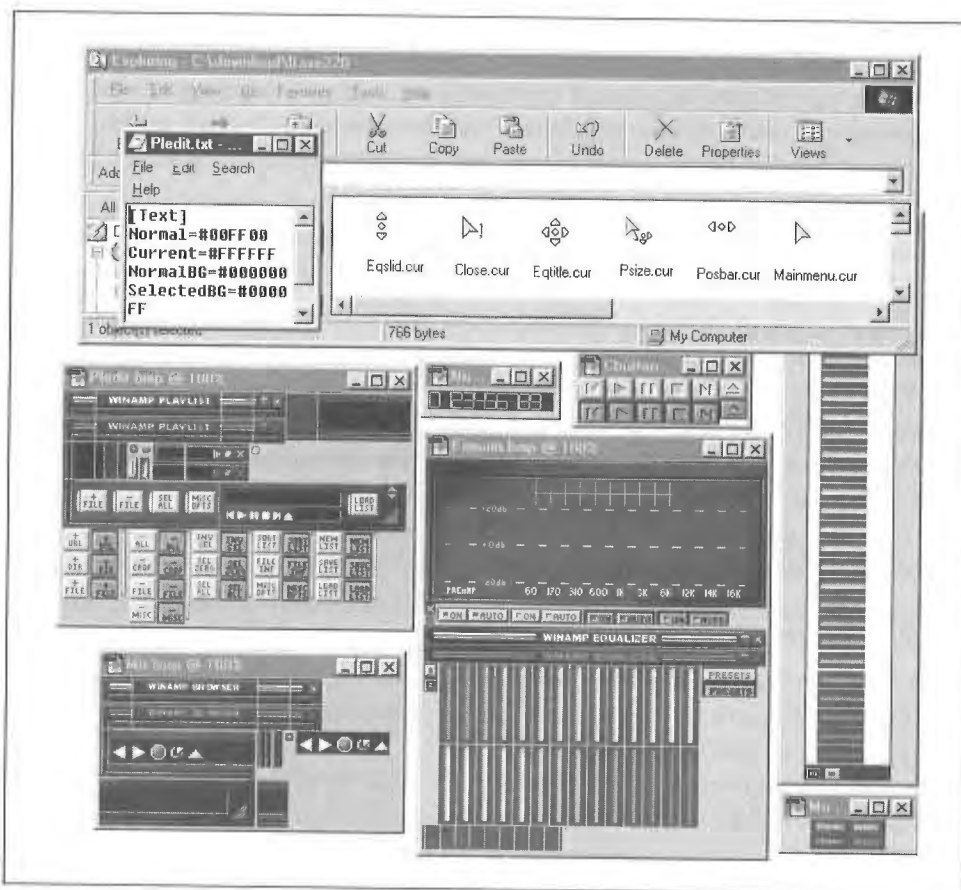


Figure 4-8. Bitmaps, base configuration files, and Windows cursors, all with specific filenames that must not change

### Cursors and control files

The *.cur* files used by WinAmp are standard Windows cursor files, and can be generated in any number of Windows shareware programs designed for the purpose. You may want to look into the excellent Microangelo ([www.impactsoft.com](http://www.impactsoft.com)) for some serious cursor customization action. You should be able to determine the purpose of each cursor file by its filename; for example, *eqslid.cur* is the cursor that goes into effect when the mouse is using the equalizer's sliders, while *volbar.cur* takes hold when adjusting WinAmp's volume.

### Manipulating fonts and colors

The most technical aspect of skin creation (and again, this is totally optional) may be the configuration of specific aspects of WinAmp appearance and behavior from the text-based configuration files *region.txt*, *viscolor.txt*, and *pledit.txt*.



The easiest to edit of these files is *pledit.txt*, which lets you control the fonts and font colors used in the playlist panel. All values are expressed in hexadecimal, a format that will be familiar to most web developers. If you need a good reference of hexadecimal color values, search the web for “netscape colors” or “hexadecimal colors.” You’ll find both web pages and applications that will help you look these up easily. The format of *pledit.txt* is simply:

```
[Text]
Normal=#00FF00
Current=#FFFFFF
NormalBG=#000000
SelectedBG=#000080
Font=Arial
```

Where `Normal` is the font color of most song entries, `Current` is the color of the selected track, `NormalBG` is the color of the playlist background, and `SelectedBG` is the background color for the selected track. If you specify fonts here, keep in mind that your users may not have this font on their system. If the specified font is not found, the default font will be used.

### *Manipulating the visualizer*

*viscolor.txt* lets you control the colors being used by the WinAmp visualization module. Unlike *pledit.txt*, colors here are expressed as RGB triplets (one value each for the red, green, and blue color values). You can use Windows’ built-in color manager (right-click on the Desktop, choose Properties, select the Appearance tab, and click Colors → Other) to determine RGB triplets. There are 24 lines in this file, each of which corresponds to an audio level threshold. Experimenting with this file is probably the best way to get a feel for which entries take effect at which audio thresholds. If you want to keep track of your changes, you can succeed each line with a comment:

```
24,33,41, // This corresponds to mid-volume signals
```

### *Manipulating skin regions*

Perhaps one of the most interesting interface controls you have as a skin developer is afforded by *region.txt*, which lets you use masks to “carve out” regions of WinAmp to be considered invisible. By using a system of X,Y coordinates, you can tell WinAmp only to draw specific sections of the interface, and to leave the rest transparent. The results can be dramatic. Working with *region.txt* is somewhat more complex than working with the other text files. The principle here is that you specify the shapes of polygons by telling WinAmp how many points each polygon has, followed by a series of X,Y coordinates for each polygon. Since WinAmp’s dimensions are 275×116 pixels, the X coordinate values can range from 0 to 274, and the Y values can range from 0 to 115. In order to determine the

exact coordinate points you need, you'll want to find the tool in your image editor that reports the exact coordinates beneath the mouse cursor. (In Photoshop, pull down Window → Show Info.) WinAmp simply draws all regions that fall within the polygons you've defined and leaves the rest invisible. Open *region.txt* in the base skin collection and you'll see lines like this:

```
;NumPoints=4  
;PointList=0,1, 275,1, 275,14, 0,14
```

Any line beginning with a `;` is ignored, so to enable any series of sample coordinates, just uncomment those lines, save, and load the skin to see the effect. Of course, you'll probably want to define more than one polygon, but for ease of development, just work on perfecting one polygon at a time (comment out your perfected polygon and start a new one). Once all polygons have been perfected, you'll need to conjoin them all into a single line, like this:

```
; NumPoints = 4, 4 ; In other words, define two squares  
; PointList = 0,1, 275,1, 275,14, 0,14, 3,15, 272,15, 272,113, 3,113  
; The space between the two sets makes the coordinate series easier to read.
```

See the comments in *region.txt* for further details. If you don't want any transparency in your skin, just delete *region.txt* or make sure all of its lines are commented out.



Mac users! MACAST supports a skin format that is incompatible with WinAmp skins. Fortunately, this also means that MACAST users are not limited to the size and shape constraints of WinAmp skins. If you do want to use WinAmp skins with MACAST, you can download a tool called SkinConverter from [www.solsticetechnologies.com/skin-converter/](http://www.solsticetechnologies.com/skin-converter/) that will do the job nicely. If you use SoundJam, you'll find a copy of SkinConverter in the SoundJam installation directory—just drag WinAmp skins onto the SkinConverter icon and they'll be transformed into SoundJam skins and placed automatically in SoundJam's skins folder, ready for use.

## Plug-ins: Extending Your Reach

Just as the capabilities of programs like Adobe Photoshop or Netscape Navigator can be extended through the addition of third-party extensions called “plug-ins,” so can many MP3 players. The beauty of the plug-in model is that the developers of the MP3 application get to have their product's capabilities extended indefinitely by the contributions of the larger developer community, and a cottage industry can be established for plug-in programmers. As a user, you'll find that your MP3 player may actually be far more capable than it originally appeared,

thanks to these additions. While skins let you customize the look and feel of your MP3 player, plug-ins let you extend its actual functionality. Unlike skins, however, which are generic enough to work across multiple operating systems and players, plug-ins must be written specifically to work with each MP3 player. While you must have programming experience to write plug-ins, anyone can use them. Some are free, some are shareware, and some are available only by paying a fee up front.



Some MP3 players, like WinAmp, are built modularly, around the very concept of plug-ins. Because much of these programs' functionality is derived from the presence of certain crucial plug-ins, you should *not* delete the plug-ins that come with the WinAmp distribution. You may very well end up disabling your player's ability to export any sound at all.

There are a seemingly infinite variety of third-party plug-ins available, but they all fall into a few overarching categories:

#### *Input plug-ins*

Allow your player to play audio file formats that are not built into the player itself, or to accept signals from external devices such as keyboards or stereo components.

#### *Output plug-ins*

Allow your MP3 player to export signals to other audio file formats.

#### *DSP/Effects plug-ins*

Let you apply any number of audio special effects, such as distortion, echo, reverb, and more.

#### *Visualization plug-ins*

Perhaps the most popular of the lot, let you "see" your music in displays far more sophisticated and psychedelic than the simple visualizers built into most MP3 players.

#### *Other plug-ins*

Let you do things that aren't easily categorized, such as handling incoming signals from universal remote control units.

The best place to find plug-ins for your MP3 player is usually at that MP3 player's web site. You'll also find a good cross-platform plug-in collection in the Software section at [www.mp3.com](http://www.mp3.com) and other popular MP3-related web sites. While the specifics of plug-in installation and configuration differ greatly from one player to another, you'll almost always need to install plug-ins in a subdirectory of the MP3 player's installation directory called "plug-ins" or "plugins." In most cases, you

should *not* create a new subdirectory for each downloaded plug-in under the *plugins* directory (yes, this is inconsistent with the skins installation model). WinAmp plug-ins are stored in *dynamically linked libraries*, or *.DLLs*. To access or activate a given plug-in, try right-clicking your M3 player's interface or looking for a Plug-ins menu. In WinAmp, just tap Ctrl+P to bring up the Preferences panel and expand the Plug-ins section of the hierarchy. To configure any individual plug-in, double-click its entry in this panel, as shown in Figure 4-9. Because most plug-ins are actual programs that run in conjunction with the MP3 player itself, most of them sport their own configuration interface.



Before starting to download and install third-party plug-ins, you should spend some time experimenting with the collection of pre-installed plug-ins, just to familiarize yourself with the huge number of options already at your disposal.

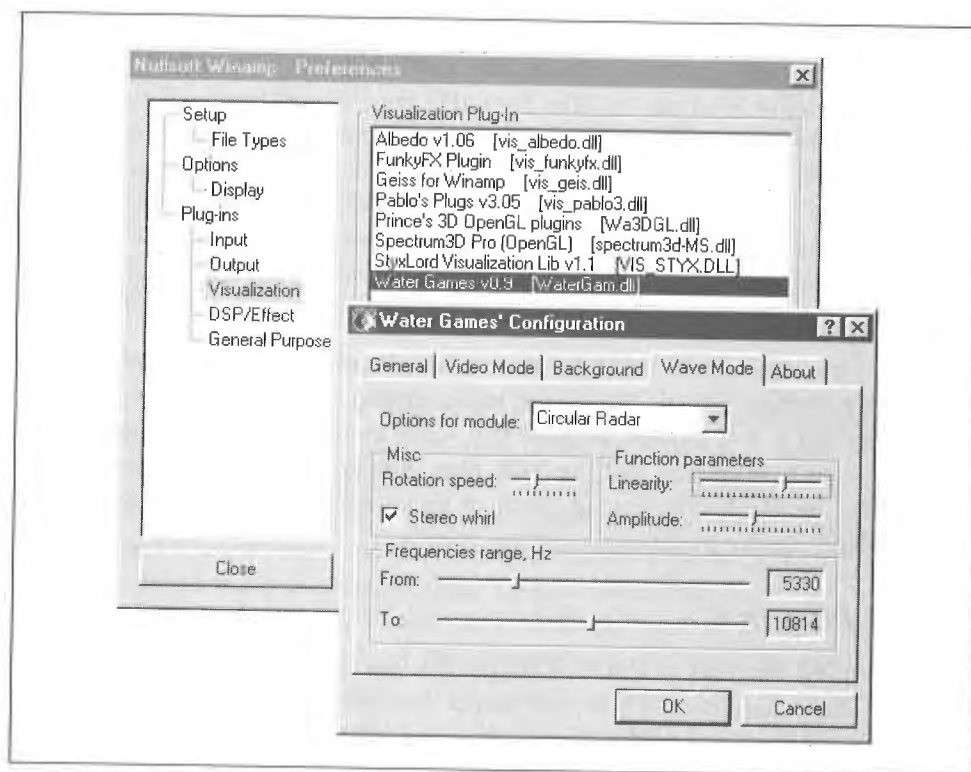


Figure 4-9. Double-click any entry in WinAmp's Plug-ins preferences panel to launch a configuration panel specific to that plug-in

Without question, the greatest number of plug-ins are available for WinAmp; at this writing, there were more than 140 WinAmp plug-ins available at [www.winamp.com/plugins/](http://www.winamp.com/plugins/), with more being added on a regular basis. That site, by the way, also includes a category called "Best Plug-ins," containing those deemed by WinAmp's staff to be of the highest quality or usefulness. You may as well start there. Let's look at a few of the most popular WinAmp plug-ins available, keeping in mind that plug-ins for other players and other operating systems perform similar functions but aren't available in such large number or variety.

### *Input Plug-ins*

This category includes plug-ins designed to let your MP3 player handle arbitrary file formats and signal types as input. You'll find input plug-ins that let WinAmp handle MIDI, MPEG, Waveform, and CD/Line input, for example. Double-click any of these for further options relating to the way these input streams are handled. Most important for MP3 players is, obviously, the MPEG input plug-in. Owners of older machines and crappy sound cards will find a number of options here that let you adjust the performance of the MPEG decoder to use fewer resources. All users will appreciate being able to customize the way song titles and artist names are displayed as they're read out of each file's ID3 tags. If you have a very slow or eternally busy hard drive, you may get better MP3 performance by setting the Full File Buffering size to a number higher than the default of 128 K (a buffer here is simply a space in program memory dedicated to holding audio data before it's processed; by using a larger buffer, you guard against the possibility of skipping audio due to the hard disk's read head not being able to gather data from disk fast enough to satisfy the player). Streaming Data Buffers will be covered in Chapter 8.

If you start looking around for third-party input plug-ins, you'll find enablers that let you play obscure formats like sound tracks ripped from video games, plug-ins that let you watch AVI movies, plug-ins that work in conjunction with radio tuner cards, listen to high-quality AAC or VQF files, Amiga SoundTracker files, or the ever-popular RealAudio file format. Each of these will come with its own documentation and unique preferences/settings panels.

### *Output Plug-ins*

Output plug-ins function as a sort of bridge that sits between the MP3 player and its output, controlling exactly how that output manifests. For general playback, the basic WaveOut plug-in (for WinAmp) is the only one you'll need, and you probably won't need to mess with its default settings unless you've got an older sound card (such as SoundBlaster 16), or need to perform some fancy footwork, like reversing the left and right stereo channels.

### *Decoding vs. Playback*

Throughout this book, you'll see the terms "decoder" and "player" used pretty much interchangeably. Technically speaking, anything that plays MP3 audio is a decoder, since it's basically running the codec in reverse, re-assembling intelligible audio from a pile of bits in accordance with the MPEG specification. However, the term "decoder" is sometimes used specifically to refer to any process that takes an MP3 bitstream and converts it back into an uncompressed PCM audio file such as WAV or AIFF. Whether the MP3 file is converted to audible signal or to an uncompressed file, the exact same processes are in use.

### *Converting MP3 to WAV files*

One of the most useful alternative outputs offered by the Output Plug-ins collection is the NullSoft DiskWriter, which lets you transform your outgoing MP3 signal into WAV files on your hard drive, rather than to audible signal. All you have to do to enable this function is double-click its entry, select an output directory on your system, and click Close in the WinAmp preferences panel. Playing MP3 files will now result in them being written to hard disk as WAV files. You will *not* hear output from WinAmp while this is happening, since only one output plug-in can be active at a time, and selecting the Disk Writer plug-in disables the WaveOut plug-in by definition. Of course, the resulting WAV files will be 10–12 times larger than the MP3 files they originate from. Remember to change back to WaveOut when you're done exporting to disk! You'll probably use this option a lot if you decide to burn audio CDs from your MP3 collection.

### *Use Microsoft Audio optimizations*

If you want your system to be able to play sounds from multiple sources at once, or if you want to launch multiple instances of WinAmp simultaneously, you'll need to take advantage of capabilities offered only in later versions of Windows, or installed with upgrades to the Windows MediaPlayer. The technology that allows this is called DirectSound, and WinAmp must be told specifically to take advantage of it. Select the DirectSound output plug-in rather than WaveOut if you want to mix multiple audio streams down into a single output, or if you just want to see what it's like to play a bunch of MP3s simultaneously. Keep in mind that Windows may still balk if you tax its audio system too hard, since the operating system is not inherently optimized for this kind of maximum audio throughput.

### *Cross-fading your files*

There aren't a ton of third-party output plug-ins floating around. The most popular is Justin Frankel's Crossfading output plug-in, which lets you start one MP3 file playing while the previous one is on its way out, with the first one's volume going

down while the next one comes up. In other words, a cross-fade plug-in can make your MP3 audio stream sound as if it were coming off a DJ's mixing bench. Cross-fade timing is controlled by adjusting the size of the buffer, measured in milliseconds (i.e., a 5,000 ms buffer will give you a 5-second cross-fade). Note that using a larger buffer will help prevent skips on slower machines, but consume more memory.



BeOS users: You can attach cross-fade settings to attributes of individual MP3 files, or to entire playlists at once, without needing a plug-in. Just click and hold on any currently playing track and select "Crossfade settings" from the context menu to adjust the settings for that file. If a playlist is currently being played, just select all tracks in the list and make your fade-out/fade-in changes, then click "Apply to Selection."

Linux users: If you use Xmms and have a multiple-CPU machine, try downloading the XAudio output plug-in from [www.xaudio.com](http://www.xaudio.com) and using that instead of Xmms's default output—you'll feel the difference in performance!

### *DSP/Effects Plug-ins*

This sophisticated (and often complex) breed of plug-in lets you apply digital signal processing and other effects to the audio signal. While you may enjoy simply tinkering with some of the amazing effects you can produce with these, many of them are geared specifically toward the more high-end user: radio stations, DJs, and sticklers for absolute quality and customization.

#### *Virtual DJ*

One of the most amazing DSP plug-ins available is Leif Claesson's PitchFork, which will literally let you treat your MP3 collection as if it were a pile of vinyl records, to be cued up, nudged, and synchronized. Pitchfork's controls are rather complex, and you'll definitely want to step through the included tutorial if you want to get the most out of the program. Of course, you'll need to be running at least two instances of WinAmp simultaneously, which means you'll need to be using the DirectSound output plug-in. If you really want to do it right, you'll want to have two sound cards installed in your computer, both of them running their line outs into an external mixing board. Needless to say, PitchFork's hardware requirements are a little beefier than your usual plug-in (but not overwhelming)—you'll need at least a Pentium 200 and 64 MB of RAM.



If you're serious about DJing with MP3 files and want something more than a plug-in, check out virtual dj at [www.virtualdj.com](http://www.virtualdj.com), a complete playlist organization/automation system that handles MP3 files (along with other formats). Linux users may want to check out DigitalDJ, at [www.nostatic.org/ddj/](http://www.nostatic.org/ddj/). BeOS disk jockeys use FinalScratch ([www.n2it.net/finalscratch/](http://www.n2it.net/finalscratch/)), which uses physical turntables and special records containing nothing but microsecond timing marks—the system depends on Be's extremely low audio hardware latencies to let DJs "scratch" MP3 files in SoundPlay from the helm of actual turntables.

### *Normalizing playback*

More down-to-earth is another of Claesson's creations, called AudioStocker. This gem provides a solution to the annoyance of having to reach for the volume control between tracks to compensate for the fact that MP3 files often demonstrate substantially different volume levels. Since most MP3 files are encoded digitally, without signal tweaking by the person doing the encoding, this artifact is due to the fact that different audio engineers and different recording labels tend to record albums at different levels to begin with, for whatever reason. You've probably noticed this effect before on your home stereo, but with MP3, where you're often jumping quickly between tracks extracted from different albums, the effect becomes much more noticeable.

The general term applied to the process of making multiple volume levels peak at similar thresholds is called *normalization*, and this is exactly what AudioStocker does. AudioStocker's controls consist of a set of push buttons that let you control the thresholds of the plug-in's intervention in the output signal. Tell the plug-in how much it's allowed to amplify a tune, how much frequency balance (equalization) can be applied, and a degree of audio compression (not to be confused with file compression) that can be applied. With a little tweaking, you should be able to get all of your tunes coming out of WinAmp at virtually the same perceived volume.

However, it's worth pointing out that this normalizing process is bound to result in some degradation of quality. Normalizing usually relies on the fact that the whole track can be examined in advance in order to establish peak levels. Normalizing on-the-fly, though, doesn't enjoy the luxury of a complete pass—it has to work by trimming the peak volumes and boosting the volume of quiet sections. The final output will therefore differ somewhat from the original by definition. While this is unavoidable in some situations, keep in mind that you'll always be better off doing your normalizing during the encoding process (from a quality standpoint).





In the future, it may become possible to store relative volume levels in a file's ID3v2 tags. Players would then be able to examine that data and adjust volume dynamically for each track. No known players were capable of working with ID3v2 tags in this way at this writing.

### *Other DSP/Effects*

Take a look around and you'll find a number of simpler DSP/Effects plug-ins available for many MP3 players, including modules that let you bend the pitch of the current signal up or down (without speeding up or slowing down the track), modules that let you apply standard effects like chorus, reverb, flange, or echo, and even modules that let you "stack" multiple plug-ins on top of one another, so that you can create cumulative effects.



While not technically a plug-in, WinAmp does offer a cool little output feature that will let you generate pure tones for testing purposes. Turn down your system's volume control so you don't accidentally hurt your ears or speakers, then tap Ctrl+L (for Open Location) in WinAmp and enter *tone://1000*. WinAmp will play a 1kHz tone until you press the Stop button. Of course, you can substitute any frequency you need. If you'd like to save the tone to disk as a WAV file, enable the DiskWriter plug-in in WinAmp's Output Plug-ins section.

### *Visualization Plug-ins*

This variety of plug-in is probably the most popular because they're the most fun. If you thought WinAmp's built-in visualization module was cool, you ain't seen nothin' yet. B-b-b-b-baby, you ain't seen nothin' yet! These modules analyze the signal flowing out of the player and into the system and run spectrum analyses, mapping changes in frequency and volume to rules that control a visual display. Because there are infinite numbers of ways to construct these rules, the appearance of these visualizers is virtually unlimited. Some of them are soothing, offering effects as subtle as drops of waters landing in a pool, while others can only be described as seizure-inducing, with wild psychedelic light shows emanating from every delta point. While some visualization plug-ins run in small or resizeable windows, others will optionally run full screen, which is perfect for use at parties, or late-night, low-light zombie sessions.

The screenshots shown in Figure 4-10 are of visualization plug-ins for WinAmp, but there are visualizers available for many different MP3 players, and for virtually all operating systems. These screenshots don't do justice to the experience of seeing these for yourself—you've just got to download a few and play with them. While all visualizers come with good default settings, most of them also include settings or preferences panels that let you customize the colors, responsiveness thresholds, and visualization forms. Dig around, turn the knobs, push the buttons, check and uncheck the checkboxes, and drag the sliders around. If you're a customize-a-holic, be prepared to set aside several hours of twiddle time.



Some visualization plug-ins are very CPU-intensive, and can have a noticeable effect on system performance. You probably won't want to run visualizers as a matter of course, reserving them for times when you're not trying to do other tasks in the background while listing to MP3 files. Not to mention the fact that many of these are wild enough to completely distract you from getting any real work done while they're running!

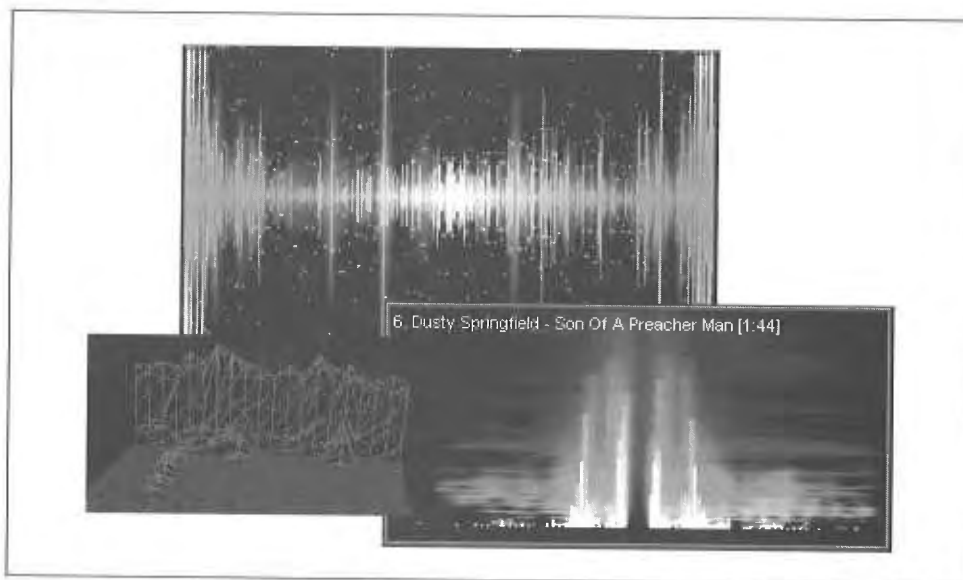


Figure 4-10. A few of the many visualization plug-ins available for WinAmp

### General Plug-ins

The category “general” is really just a nice way of saying “miscellaneous,” referring to plug-ins that do odd-jobs around the house. Did you know, for instance,

that you can control WinAmp with a standard, \$25 universal remote control? You'll need to purchase a separate infrared receiver to connect to your computer, and then install a corresponding IR plug-in to process the invisible signals it generates. This is the kind of connectivity that people are talking about when they refer to "convergence"—the computer becoming the center of the home infotainment system. See Ampapod ([www.v.nu/core/ampapod/](http://www.v.nu/core/ampapod/)) or Irman ([www.evation.com/irman/](http://www.evation.com/irman/)) for more information.

Other general plug-ins let you output your current playlist to an HTML file for use with SHOUTcast or icecast, some let you operate WinAmp controls from the Windows TaskBar tray, or even control WinAmp over a local area network (LAN).

## *Listening to MP3 Streams*

MP3 files don't have to be downloaded to your hard drive, necessarily. In many cases, you'll be able to play them back in your favorite MP3 player directly from the server on which they're located. As you'll discover in Chapter 8, there are several ways in which MP3 webmasters can dish up files for "streaming." As an end user, you won't need to worry much about the particular streaming technique in use, though it's interesting to know the difference. In some cases, you may have to tweak a few settings in your browser or player to make sure streamed files are handled by your operating system properly.

### *Types of Streaming*

There are two primary ways in which MP3 files can be streamed to users without being downloaded: MP3-on-demand and MP3 broadcast.

#### *MP3-on-demand*

In this form of streaming, control of the download is in the hands of the MP3 player, rather than the browser. Because this capability is built into most MP3 players, users can choose at any time to listen to an MP3 file directly from a web server, without saving it to their hard drives first. Of course, this assumes that the user has sufficient bandwidth to listen to the file in real time without it skipping or halting, but we'll get to bandwidth issues later. If you have a fast Internet connection, look around in your player's menus for an option labeled something like "Open Location" or "Play URL" and enter the URL of any MP3 file on the web. The easiest way to get this information is to right-click a link to an MP3 file in your browser and choose "Copy Link Location" from the context menu, then paste the URL into the Open Location dialog in your player.

In addition, MP3-on-demand can be forced by the webmaster, so that clicking a link normally will cause MP3 files to be pulled down by the player and played

directly, rather than saved to hard drive as with a normal download. To do this, the webmaster creates an "M3U" (MPEG URL) playlist file, which is a plain text document containing the full URL to an MP3 file (or list of files) on a web server. Because the text file is tiny, the browser can download the M3U file to the user's hard drive nearly instantaneously. The web server sending the M3U file should (if it's configured correctly) dish it up with the MIME type *audio/x-mpegurl*. This MIME type should, in turn, be associated in the user's browser or operating system with a preferred MP3 player capable of handling MP3 streams.

Once the M3U file is downloaded, it's launched in the preferred MP3 player, which reads URLs out of the file and takes over control of the actual download.



Note the difference here: when you download an MP3 file normally, the browser itself handles the entire download, and users have to then launch the MP3 file in an MP3 player manually. The MP3 file is stored on the user's hard drive for future use. With MP3-on-demand, the MP3 player handles the download, not the browser. The MP3 player plays the file *as* it's being downloaded, not later on. And unlike a standard download, the MP3 file is not present on the user's system after they've finished listening to the track.

The advantages of using the MP3-on-demand technique are:

- The user does not have to wait for the download to complete before beginning to hear music.
- The user has more control over playback than with real streaming (e.g., the user can skip around between songs or fast-forward through songs at will).
- The webmaster has a degree of protection against MP3 files being stored permanently on the user's system.
- The publisher or webmaster does not have to set up any special MP3 serving software by having the MP3 player "suck" the file down. A plain vanilla web server running on any operating system is capable of serving up MP3-on-demand.

Because MP3-on-demand offers so much flexibility to both the webmaster and to the user, it may eventually become more popular than true MP3 streaming if and when all of us have lots of bandwidth. As long as our bandwidth is limited, however, true streaming solutions will continue to outweigh on-demand systems in popularity.



An important aspect of the MP3-on-demand technique is that it's "asynchronous," or outside of time. In other words, it doesn't matter what time of day the user accesses the file—he'll hear it from the beginning. This is very different from TV, radio, or MP3 broadcast, where you get whatever is being broadcast at the moment in time when you tune in to the channel. For this reason, the MP3-on-demand technique is also sometimes referred to as "pseudo-streaming."

A good example of MP3-on-demand can be found at MP3.com. Access any artist's page and access one of the Hi-Fi or Low-Fi links. Rather than being prompted for a download location, your favorite MP3 player will be launched and the file should start playing immediately.

See Chapter 8 to learn how to set up your server for MP3-on-demand.

### **MP3 broadcast**

In contrast to pseudo-streaming, MP3 broadcasting (or real streaming), is "synchronous," and thus more akin to TV and radio broadcasting. In this case, the user tunes in to a channel or station which is playing an MP3-encoded bitstream much like a radio station, sometimes complete with live announcements and commercials. The person running the MP3 server is running the show in real time, and the listener only hears the portion of the show currently being dished up. When you tune in to an MP3 broadcast, you can't just pick an arbitrary tune from the show, any more than you can with radio.

Running an MP3 broadcast station is a fairly complicated matter, and you'll learn all about that in Chapter 8. As a user, however, listening to streamed MP3 is rarely more than a matter of point and click. To find MP3 broadcasts, check out sites such as [www.shoutcast.com](http://www.shoutcast.com), [www.iccast.org](http://www.iccast.org), [www.radiospyspy.com](http://www.radiospyspy.com), [www.mycaster.com](http://www.mycaster.com), [www.greenwitch.com](http://www.greenwitch.com), or [www.live365.com](http://www.live365.com) and you'll find dozens—or hundreds—of ongoing broadcasts. If your system is configured properly, clicking a link to a stream in progress will cause your MP3 player to be launched and (after a short delay) for that stream to begin playing. If it doesn't, see the following section, "Configuring Your System to Handle Streaming MP3."



Real MP3 streams are usually sent as *.pls* files (MIME type `audio/x-scpls`), rather than *.M3U*. The difference between these two playlist types is described earlier in this chapter.

The advantages to real-time MP3 streaming are:

- Much greater control for the webcaster (voice, live mixing, etc.)
- Webmaster can send a stream to many people without needing tons of bandwidth on the playback machine (though they still need access to a server with a fast connection)
- Difficult for user to save MP3 bitstream to hard drive
- Optimization of bitstream for various client bandwidths

### *Of bandwidth and buffers*

To receive MP3 streams from MP3 broadcasts or pseudo-streams, your player must be capable of managing downloads and buffering streams over the Internet on its own. The vast majority of popular MP3 players are stream-enabled—even many of the command-line players for Unix/Linux.

The biggest concern for most users, of course, is the speed of their Internet connections. If you're on a slow connection and the stream being served up (or pulled down) carries more bits per second than your modem is capable of delivering, you'll experience choppy, halting playback.

This can be mediated somewhat by two solutions. On the client (user) side, a process called *buffering* can be used. In the buffering process, the MP3 player grabs a good chunk of data before it begins to play, and continues to read ahead in the stream. The music being played is thus delayed by a few seconds. The slower the connection, of course, the larger the buffer required. Theoretically, one could utilize a buffer so large that the entire song was downloaded before a second of music was played. This would guarantee perfect playback over even the slowest connections, but would undermine the advantage of listening to streams.

Most users, however, require more modest buffer settings. If you find that your MP3 streams are skipping or pausing as they're played, dig around in your player's options and preferences for something like "Streaming Preferences." In WinAmp, tap Ctrl+P to bring up the preferences screen and navigate to Plugins → Input → NullSoft MPEG Audio Decoder. Click the Configure button and select the Streaming tab, where you'll find an array of buffering options. Most likely, you'll just want to change the numerical value for kilobytes of prebuffered audio (try increasing it by 25% or 50% for starters). You can also control how much of a track will be grabbed before a single byte is played.

On the server side, webmasters can do a number of things to make things easier for their modem-connected users, including downsampling MP3 files to lower frequencies, encoding files at lower bitrates, and sending mono, rather than stereo, streams over the Internet. All of these, of course, degrade sound quality, but are

necessary to deliver acceptable streams to modem users. A sophisticated MP3 server will offer high-bandwidth and low-bandwidth options so users can select the best possible quality for their connection speed (such as the Hi-Fi and Low-Fi pseudo-streaming options found at MP3.com).



Technically speaking, on-demand servers are capable of doing some of the things that broadcast servers do, such as downsampling MP3 audio on the fly. For the technically inclined, this would be a matter of creating a CGI interface that would invoke a downsampling program when the user accesses a link on the server, and sending the output of that program to the user rather than the actual file. This is really a CGI implementation, rather than an MP3 issue. While there aren't many sites doing this currently, the technique would have certain advantages for some users, since it wouldn't require the installation and configuration of special broadcast software. More on this in Chapter 8.

### *Configuring Your System to Handle Streaming MP3*

Regardless whether you're accessing real MP3 broadcasts or pseudo-streams, the ideal is to have your browser, your operating system, and your MP3 player all configured to interoperate correctly so that accessing a link to an MP3 player automatically results in the right thing happening—your player gets launched and begins to play the stream with no further intervention on your part. In most cases, simply installing an MP3 player capable of handling MP3 streams is all it takes to set things up properly, but it's possible for the necessary associations to become broken if you fiddle around with a lot of software or tweak your browser settings. In addition, you may at some point want a different player associated with MP3 streams.

The easiest way to create an appropriate association is to look in the options and preferences of the preferred player for something like "Make preferred for all types." Better players will let you establish associations on a per-filetype basis so that you can, for example, make Sonique your preferred MP3 player for regular MP3 files, and WinAmp the preferred player for streamed *.M3U* and *.pls* files.



If you have multiple MP3 players and browsers on your Windows system, the easiest way to establish associations may be to download the MP3Fix utility from [help.mp3.com/help/diagnosis/](http://help.mp3.com/help/diagnosis/). Although MP3Fix is packaged in a Windows InstallShield package, running it will not in fact install anything on your system. Rather, it will scour your system for known MP3 players and allow you to make one of them the preferred player for all browsers.

*Netscape Navigator/Mozilla*

If Netscape Navigator is your primary browser, you can tell the program exactly how to handle any incoming MIME type (file type) by pulling down Edit → Preferences → Applications (this may be slightly different in various versions of Navigator). Scroll through the list of known file types for something like MPEG Audio File or WinAmp Media File and click the Edit button. To change the associated MP3 player, click the Browse... button and navigate to the location of your preferred MP3 player. If you don't find an entry in the list for the file type you want to associate, you can create a new one by clicking New Type... and filling in a description, file extension (e.g., MP3), a MIME type (e.g., *audio/x-mpeg*), and the path to an MP3 player.

*Internet Explorer*

If you use Internet Explorer as your browser, remember that Explorer is integrated into Windows itself. Therefore, it does not use a separate MIME association database. Instead, it uses the operating system's FileTypes panel. To change an association manually, open Windows Explorer (not Internet Explorer) and pull down View → Folder Options → FileTypes. Navigate to the Playlist file type (which may be called, for instance, "WinAmp Playlist"), click Edit, and use the resulting dialog to change the description and program association. More details on this procedure can be found in the sidebar "Configuring the Default Handler for MP3 Files in Windows" in Chapter 3.

## *Performance Considerations*

The average consumer machine of today is much more efficient than it was a few years ago, and most people have gobs of underutilized computing horsepower to spare. Nevertheless, there are still plenty of older machines hanging around out there, and owners of those machines will want to make sure they've got the most efficient decoder available. And, of course, true geeks will want access to the most efficient decoder whether they need it or not. Here are a few techniques you can use to determine the efficiency of your MP3 player. The same techniques apply to MP3 encoders, by the way.

### *Benchmarking Decoders*

As mentioned in Chapter 2, MP3 decoders have a lot less work to do than encoders, since the task of MP3 playback is far less CPU-intensive than encoding. All the decoder has to know is how MP3 files are structured, and how to reconstruct a coherent signal from the combination of audio data and "side information" stored in the files' frames. In days of yore, when a Pentium 90 was considered top-of-the-line and most people were using 486-based computers, playing MP3 files was proportionally intensive enough to be cause for concern, and many experienced a significant impact on system responsiveness when MP3s were playing in the background.



However, modern CPUs have so much processor bandwidth to spare that decode speed isn't much of an issue for most users. For example, playing an MP3 stream on a Pentium 233 with the average decoder for Windows will generally consume only 5% of your processor speed.

If you're experiencing a noticeable performance impact when playing MP3 files, you might want to do a little benchmarking of your own. CPU/resource monitors are available for most operating systems, either as part of the system itself or as a separate download. A few of the more popular options are listed in this section. If you can't find a CPU monitoring app for your system that breaks up CPU usage by task (as opposed to giving you an overall rate), just note CPU consumption with and without your MP3 decoder running, then calculate the difference.

Once you've got a resource meter up and running, try several different MP3 players and take note of how many system resources (CPU and memory) they consume. Keep in mind, however, that even if a particular MP3 player consumes less resources than another, it may do so at the expense of sound quality or important features. You'll have to establish your own criteria for the compromises you're willing to make.

#### *Windows 95/98*

Windows 95/98 both ship with a resource meter bundled in the system. If `C:\WINDOWS\SYSMON.EXE` is present on your system, you're all set. You may find a shortcut for Sysmon pre-installed in Start → Programs → Accessories → System Tools. If not, go to Control Panel → Add/Remove Programs → Windows Setup → Accessories and add it. Sysmon's default display shows the percentage of total CPU available being consumed by all currently running tasks, but does not tell you exactly how many resources are in use by each individual process. The large spikes in Figure 4-11 occurred when launching applications, and resulted in WinAmp skipping. The flat areas represent straight MP3 playback. Several other applications were open, but no work was being done. Sysmon reports approximately 15% processor consumption, but gives no indication of exactly how much of this is being consumed by MP3 playback. For a more specific breakdown or resource consumption, download Microsoft's "Kernel Toys," right-click *top.inf*, choose Install, then click Start → Run, and type *wintop*.

Popular third-party resource meters for Windows include L-Ement and sysmeter, both downloadable from [www.skinz.org](http://www.skinz.org).

#### *Windows NT*

Windows NT 4.0 has a similar performance monitor built into the operating system. To access it, tap Ctrl+Alt+Del, click Task Manager, and select the Performance tab. Alternatively, use Start → Programs → Admin Tools → Performance Monitor.

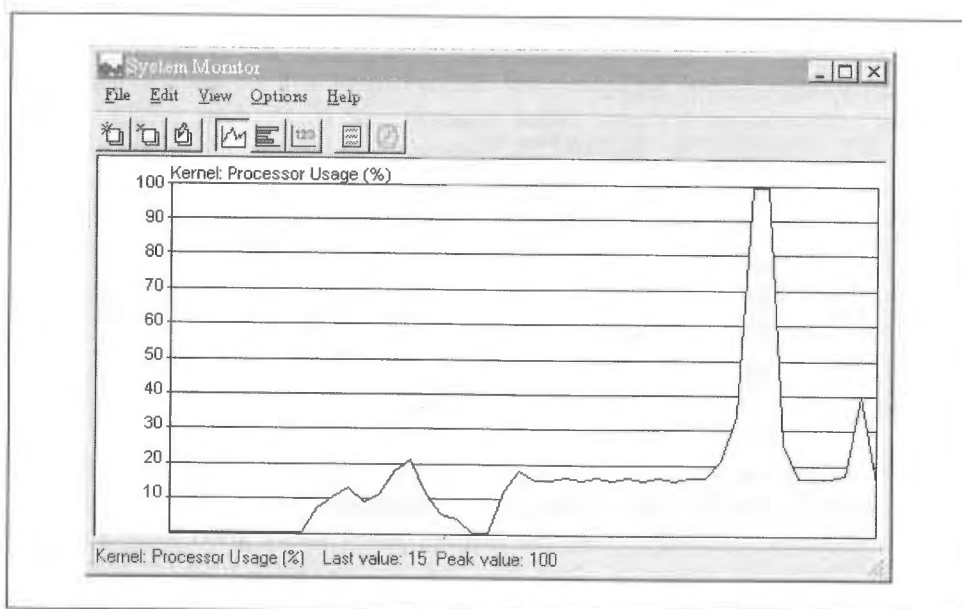


Figure 4-11. Using Sysmon to track CPU resource consumption as various tasks are performed

### Mac OS

Search your favorite Mac software library for a utility called Process Watcher by Hugues Marty, which will allow you to view information on all currently running processes. However, Process Watcher will not show you a breakdown of CPU activity *per application*, so its ultimate usefulness is somewhat limited. You'll find a copy at [perso.magic.fr/suli/pw.html](http://perso.magic.fr/suli/pw.html) if you want to check it out. Users may also want to experiment with Process Monitor or Process Spy, though this utility suffers the same limitation.

### Linux

There are dozens of options for Linux users. Typing `top` at any prompt will give you a detailed overview of resource consumption, broken down by process. By default, `top` will update its report once per second, which may be too rapid to study properly. If you want a better view, use:

```
top -d 5
```

to have `top` update itself every five seconds. The problem with `top` is that it isn't always easy to figure out exactly how much CPU a given application is consuming, because many apps do their work in worker threads that may be difficult to identify. Fortunately, you can launch an app through the "time" utility. The operating system will calculate the total time the app was running and report the total

number of seconds of processor time consumed in system space and in user space. From that, calculating the percentage of available CPU consumed by that app time's syntax is pretty simple:

```
time /path/to/mp3player /path/to/Song.mp3
```

As soon as mp3player finishes playing the song, close it and you'll get a report like this:

```
real 1m42.708s
user 0m9.427s
sys 0m1.769s
```

To calculate CPU usage, use this formula:

$$(\text{user} + \text{sys}) * 100 / \text{real} = \text{percentage of available CPU time}$$

In this example, the player in question is consuming around 10.9% of available CPU resources. Note that these figure will be slightly off because launching the app will consume extra CPU time, but it gives you a pretty good idea of what's going on. There are dozens of alternative resource monitors available for Linux. Search your favorite software archive for GUI resource meters such as xosview.

### BeOS

In addition to the Pulse application built into the system (which shows CPU load indicators for each detected processor), there are quite a few third-party tools available, downloadable from [www.be.com/beware/](http://www.be.com/beware/), such as the graphical process controllers TManager and ProcessWatcher. The "top" and "time" utilities described for Linux are also available for BeOS, and function identically.

## System Requirements

A few MP3-related hardware issues were covered in Chapter 2. To summarize and put it all in one place, here are the most important things you need to consider when contemplating hardware upgrades for your MP3 playback machine.

### CPU speed

While you can theoretically use a 486 for MP3 playback, the experience is likely to be painful, as even the comparatively low-resource act of decoding MP3 is going to consume a good deal of horsepower that you'll probably want to give to the operating system itself. Trouble-free MP3 playback really demands a Pentium-class computer or better, where processor consumption may be around 5–10%, rather than 50–75%. Mac users, similar deal: Anything slower than a PowerMac or PowerMac clone is going to give you problems with MP3 playback, if you can even find an MP3 player that works on your machine at all.

Encoding is another matter altogether. Because encoding requires so much math, you'll be happiest with the fastest machine you can get your hands on. While fast encoders running on a modern Pentium can encode a song faster than it would take to play it back, you'll hear absolute horror stories from users who try to do their encoding on an antique machine. An old 386, for example, may take 12 hours or more to encode a single 3-minute song.

### *Memory*

MP3 encoders and players aren't huge memory hogs, but they do require more memory than, say, a simple text editor. The process of encoding will benefit more from extra RAM than will playback, but unless you already feel your memory subsystem could use a shot in the arm, don't go running out to buy more memory just to improve the MP3 experience. At the most, getting into MP3 probably means you'll be running one additional application in the background most of the time, and a relatively lightweight application at that.

### *Sound cards and speakers*

Again, this is a critical area. Most people have traditionally thought of their sound cards as a vehicle for games, or for listening to streamed news broadcasts. But as you get more involved in MP3, you'll start to think of your computer as a cousin of your home stereo in some ways, and will want to get the best quality you can. Do some research online to find out what the audiophiles are recommending this month as the best-sounding card at an affordable price. In the sub-\$100 range, top cards from CreativeLabs and Yamaha were among the most popular cards among MP3 buffs at this writing.

Chances are, the speakers that came with your computer aren't of the best quality. Again, look around online for recommendations. Get yourself a nice set of powered subwoofer/satellites, connect them to your new sound card, and be prepared to be blown away by the difference they make over the default audio hardware that probably shipped with your computer, as shown in Figure 4-12. Because the system has a built-in amplifier and runs on standard wall power, you don't need to rely on the amplifier built into your sound card. If you're going speaker shopping for your computer, you'll get better results by going to an audio store and telling them you want speakers for use with your computer than by going to a computer store and telling them you want high-quality computer speakers.

Of course, you can always route audio signal from your computer directly to your home stereo by purchasing a length of RCA cables and a simple adapter. See Chapter 6, *Hardware, Portables, Home Stereos, and Kits* for details.

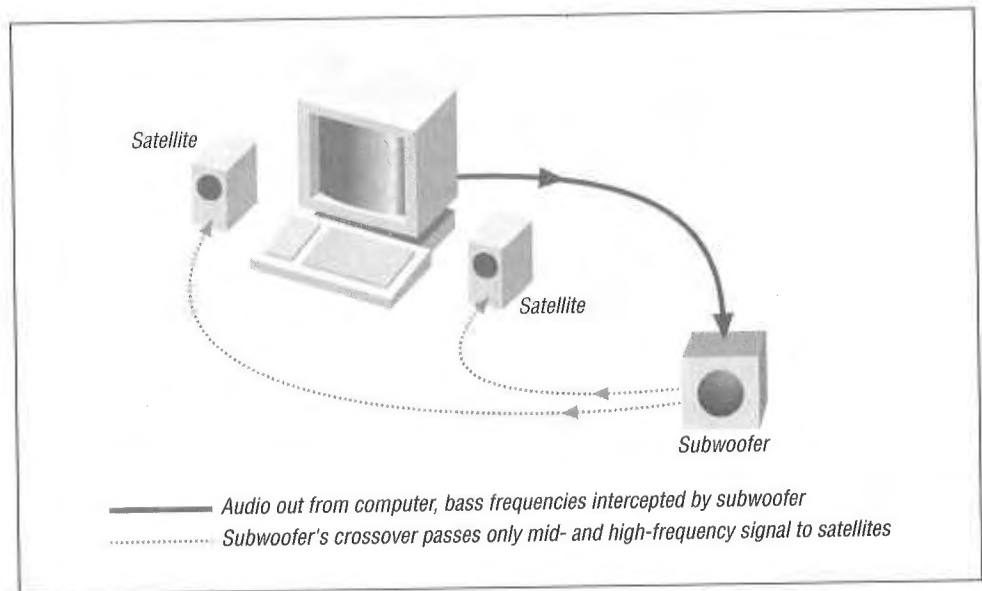


Figure 4-12. Mid-range and high frequencies are handled by satellite speakers

#### Disk speed and space

People sometimes wonder whether their hard drive is fast enough for good MP3 playback, or contemplate going with a high-end SCSI adapter/disk to optimize their systems for MP3 use. For the most part, this is piffle. At the default bitrate of 128kbps, a mere 16 K is being scooped from disk every second—even the lowly floppy drive can keep up with that. All of the work is going on in the CPU. The fact that MP3s are so highly compressed not only means you save on disk space, but that the hard drive has to do very little work.

However, there are some caveats to this point. If your all-IDE system has more than one hard drive and one CD-ROM drive, you'll end up putting two devices on a single channel with IDE. But IDE, unlike SCSI, cannot transfer data back and forth between two devices on the same bus or channel simultaneously—IDE accesses are serial. Therefore, you can end up with your MP3 player requesting data from a storage volume while the operating system is requesting data from the boot volume simultaneously, which can potentially lead to skips or pops in the audio stream. But IDE disks are so much more affordable than SCSI that most users in this position will simply have to deal with it. If you need to run several IDE devices and want to ensure optimum performance, you might want to consider purchasing inexpensive IDE controller cards for the additional drives, so that each one gets a channel all to itself.

Note also that IDE disks are more CPU-intensive than SCSI disks, which can have an impact on low-performance systems. If you have UDMA IDE drives, make sure they're running in UDMA mode to lessen the processing impact on the CPU.

Once you start on the task of encoding your entire collection, or downloading lots of tracks from the Internet, you'll likely find disk space being chewed up faster than you ever dreamed possible. Suddenly those 30 GB drives on sale down at the local computer superstore don't seem obscenely huge after all. Of course, it won't be long before you're looking for some kind of external storage solution, and most people start burning data CDs full of MP3 files to deal with the space issue. See Chapter 5 for more on that.



When playing MP3 files from CD rather than from your hard drive, you may experience skipping, or "halting" playback, especially if you're using an older or cheapo CD-ROM drive. This is probably not happening because the CD-ROM drive isn't fast enough, but because it isn't accurate enough and has to do a lot of error correction to grab data accurately. You may be able to correct for this by increasing the "buffer" or "preload" size in your MP3 player. Dig around in the options and look for a control that will let you establish the size of the "preload" or "buffer" size, and turn it up until the problem goes away. By doing this, you're telling your MP3 player to read data from disk in larger "gulps," which gives the drive more time to retrieve data from CD successfully before the amount of music stored in RAM is exhausted.

#### *Sound/video card interactions*

One of the most frequently encountered hardware-related problems users find is that even very fast machines can "skip" or "pop" MP3s when switching tasks, scrolling windows, or doing other things that require a lot of screen redraw. This typically occurs for users with PCI graphics cards, because some card manufacturers have attempted to increase card performance by letting video cards take control of the entire PCI bus when necessary, which can prevent an even stream of audio data from reaching a PCI sound card at an adequate rate. Many people have noted that this does not occur with AGP graphics cards or ISA sound cards (since they're on a separate bus), which you may want to keep in mind when shopping for new hardware. If your audio and video cards are both on the same bus, you may be able to tweak some of the video card's driver or BIOS settings to disable or lessen this effect, though possibly at the expense of some graphics performance.

# 5

## *Ripping and Encoding: Creating MP3 Files*

### *In this chapter:*

- *General Encoding Principles*
- *General Ripping Principles*
- *Ripping and Encoding Tools*
- *Ripping from Other Sources*
- *Roll Your Own Compact Discs*

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/](http://www.syntrillium.com/cooledit/)) or GoldWave ([www.goldwave.com](http://www.goldwave.com)). However, these tools are actually decoding MP3 to PCM audio behind the scenes, and then re-compressing 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 psychedelized 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/](http://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 Real-JukeBox 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 as 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](http://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](http://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](http://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](http://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 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 RAW 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](http://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](http://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.

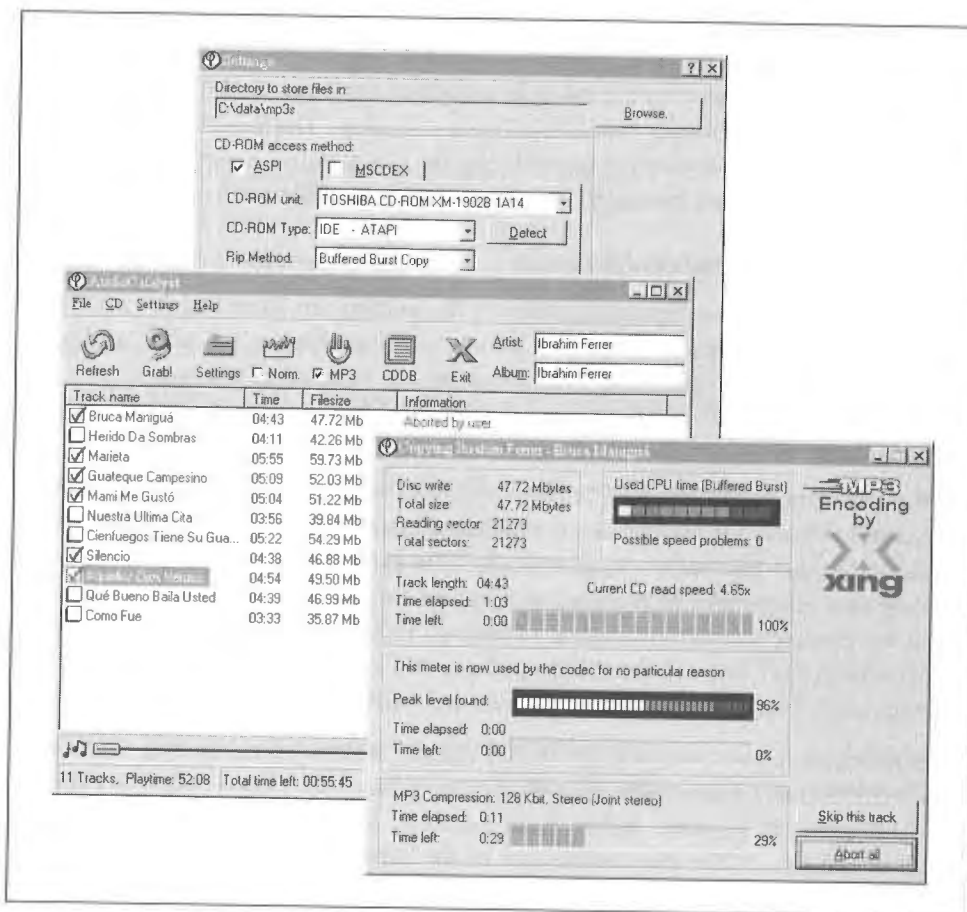


Figure 5-2. AudioCatalyst has it all: a fast encoding engine, a no-nonsense interface, and tons of features and options



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](http://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 Real Networks' "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 "cdfv.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.swtpnet.se/~w-82625/](http://home.swtpnet.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](http://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](http://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, sample rate, and (very important) whether or not to use ID3 tags. Click the Start Converting button and let ‘er rip.

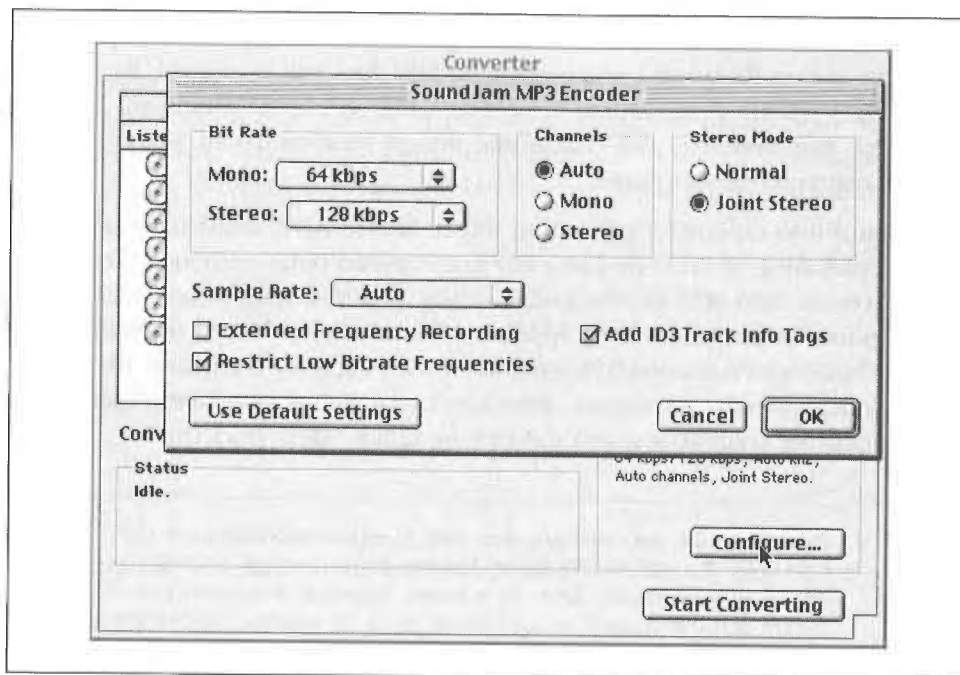


Figure 5-3. Before encoding with SoundJam, be sure the “Add ID3 Track Info Tags” option is enabled



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](http://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](http://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/](http://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/](http://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.

### *cd-da2wav*

*Available at nearly any Linux software site.* If you prefer speed over accuracy, you might want to try the somewhat faster `cd-da2wav`, instead. Don't leap to the conclusion that your rips are going to be full of errors if you use `cd-da2wav`; 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. `cd-da2wav` 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 SCSI 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/](http://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](http://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/](http://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 CDDDB 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](http://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.

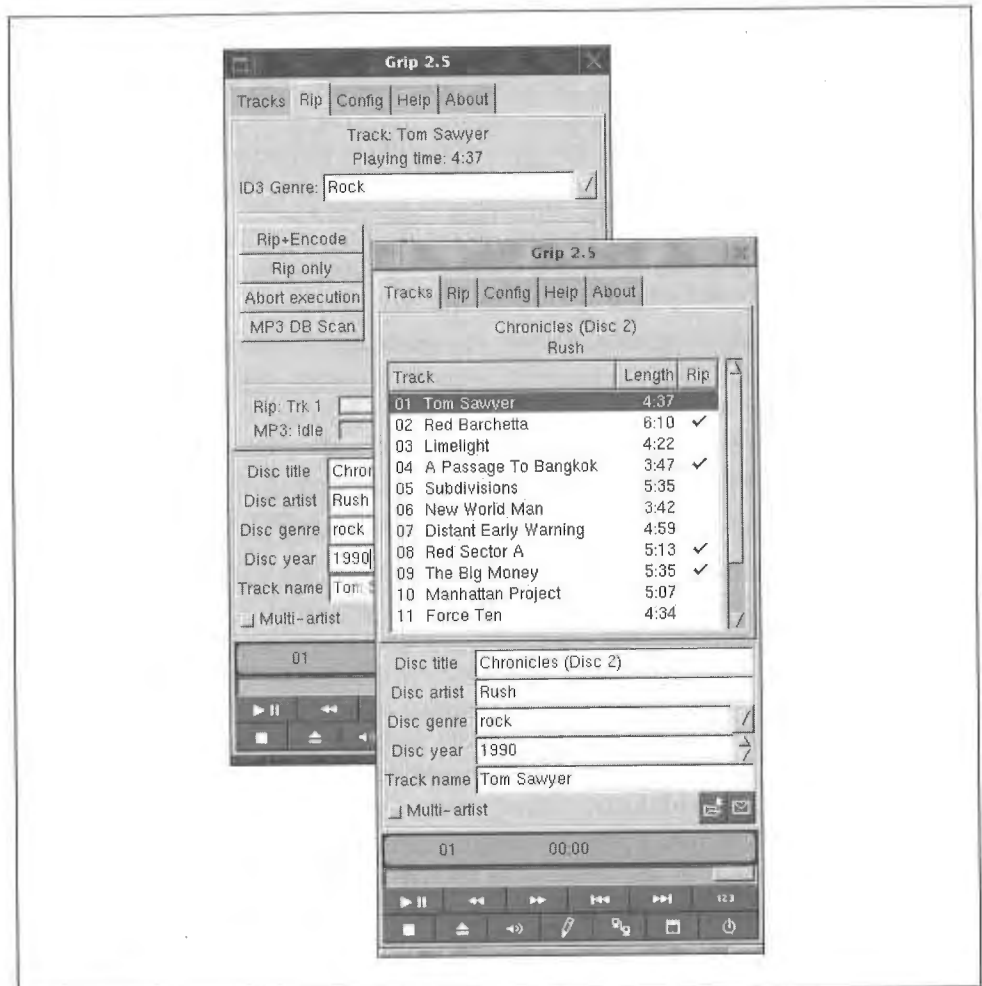


Figure 5-4. Grip for Linux serves as a GUI frontend to command-line tools

### RipEnc

[www.asde.com/~mjparme](http://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](http://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*.

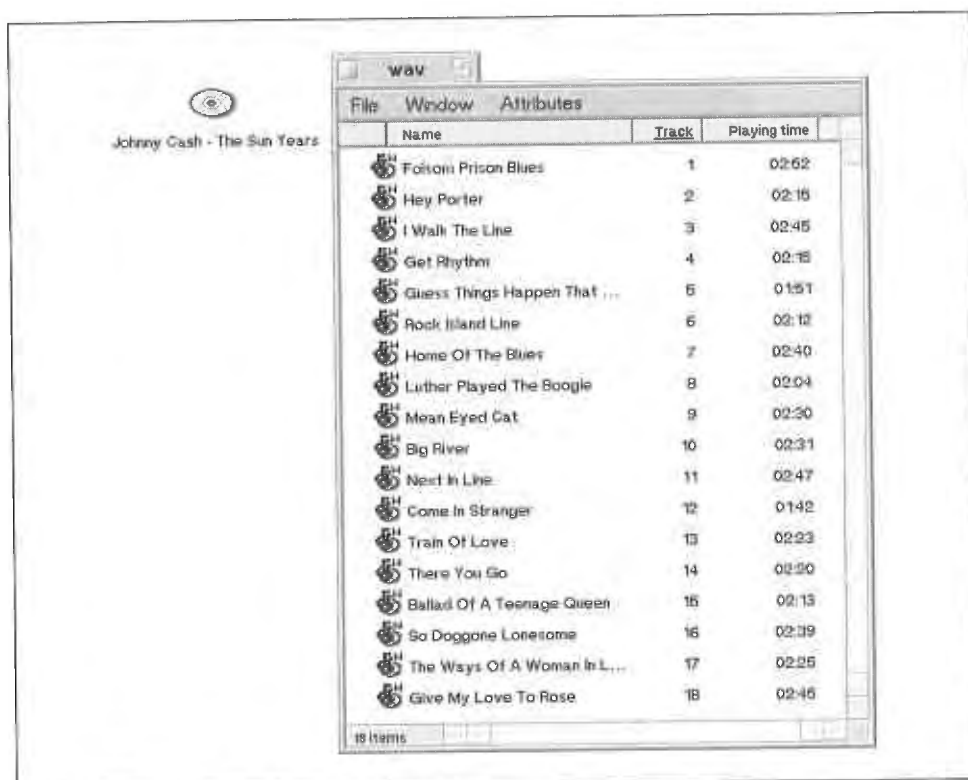


Figure 5-5. Tracks on audio CDs in BeOS appear to the Tracker as WAV files

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).