

Ask Ars: CD Ripping and encoding guide

By Rian J Stockbower

July 2003



ars technica

Copyright CondéNet, Inc. 1998-2010. The following disclaimer applies to the information, trademarks, and logos contained in this document. Neither the author nor CondéNet, Inc. make any representations with respect to the contents hereof. Materials available in this document are provided "as is" with no warranty, express or implied, and all such warranties are hereby disclaimed. CondéNet assumes no liability for any loss, damage or expense from errors or omissions in the materials available in this document, whether arising in contract, tort or otherwise.

The material provided here is designed for educational use only. The material in this document is copyrighted by CondéNet, Inc., and may not be reprinted or electronically reproduced unless prior written consent is obtained from CondéNet, Inc. Links can be made to any of these materials from a WWW page, however, please link to the original document. Copying and/or serving from your local site is only allowed with permission. As per copyright regulations, "fair use" of selected portions of the material for educational purposes is permitted by individuals and organizations provided that proper attribution accompanies such utilization. Commercial reproduction or multiple distribution by any traditional or electronic based reproduction/publication method is prohibited.

Any mention of commercial products or services within this document does not constitute an endorsement. "Ars Technica" is trademark of CondéNet, Inc. All other trademarks and logos are property of their respective owners.

Ask Ars: CD Ripping and encoding guide

By **Rian J Stockbower**

Introduction to audio formats and bitrates

MP3, MP2, MPC, APE, FLAC, AIFF, WAV, OGG, etc. The list of audio formats out there seems to go on and on, and to the uninitiated, it's a daunting task sorting through them all. Chances are, most of us started out using an all-in-one ripper/encoder. For me, it was MusicMatch JukeBox, back when you had to pay to encode anything higher than 96Kbps. Times changed, and I discovered the joys of p2p, back before the RIAA made a big stink about file-sharing. I am embarrassed to admit that I downloaded MP3s based on their file size... the smaller the better. You see, I had not made the tenuous connection between quality and file size. All that mattered was that I had dialup Internet access, and the quicker I could download a file, the better. Come to think of it, I didn't even know what variable bit rate (VBR) was, and when I saw the bitrate changing in Winamp one time, I deleted the file because I thought there was something wrong with it.

Needless to say, things changed somewhere along the way; I started choosing files that had a bitrate of at least 160Kbps. Then I started learning about MP3 encoding in general, and more specifically, that not all encoders were created equal. I still had a lot to learn. Chances are, there are quite a few people out there who are like me back in the day: people who haven't the foggiest idea what terms like "variable bit rate" and "lossless encoding" actually mean. (And who perhaps couldn't be bothered to learn, either.)

Audio formats are roughly divided into two groups: lossy and lossless. The words are relatively self-explanatory. Lossy formats are those codecs that approximate the audio you hear rather than actually storing the complete audio data. Lossless audio uses codecs that compress the audio without losing any quality. You might compare a lossless encode to a zip file: smaller than the original, but no data loss. Typically, lossy encodes produce file sizes that are smaller than lossless encodes, which is one of the reasons that lossless formats are not as ubiquitous. As the cost per gigabyte of storage decreases, file size is becoming increasingly less relevant, and more people are beginning to favor lossless encodes.

"Bitrate" refers to the rate that data is flowing in bits per second. If one were to encode a three minute track at 160Kbps in MP2 format, it would be the exactly same size as the same track encoded in MP3 format. Bitrate says nothing about quality directly; it only describes how much data is being processed in any given second. While this does translate to measures of quality when encoding audio in a general sense, it is not a perfect measure of quality. What it boils down to is this: the greater the bitrate, the more data is played back every second. It is imperative that one understand this before truly understanding the whys and hows behind CBR, VBR, and ABR encodes. (And why, for instance, a VBR encode might be smaller and yet sound better than a CBR encode.) There will be much more on bitrates later.

The MP3 format

History of the MP3 format

The MP3 format is actually a pretty old technology, though one wouldn't think so. With the Internet boom in the late nineties, "MP3" became a household term. In reality, MP3s have been around since the eighties. The standard was created by Fraunhofer Institute, the same people that created the Fraunhofer encoder that Ars determined was the best back in 2000. Since that time, however, Fraunhofer has closed their source, and is now charging a licensing fee for their encoder. More on Fraunhofer later.

The time was right for MP3 in the late nineties, as a lot of factors were coming together that would allow the MP3 music format to thrive. Hard drive space was increasing, the cost per gigabyte of storage was decreasing, and the number of people with broadband was increasing rapidly (which has led to the file-sharing controversy, which I'm sure most of you are familiar with). The audio equipment available to audiophiles and PC enthusiasts alike was becoming more powerful and less expensive. MP3, which had been around for ages merely filled a void.

Since the beginning of the Internet boom, audio compression and encoding technologies have advanced far beyond the MPEG-1 Layer III format. Nonetheless, the ubiquity of MP3s will likely render the format popular for a long time to come... for better or worse.

Encoding MP3s

Well this is why you are here, is it not? Encoding MP3s. There are three types of MP3 encodes: variable bitrate (VBR), constant bitrate (CBR), and average bitrate (ABR).

Constant bitrate MP3s are the most common type of MP3 found. If you download an MP3, chances are it's going to be a CBR MP3. What this means is that the file was encoded using the same bitrate for every part of the song: it is constant from start to finish, unlike in an ABR or VBR encode. The user simply picks a bitrate that they wish to encode at, and they're off to the races.

CBR has some shortcomings, unfortunately. Let's use a 192Kbps encode as an example. Suppose you are listening to a piece of music; it starts off quietly: soft and simple. 192Kbps is perfectly fine here: in fact, it's wasteful. There is no need to be using 192Kbps of bandwidth to accurately describe the audio you're hearing, but we're not losing any quality. The piece continues getting more complex, and the sheer volume of information needed to accurately describe what is happening in any given second increases. Suddenly, 192Kbps is not enough, and the encoder is forced to make a compromising choice which often results in distortion and distracting audio artifacts. Needless to say, CBR falls short.

What we're left with is a case of too much bandwidth in some places, and not enough in others. So what's the solution? A 320Kbps CBR encode? Well, this is certainly one solution, and it's a solution that many audiophiles use. For most people, however, a VBR encode would have been a better choice, for reasons that I shall detail shortly.

VBR MP3s are a slightly different animal. In the past, there has been some controversy surrounding VBR MP3s, because for a long time (back when Ars first tackled this subject), VBR was a crude hack. It was not widely supported: encoders did not encode it correct, and many of the available decoders were non-standard. Since then, however, things have changed. VBR is widely accepted and software support is universal. It is considered the most desirable way of encoding an MP3 for most people.

A VBR encode is exactly what it says: an MP3 whose bitrate changes throughout. What might not be obvious is why a VBR encode is effective or desirable. Basically, the encoder analyzes the audio file it is encoding, and changes the bitrate as it encodes based on how much data it needs to accurately describe the music you're hearing. For instance, when there is a whole lot of sound for the encoder to encode, it increases the bandwidth as needed so as not to produce artifacts. In contrast, a different part of the song might be less complex. On this part of the track, the encoder will lower the bitrate because a higher bitrate is unnecessary and simply consumes more space.

Going back to the CBR example, a VBR-encoded file would have adjusted the bitrate as needed; producing a better-sounding rip, and possibly a smaller file to boot. It's rather the best of both worlds. The LAME encoder (unless you specify otherwise) uses the full 32Kbps to 320Kbps available when encoding a VBR file. Some people prefer to set lower and upper bounds on their encoder when they encode their music files, but a good encoder like LAME with one of its presets will adjust as necessary. Some people have complained about poor results with VBR encodes. This has nothing to do with VBR technology itself, but rather poor implementation in a particular encoder. This is why LAME is preferred: it works like it's supposed to.

Average bitrate, or ABR, is very similar to VBR. ABR is not a constant bitrate encode, nor is it a pure VBR encode. It's similar to CBR in that the user picks a bitrate that they would like to encode at. Let's say 192Kbps. The encoder then begins encoding in a manner similar to VBR, but it tries to stay close to the bitrate that the user specified: in essence, have an average bitrate of 192Kbps when the file is finished encoding.

Some last notes: Windows Media Player now freely supports encoding to MP3 from CD. Unfortunately, it uses the inferior Fraunhofer codec. iTunes, however, does not use the Fraunhofer codec. Despite some recent confusion, it turns out Apple wrote their own MP3 encoder for use in iTunes. Unfortunately its quality is much worse than their AAC encoder or other free MP3 encoders. Regardless, WMP does not support VBR encoding, and iTunes doesn't use LAME, and LAME is what all the cool kids are doing, so I recommend you use LAME. It's just better.

Generally speaking, VBR is better than ABR is better than CBR. But remember, there will always be people telling you that your way sucks, no matter what you do. It is inevitable. I'd probably be one of them if you were encoding to MP3 using anything but LAME.

Why LAME?

The history of LAME

The history of LAME is rather long and involved, nonetheless, it is interesting to see how some open source projects develop over time. Basically, LAME was started in 1998 by a programmer and audiophile named Mike Cheng. In his November, 1998 [rationale](#), Cheng outlines why and how he started out playing with MP3-encoding technology. Cheng was an Amiga addict, and after playing with layer 2 compression, he heard of layer 3, and decided to play around with Fraunhofer's source, and port it to Amiga and tweak it to speed it up. The group he was involved with, 8hz, was forced to halt their efforts (along with other groups) when Fraunhofer cracked down on MP3 encoding development in September of 1998.

At this point, Cheng had not released an MP3 encoder: just a patch to the dist10 source. Anyway, this patch was called LAME which stands for LAME Ain't an MP3 Encoder. The patch he released was incapable of being compiled or producing an MP3—in order to create an encoder capable of producing an MP3, one needs the ISO mpeg source.

Cheng goes on to outline two of the things that he wanted to do to simplify MP3 development. His list is worth mentioning simply to see how much things have changed since his rationale was written

- Only 44.1KHz samples
- Only 128Kbps bitrate

Cheng wanted to limit bitrates to 128Kbps because he believed that the MP3 standard was unnecessarily complicated as it attempts to support bitrates from 16Kbps to 320Kbps, which require different acoustic and computational models. (He also considers a 128Kbps MP3 CD-quality.) It sounds as though I am knocking Cheng's efforts, but this is not the case at all. Cheng was one of the main pioneers of MP3-encoding technology, and for when this rationale was written, the changes he wanted to make made absolute sense. As Twain said, we must judge people from the perspective of their own time, not ours. For his time, Cheng was certainly a pioneer.

Cheng exited LAME development in 1999, when Mark Taylor took over. Taylor still runs the LAME project.

Why the big deal?

So why the big deal over LAME? Quite simply, because it's the best. Unlike the other developers of MP3-encoding technology, LAME is constantly under development. LAME is the only MP3 encoder that properly supports VBR and ABR encoding. The LAME developers also listen to the audio community, which is not surprising, because the majority of the LAME developers are audiophiles themselves. The LAME encoder supports a myriad of presets, which makes life a bit easier by eliminating the need for lots of commandline arguments. I suppose that one could say that they are the epitome of the "Faster, Better, Smaller" computing adage.

Perhaps the only shortcoming of LAME is that its development often outruns the available documentation. As a result, much of the official LAME documentation is out-of-date and/or downright wrong. Some of the documented switches are broken or no longer supported at all. The current recommended version of LAME is 3.90.3, with other variations on this version floating around with slightly different version numbers.

Presets

Unless you are intimately familiar with the LAME encoder, your best bet is sticking with the presets built into LAME itself. You will not get better results than a preset at a given bitrate, because every time someone has figured out a way to better the presets, they have been promptly updated. Beyond even this, the presets make use of the psy model itself, which is not available to anyone but the programmers themselves. Specifying each and every switch yourself is not a good idea, because the switches are not accurately documented. Unless you have studied LAME in great detail, it's best to stick with a preset. The presets have gone through hundreds of double blind tests, and it is known exactly how they'll perform.

The presets are all VBR unless otherwise noted.

- `--alt-preset standard`
 - This is around 190Kbps; it typically fluctuates between 180Kbps and 220Kbps
- `--alt-preset fast standard`
 - This is also around 190Kbps, but it encodes faster, but potentially at a lower quality
- `--alt-preset extreme`
 - This is around 250Kbps; it typically fluctuates between 220Kbps and 270Kbps
- `--alt-preset fast extreme`
 - This is also around 250Kbps, but it encodes faster, but potentially at a lower quality.
- `--alt-preset insane`
 - This is a 320Kbps CBR encode. It is the highest possible quality using MP3 technology.

Other MP3 encoders

There are other encoders besides LAME, however they are less commonly recommended. Xing and Fraunhofer are two other commercial codecs that are of reasonable quality but not equal to LAME. Fraunhofer's VBR encoding is not widely used and generally not considered poor. However the codec does support Intensity Stereo, a form of Joint Stereo that will be implemented in LAME 4.0 that increases quality at extremely low bitrates.

Finally MP3Pro is a proprietary extension to the MP3 spec to enhance very low bitrate applications. However it is not widely supported and generally considered a hack.

MP3Pro stands apart from the other two because it's considered better than both of them. But at the same time, MP3Pro is intended for low bitrates, with decent quality at 64Kbps, but it is incapable of being truly good-sounding. It's also a hack, which leads to incompatibility issues: a 64Kbps MP3Pro is in actuality a 48Kbps MP3 with a low frequency cutoff, and which uses a different encoding scheme for higher frequencies—which is woefully inaccurate, but you can't expect much out of 16-32Kbps encode. MP3Pro supports the following:

- Mono: 18, 20, 24, 32, 40, 48, 56 Kbps
- LC-stereo: 18, 20, 24, 32, 40, 48, 56 Kbps
- Stereo: 32, 40, 48, 56, 64, 80, 96 Kbps

Regardless of what type of MP3 encoding you are doing, LAME is the best choice by a long shot. You can find the 3.90.3 binary [here](#).

Lossy alternatives to MP3

MP3 is by far the most popular method of encoding music. There are other formats that are gaining popularity, however. Most notably are Ogg Vorbis, AAC, and MPC. Each of these formats have their own merits and shortcomings. The main problem with all of them is that hardware support in portable media players is limited and/or nonexistent.

Ogg Vorbis

Ogg Vorbis is probably the most popular format behind MP3, with AAC rising quickly thanks to Apple. Vorbis has two main advantages: it allows for more efficient compression than MP3, and it's an open source standard, with no royalties needing to be paid to any party. Quality-wise the format excels at mid-to-lower bitrates, where it is surpassed only by the latest AAC codecs. Its shortcoming is a lack of industry support which makes its future somewhat less certain than MPEG-standardized formats. However, some Ogg Vorbis-capable players have been announced recently, and iRiver has released firmware upgrades for many of its popular players which adds Ogg Vorbis support.

Vorbis has made some progress in being more widely accepted, most notably in video encoding technologies. The Ogg media container (OGM) is gaining popularity in A/V circles where Vorbis is often used to encode audio tracks for use with digital video. The Ogg Vorbis audio format is also increasingly being used in games. The biggest game to date that uses Vorbis is probably Unreal II. Ogg Vorbis is certainly making inroads into being more widely accepted and used.

In the last year, some in the open source community have gotten frustrated with the slow pace of Xiph.org's official development of Vorbis, so they decided to work on it themselves. As a result Vorbis 1.1 is now out, and has bested every other codec out there in the age-old 128Kbps listening test. Without going into too much detail, Vorbis came in first, followed by MPC, iTunes 4.2, LAME, WMA, and finally ATRAC3. Please keep in mind that these tests (with the exception of MPC) are mostly CBR, and that LAME really shines at one of it's more normal, VBR presets. *These results do not scale well at higher bitrates.*

AAC

AAC has been popularized by Apple and backed by numerous companies. Apple's iTunes Music Store makes use of 128Kbps CBR AAC files and the MPEG4 video standard strongly encourages the use of AAC audio in next generation devices and software. Apple and Ahead like calling AAC "MP4" or "m4a" for the obvious reason that people will see the "4" and automatically think it's better than MP3. In many ways this is a good assesment of AAC, however it is confusing because MP4 files can be used for much more than just AAC audio.

AAC is also confusing because it comes in a variety of forms: MPEG4 AAC along with a variety of MPEG2 flavors: AAC, AAC HE, and AAC LC, to name a few. MPEG2 AAC isn't used much anymore (from here on when I refer to AAC, it means MPEG4 AAC). AAC as a whole is newer than MP3, and is progressing far more quickly than MP3 has at any point in its history. AAC lacks the refinement and maturity that MP3 enjoys as a result of efforts of groups like the LAME developers, but it is a much more modern format. One could think of AAC as MP3v2: it's basically MP3 without many of the imperfections, but at the same time it lacks the well-refined codecs that make MP3 a good format.

The vast majority of AAC files are using AAC Low Complexity, or AAC LC. This is the basic form of AAC and the only one with significant hardware support. The other major AAC form being currently pushed is AAC High Efficiency, or AAC+ as it is sometimes referred to by its proponents at Ahead. This form of AAC is similar in principle to the MP3Pro format, in that it consists of a standard AAC file with additional information included to better approximate audio at very low bitrates.

Currently AAC HE cannot be used effectively above 96kbps and is typically used at much lower bitrates. Expect AAC LC to be pushed as a quality-oriented form of AAC and AAC HE pushed as a good way to fit lots of music on portable players. However, as recent listening tests have showed, even at 64 kbps AAC HE can have relatively good quality, exceeding that of all other competitors at such low bitrates.

A little while ago, HydrogenAudio did a codec test which ranked Apple's QT codec best at 128Kbps, but it was CBR-only which held back Ahead's forward-looking VBR-optimized codec. Conversely Ahead's codec took the lead at 64kbps. Both are quite good, but since AAC is heavily licensed, it is not free. Expect to see AAC go far in the embedded and portable markets since DSPs were in the designers' plans when it was created.

MPC

MPC (or MPEG+ or Musepack) is another newcomer on the scene. It's considered better than MP3, and its only real shortcomings are its lack of wide acceptance (its usage is pretty much limited to audiophiles at the moment) and its lack of hardware support. It's smaller than MP3 and considered one step below lossless by many.

MPC uses a radically different approach to encoding than all other codecs save MP2, from which it is descended, so to speak. One could think of it as a VBR MP2 codec with lots of extensions and improvements; it is aimed at perfect quality above all other things. At just 160-170Kbps, it achieves higher quality than all other lossy codecs at any bitrate. There are extremely few known problem samples, and MPC is incredibly fast in both encoding and decoding: moreso than any other modern codec — it encodes at roughly 6x on a PIII-900 and decodes many times faster then even lightweights like MP3.

MPC is basically an audiophile's codec and it's built like one. Tagging is done APE-style or ID3. It also supports [Replaygain](#) neatly, unlike many of the alternatives. MPC is also gapless, unlike AAC or MP3, supporting many channels, 48Khz audio, and 32-bit precision. MPC is at SV7 right now; SV8 is currently in the works, and is expected to more or less complete MPC by adding better multichannel support.

The downside to MPC is that the quality goes downhill below 128Kbps: it's considered MP3-like at best at lower bitrates. Above 128Kbps, nothing can match its quality, save 500Kbps+ lossless formats. Ideally people would use OGG or AAC below 128Kbps, and MPC for anything above. However, lingering concerns over licensing and the fact that it is solely used

by audiophiles will likely preclude it from ever having portable support. Most MPC users choose to archive in MPC and then transcode it's typically flawless encodes to AAC, Ogg, or MP3 for use on portables.

Lossless formats

There are many diehards that will tell you to forget MPC, MP3, AAC, and all the rest. They'll tell you to use a lossless format. What makes these formats different from a traditional lossy format like MP3 is that music duplication is exact. With the MP3 format, some information is removed from the final product. With lossless formats, bitrates are up around 500Kbps (± 300 Kbps), which leads (obviously) to larger file sizes and exact music duplication. Oftentimes, people use lossless formats as a means of archiving their CDs in case the original gets damaged; they will then batch transcode from the lossless archive copy to a lossy format like MP3 for use in their portable music players.

APE, otherwise known as Monkey's Audio, is probably the most popular lossless compression format at this time. I wouldn't be surprised if it starts losing marketshare to FLAC, because of FLAC's affiliation with Ogg: Ogg's popularity is steadily rising, FLAC will probably go with it. FLAC is also an asymmetric codec. APE is a symmetric codec. In other words, APE takes just as long to encode as it does to decode. 10 minutes to pack and unpack. FLAC will always decode fast, but take longer to pack upfront (like MP3) depending on how much you want it to try to squeeze. At the default setting it tends to take twice as long to encode as it does to decode. FLAC is streamable, whereas APE is not.

Apple Lossless Encoding (ALE) will probably become more popular as time goes on for a two major reasons. One, it's built right into iTunes, so it's a fairly easy process to encode directly to it. Secondly, the iPod now supports it, for those people that hate losing audio quality on their portable. ALE also supports tagging like APE and FLAC. ALE is better than AIFF because it's about half the size, with no quality loss.

There are a few people at the Ars HQ who swear by WMA lossless. Arguments over WMA in general aside, WMA lossless is lossless, just like APE, FLAC, and all the rest.

WAV is lossless, of course, but the problem with WAV is that it does not support metadata, and hence lacks any sort of tagging functionality. It is also uncompressed, which leaves one with filesizes that are larger than the same track in either APE or FLAC format.

There isn't much to say about lossless formats: they give you an exact copy of your music. Which you choose depends on how many hoops you want to jump through to encode your music, or if you want to click one button and be done. For those of you who want to use one of the more esoteric formats, there is a nice selection of front-ends for various encoders found [here](#).

Encoding options

Windows options

So you want to rip and encode, which is, after all, the purpose of this article. At this point, we want to cut through all of the extraneous BS and teach you how to rip CDs easily, and encode them to MP3 in the manner most transparent to the majority of people: `--alt-preset standard`.

Ripping a CD is fairly self-explanatory. Ripping is simply the act of extracting the raw audio track from the CD. This results in an uncompressed WAV file. Assuming you have a 60 minute CD, the size of all of the raw tracks should be 605MB — $1411.2\text{Kbps} \times 60\text{ minutes} = 605\text{MB}$.

For better or worse, all rippers function roughly the same. Some rippers are preferred if you have a badly damaged CD, or if you would prefer to encode your audio right after ripping it without using a separate program, or if you want to do on-the-fly encoding as the audio is ripped, thereby skipping the intermediate WAV file.

All of the rippers I have listed can do a few things. All of them can rip audio, can rip and encode audio, or simply encode audio, unless otherwise noted. They all support CDDDB album lookup, though the servers they access may differ. They are all freeware as well, unless otherwise noted.

Exact Audio Copy (EAC)

[EAC](#) has been around for a while, and it has also been the preferred ripper for audiophiles everywhere. The main reason for this is EAC's "secure ripping" mode. What this means is that if there is an imperfection in a CD, EAC will return to the point of the problem, and do its best to accurately reproduce the data on the disc. The number of times that it returns to this problem area depends on what you chose for the "Error recovery/correction quality" setting: low, medium, or high. EAC can take quite a while on a badly damaged track: times of an hour or longer spent on a single track have been reported. While EAC's secure mode is excellent, it cannot reproduce data that just isn't on the disc. If you've got a CD with holes in songs, you cannot expect EAC to recreate the data from nothing; it's not a miracle worker... but it's damn close.

Over time, EAC has evolved into a ripper as well as a front-end for encoding, but EAC's main forte remains ripping. It's not brain-dead simple to use, but it isn't too difficult to figure out, either. Many audiophiles use EAC as their secondary ripper for only those discs which are damaged. For normal use, many use CDex. If you do want to use EAC, and learn the ins and outs of configuring it, an excellent guide can be found [here](#).

A brief note before continuing: many rippers boast a "secure mode" setting or something similar. As of now, the only ripper with EAC's error-correcting power is EAC itself. For those with really badly damaged CDs, accept no substitutes.

CDex

[CDex](#) is probably the second most popular ripper among the audiophiles. Because EAC isn't the easiest program in the world to use, many people choose to use CDex instead. CDex lacks a "secure ripping" mode, but it does have a "Paranoia" setting. Unfortunately, this Paranoia mode isn't well documented. The latest information I could find on it was from 1999. Every ripping test I have looked at has placed EAC's secure mode ahead of CDex's full paranoia setting by a large margin.

Nonetheless, CDex does an ample job of ripping CDs, and for a casual music listener, it's fine, assuming your CDs are in good condition. Interestingly, Sourceforge has the number of official CDex downloads at over 15 million. I'd like to say that 15 million people can't be wrong, but let's face it, AOL has more subscribers than that.

Easy CD-DA Extractor

[Easy CD-DA Extractor](#) is relative newcomer compared to CDex and EAC. It got several positive reviews in the A/V Club when first released, and it's certainly worth a mention here. It's easier to configure than EAC, and it reportedly has error-correcting routines similar to EAC's "secure mode" and CDex's "paranoia" settings. What exactly this means is not well-documented. Easy CD-DA Extractor is available for US\$40 with a 30-day trial.

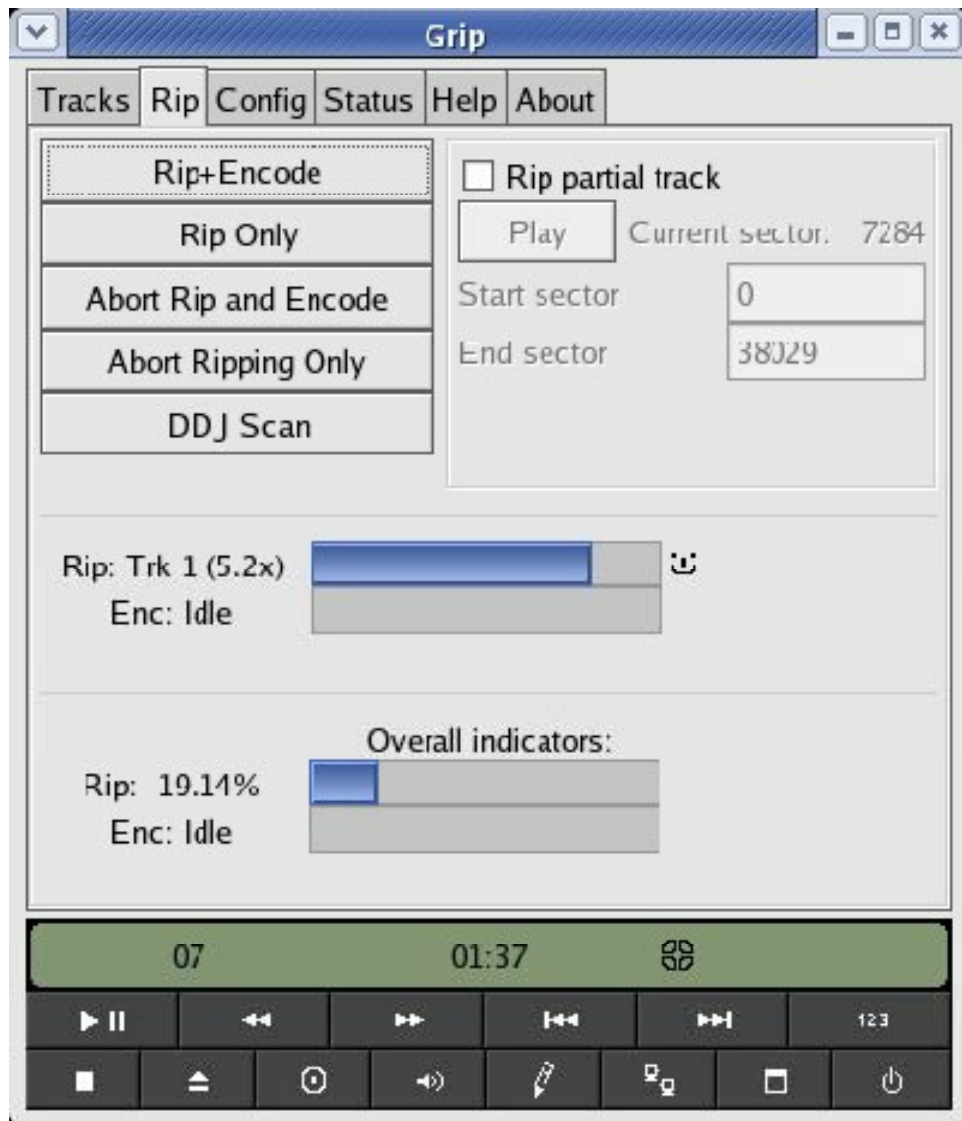
Audiograbber

[Audiograbber](#) is primarily a ripping tool, though it can serve as a front-end for various encoders. Audiograbber used to be crippleware — in order to have a fully functioning copy, one had to pay the licensing fee. It became freeware on February 9th of 2004, and hasn't been updated since. Personally, I'd recommend that you avoid Audiograbber and use EAC, CDex, or Easy CD-DA Extractor, as I think they're better rippers.

Linux

Ripping and Encoding audio in Linux is straightforward, as long as the format you choose is open source. Nearly every distribution includes tools to make encoding into OGG or FLAC work out of the box. Unfortunately, due to patent and licensing conflicts, most free distributions do not include an MP3 encoder (and in some cases decoder) in the base distribution. This makes encoding into more popular format very difficult. Fortunately, encoders like LAME, however, run very well in Linux.

Due to the sheer number of encoders available, most graphical frontends are encoder-independent. This means that they're usable with whatever encoder you wish to use. One of the most popular is grip, which will use LAME, oggenc, or almost any encoder is present on your system:



Many of the popular Linux desktops also include their own native ripper/encoder. If you use KDE, conqueror provides the `audiocd:/kioslave`, which shows the audio CD in konqueror as an audio CD and two folders, an MP3 folder, and an OGG folder. You simply drag the folder you want the audio to be encoded onto your target, and KDE handles the rest, ripping and encoding the audio in the format you chose. For GNOME users, soundjuicer recognizes the CD when it is inserted into the drive, and you choose the available formats from the drop down menu, click OK, and the program will rip and encode the CD.

All three programs use CDDb or freedb.org to find the identity of the CD when it is inserted, so all the ripped music will be tagged properly as it is written.

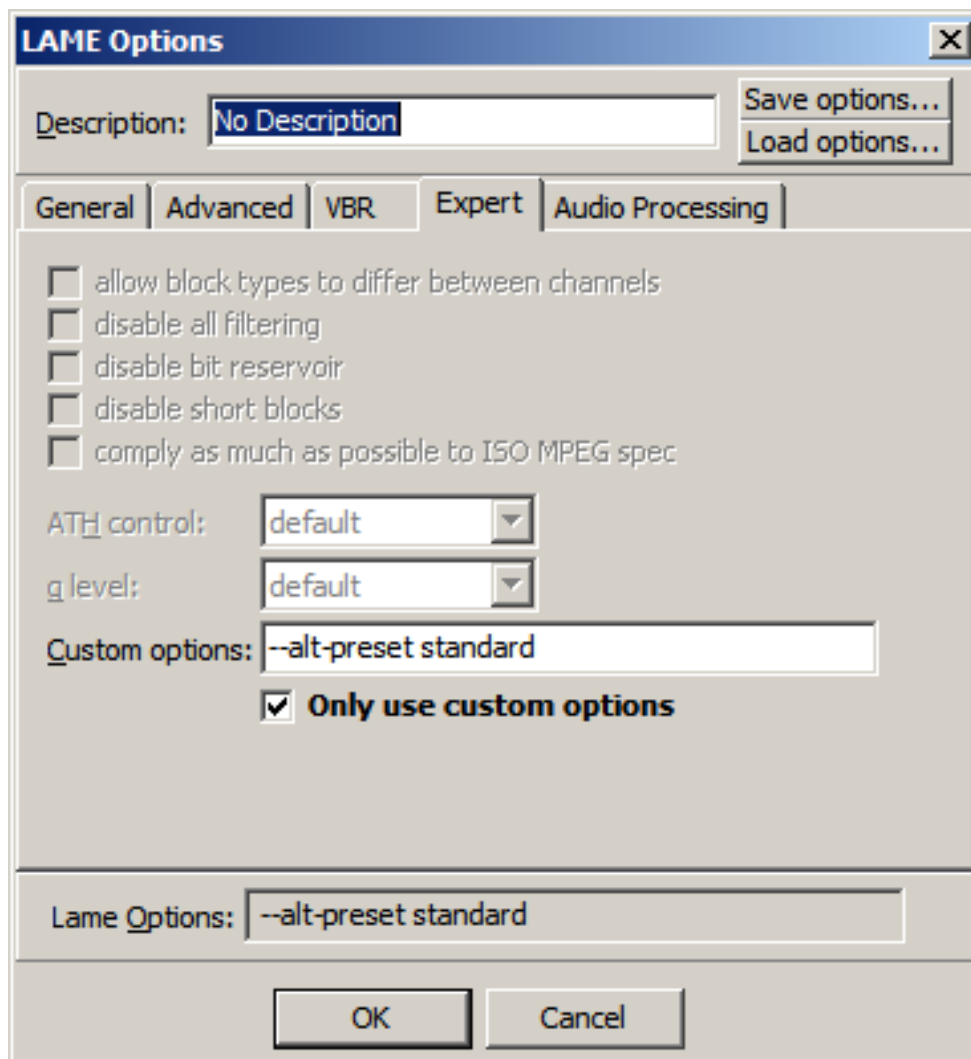
The quick and dirty guide

We have covered CD rippers on Linux and Windows. Regardless of what you choose, a ripper extracts the audio file. Many rippers also offer the ability to encode on the fly, and attempt to make life easier by providing drop down menus and tick boxes for the user to select their preferred bitrate and whatnot. I suggest you stay away from these. Whether or not you use the CDDB functionality on your ripper is up to you, of course. Frankly, I prefer to extract my audio and use a third program to tag it and rename it.

Anyway, once you've got your raw audio extracted (and you're using Windows), snag a copy of [RazorLame](#) if you don't already have it. I like RazorLame because I can queue up hundreds of raw audio tracks to encode quickly and easily for batch encoding runs while I'm asleep or at work or whatever. You can also do this with speeks's [ALL2LAME](#) which also supports tagging and a few other things.

Configuring RazorLame is a simple one-step process. Go to "Edit" → "Options." RazorLame needs to know where the LAME executable is, so navigate to your LAME.exe. This is the only really necessary configuration that RazorLAME needs to work. Feel free to change any of the other options if you so desire.

Once you're done that, go back to the edit menu and this time hit "LAME options." You'll want to go to the "Advanced" tab, and tick the "Only use custom options" box. Enter in "--alt-preset standard," like so:



RazorLame will not forget your settings when you close it, so you don't need to worry about reconfiguring it each time you use it. Once you've finished setting it up, you can drag and drop files or use the "Add" button. Once you're ready to go,

click the "Encode" button. If you no longer want the raw WAV files, tick the checkbox to delete them in the window that pops up once encoding starts. At this point you can walk away.

Tagging

When you're done you will need to tag the files (some people don't tag, they believe that a well-named file negates the need for tagging; it's a personal preference thing). There are various freeware tagging/renaming utilities out there. My personal favorite, though, is [Tag&Rename](#) — for which I actually bought a license; something I am usually loathe to do, unless I love the program in question. The 3.0 beta is the better of the two clients by far. I like the program because of its intuitive interface, CDDB and Amazon.com interoperability, and various other features. I specifically like the Amazon feature because of T&R's ability to snag album art automatically.

Tags allow us to store information about audio files within the file itself. Think of them as the digital version of the info typically written on album covers. However, because they're digital they can store almost anything and make that info available to any software designed to read them.

Different formats have different standards for tags. We'll briefly review these here for clarity.

ID3v1: Originally MP3s did not support tags, and to some extent tagging them at all could be considered a hack. The first attempt at tags was ID3v1. This format supports basic fields of limited length and is included at the end of many MP3 files. Because it is the oldest it is also the most universally supported tag format.

ID3v2: v2 actually represents several different variations of the same tagging system. They include support for Unicode, unlimited tag lengths, and custom fields; all of which are quite welcome given ID3v1's shortcomings. Unfortunately ID3v2 support tends to be inconsistent with some programs reading some fields, depending on which version they implement. Furthermore because the data is stored at the start of the file, v2 tags typically require that the entire audio file be erased and recreated, a process which is time consuming and tedious. Despite these faults, ID3v2 is widely supported, both by popular software programs and in hardware from iRiver, Apple and others.

APEv2: An improvement to the original tag format used on APE lossless files that aims to improve many of ID3v2's shortcomings. Like ID3v2, it supports Unicode and custom fields. However unlike ID3v2, UTF-8 characters are mandated ensuring that non-Latin alphabets are correctly read by any software employing APEv2 tags. Furthermore tags are now written to the end of files allowing them to quickly be edited. APE is used by several different audio formats.

Ogg/FLAC comments: The default tagging scheme used in ogg and FLAC files. Tags are written at the start of the file, and can contain user defined fields and support Unicode.

The above was mostly written with MP3 in mind, because many other formats use their own system of tags. Ogg, for instance, uses its own tagging system and AAC/MP4 audio typically uses a version of tags borrowed from Apple's .mov format. FLAC uses the same system as ogg, but is often hacked to use ID3 or ID3v2 tags as well. Confused yet?

MPC officially supports APEv2 tags, although some software will also hack on ID3v1 and ID3v2 tags. Finally that leaves Monkey's Audio which uses plain APE tags.

Actually tagging your audio files can be accomplished by a variety of means. Obviously the ideal way is to tag during encoding because all metadata is available via CDDB/FreedB and it avoids having to rewrite the entire file in the case of ID3v2/Ogg/FLAC tags. However sometimes we **ahem** acquire improperly tagged files or just need to update files to take advantage of new software, hardware or formats.

Mass tagging can be handled by a variety of players, free programs and commercial software. Things to look for in a tagging program are FreedB support (more in a moment) so that you don't need to type in tags by hand, support for the audio formats you use and support for whichever type of tags you're using. [Tagger](#) is a nice free program based around the command line tag program tag.exe (which ALL2LAME also supports). It's fairly powerful, supports nearly all tag formats and will tag MP3, Ogg, MPC, APE and FLAC files. Another good program is Tag&Rename, though it is shareware. It supports MP3, Ogg, WMA, and FLAC tags. Unfortunately it will not do APE tags, however it is very easy to use and extremely effective. Finally some audio players make effective mass-taggers. Apple's iTunes is another option if you're not running Linux. Foobar2000 is another that incorporates FreedB support as well as the flexibility to tag virtually all modern tags and formats.

Finally a quick word on freedb. A good tagger can look through the lengths of all the songs in an album and guess the needed tag info based on the data in FreedB. The caveat is that you need to have all files and you need to know their proper order on the CD. Some software like Tag&Rename goes a step further and can even search web sites for info and pull that into tags. (As well as pulling album art if so desired.)

Conclusion

The best method of audio encoding really depends on what you are doing with your music. If you have a portable music player, your best bet is MP3. If you're somewhat space-limited, but you don't have to worry about your files working with a portable player, MPC is probably your best bet.

If you want quality above all else, your best bet is FLAC, APE, WMA lossless, or Apple Lossless Encoding. Sometimes, those who are particularly picky will have two copies of their music: a lossless copy, and an MP3 version for their portable. There are as many solutions as there are problems. Personally, I use LAME with the --alt-preset standard switch. I like the quality, size, and usability in my portable. You might find that something different is better for you.

This is merely *a guide*, not *the guide* to encoding your music. The digital music landscape changes quickly, particularly in the open source circles (e.g., Vorbis and LAME). In another two years, the contenders will likely be as different from today as today is from the last time Ars visited this topic. Hopefully at some point it will get simpler rather than more complex. But when does that ever happen?